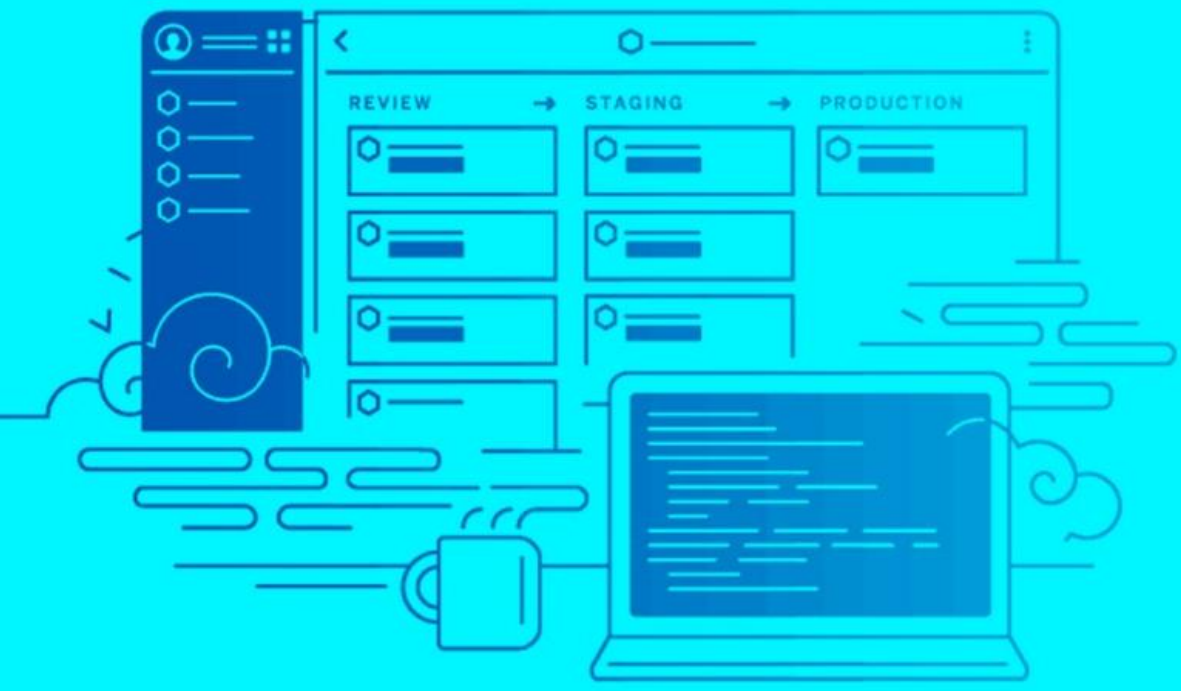


PENGEMBANGAN Aplikasi CLOUD HEROKU

Dian Resha Agustina | Onno Widodo Purbo

Undang-undang Republik Indonesia Nomor 28 tahun 2014 tentang Hak Cipta
Lingkup Hak Cipta
Pasal 1
Hak Cipta adalah hak eksklusif pencipta yang timbul secara otomatis berdasarkan prinsip deklaratif setelah suatu ciptaan diwujudkan dalam bentuk nyata tanpa mengurangi pembatasan sesuai dengan ketentuan peraturan perundang-undangan.
Ketentuan Pidana Pasal 113
(1) Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (seratus juta rupiah).
(2) Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).
(3) Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp 1.000.000.000,00 (satu miliar rupiah).
(4) Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp 4.000.000.000,00 (empat miliar rupiah).

PENGEMBANGAN
Aplikasi
CLOUD HEROKU

Dian Resha Agustina
Onno Widodo Purbo

PUSAKA MEDIA

Perpustakaan Nasional RI:
Katalog Dalam Terbitan (KDT)

PENGEMBANGAN APLIKASI CLOUD HEROKU

Penulis:
Dian Resha Agustina
Onno Widodo Purbo

Desain Cover & **Layout**
Pusaka Media Design

vi + 278 hal : 15.5 x 23 cm
Cetakan, Januari 2023

ISBN: 978-623-418-174-6

Penerbit
PUSAKA MEDIA
Anggota IKAPI
No. 008/LPU/2020

Alamat
Jl. Endro Suratmin, Pandawa Raya. No. 100
Korpri Jaya Sukarame Bandarlampung
082282148711
email : cspusakamedia@yahoo.com
Website : www.pusakamedia.com

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan kami kemudahan sehingga kamiaya dapat menyelesaikan buku ini. Tanpa pertolongan-Nya tentunya kami tidak akan sanggup untuk menyelesaikan buku ini dengan baik.

Penulis mengucapkan syukur kepada Allah SWT atas limpahan nikmat sehat-Nya, baik itu berupa sehat fisik maupun akal pikiran, sehingga penulis mampu untuk menyelesaikan buku ini dengan judul "Pengembangan Aplikasi Cloud Heroku (Panduan komprehensif untuk membantu Anda membangun, menyebarkan, dan memecahkan masalah aplikasi cloud secara tepat menggunakan Heroku)".

Penulis tentu menyadari bahwa buku ini masih jauh dari kata sempurna dan masih banyak terdapat kesalahan serta kekurangan di dalamnya. Untuk itu, penulis mengharapkan kritik serta saran dari pembaca. Kemudian apabila terdapat banyak kesalahan pada buku ini penulis mohon maaf yang sebesar-besarnya.

Demikian, semoga buku ini dapat bermanfaat. Terima kasih.

DAFTAR ISI

KATA PENGANTAR ..... **v**
DAFTAR ISI ..... **vi**
BAB 1 PERKENALAN HEROKU 1
BAB 2 ISI HEROKU 24
BAB 3 MEMBANGUN APLIKASI HEROKU 50
BAB 4 MENYEBARKAN HEROKU APLIKASI 81
BAB 5 MENYATUKAN SEMUANYA 116
BAB 6 PRAKTEK HEROKU 145
BAB 7 KEAMANAN HEROKU 193
BAB 8 MENGATASI MASALAH APLIKASI HEROKU 218
BAB 9 PENGGUNAAN HEROKU TINGKAT LANJUT 250

BAB 1
PERKENALAN HEROKU

Kemajuan terbaru dalam teknologi ditambah dengan permintaan untuk komputasi yang efisien biaya telah menyebabkan pertumbuhan besar penggunaan komputasi awan dalam bisnis modern. Pengguna ingin mengoptimalkan penggunaan sumber daya dari CPU, jaringan, dan memoridan hanya membayar apa yang mereka gunakan. Virtualisasi, kekuatan komputasi yang lebih cepat / lebih besar, dan tulang punggung jaringan berkecepatan tinggi telah menyebabkan ledakan dalam penyebaran infrastruktur cloud di seluruh dunia. Seiring dengan infrastruktur yang mendasarinya, banyak platform telah berevolusi untuk mendukung kemampuan untuk mengembangkan aplikasi nyata untuk lingkungan cloud secara virtual dari mana saja. Selain itu,perangkat lunak berbasis cloud baruaplikasitelah berkembang begitu cepat sehingga mereka telah menggantikan gagasan perangkat lunak yang diinstal untuk selamanya.

Saat ini, tidak ada pengembang yang tidak tersentuh oleh cloud. Dalam kehidupan sehari-hari mereka, pengembang menggunakan satu atau bentuk lain dari server cloud – apakah itu mesin virtual yang di-hosting Amazon untuk melakukan pengujian atau lingkungan pengembangan berbasis cloud untuk menulis kode untuk aplikasi bisnis mereka. Cloud ada di mana-mana.

Dalam buku ini, kami akan mengeksplorasi aspek komputasi awan yang sangat signifikan dan spesifik, yaitu bagaimana

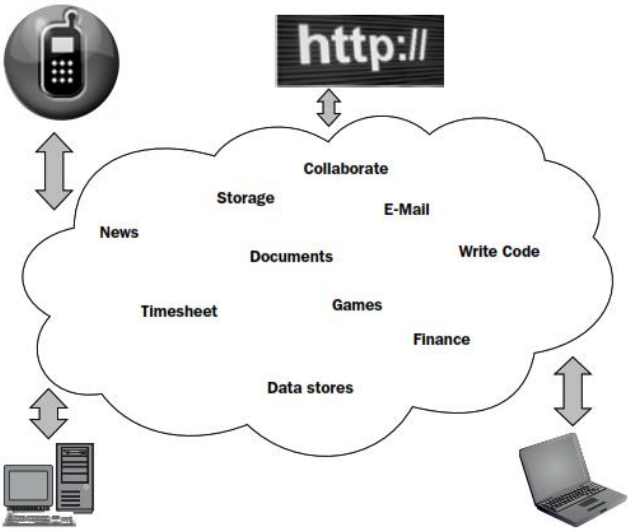
mengembangkan aplikasi web di cloud. Kami akan menggunakan platform Heroku untuk membangun aplikasi web yang kuat dan dapat diskalakan dan dalam proses memahami berbagai aspek platform Heroku.

Tidak efesien? Mari kita evaluasi. Dalam bab ini, kita akan:

1) Menentukan komputasi awan dan memahami berbagai komponennya
2) Memahami apa pengembangan aplikasi cloud dan apa keunggulannya
3) Memperkenalkan Anda ke Heroku dan menelusuri sejarahnya• Mengenal
4) Meninjau fitur-fitur arsitektur Heroku high level
5) Pelajari cara menginstal Heroku
6) Test drive Heroku

Apa itu cloud computing?

Sederhananya, komputasi awan adalah bentuk komputasi di mana pengguna mengakses sumber daya komputasi apa pun dari jarak jauh melalui klien sederhana, yang dalam kebanyakan kasus adalah browser web. Sumber daya ini dapat berupa aplikasi perangkat lunak atau sistem operasi atau perangkat keras yang terletak dari jarak jauh. Cloud computing adalah perwujudan dari keinginan untuk mengoptimalkan penggunaan sumber daya komputasi bersama dengan menciptakan infrastruktur yang memungkinkan Anda menggunakan daya komputasi, penyimpanan, dan jaringan secara optimal dan hanya membayar apa yang Anda gunakan.



Cloud model server

Server yang ditawarkan oleh cloud computing dibagi lagi menjadi server yang berbeda model : infrastructure as a service (IaaS), platform as a service (PaaS), dan software as a service (SaaS). Klasifikasi ini dilakukan untuk memisahkan berbagai jenis server yang dapat dibeli pengguna untuk memenuhi kebutuhan bisnis mereka dalam lingkungan komputasi awan.

IaaS adalah model server cloud yang memungkinkan penyediaan sumber daya perangkat keras virtual oleh pengguna sesuai permintaan. Secara fisik, sumber daya ini dapat tersebar di beberapa pusat data, yang dikelola oleh penyedia server. Sumber daya ini termasuk penyimpanan virtual, koneksi jaringan, dan load balancers untuk sumber daya perangkat keras yang disediakan. Pengguna dapat menggunakan sumber daya sesuai permintaan dan membayar per penggunaan. Jika pengguna membutuhkan lebih banyak sumber daya, penyedia memiliki kemampuan untuk secara otomatis meningkatkan perangkat keras sesuai dengan kebutuhan dan sebaliknya. Contoh yang baik dari penyedia IaaS adalah Amazon Web Services (AWS-http://aws.amazon.com). Ini adalah penyedia IaaS paling popular di cloud. Rackspace (http://www.rackspace.com) adalah contoh lain.

PaaS adalah model server cloud yang menyediakan alat untuk membangun aplikasi perangkat lunak di cloud. Analogi yang dekat adalah dengan melihat PaaS sebagai sistem operasi dan middleware dari lingkungan cloud. PaaS memberi pengembang platform yang mendasari untuk digunakan untuk mengembangkan aplikasi mereka. Dibutuhkan perhatian untuk mendukung bahasa atau teknologi tertentu yang ingin digunakan oleh pengembang tumpukan. Banyak penyedia PaaS juga memungkinkan penskalaan berdasarkan permintaan dari sumber daya komputer dan penyimpanan yang mendasarinya, secara otomatis, untuk membebaskan pengguna cloud dari pekerjaan mengalokasikan sumber daya secara manual. Di PaaS, konsumen server mengontrol penyebaran dan konfigurasi. Penyedia PaaS menyediakan server, jaringan, dan kebutuhan komputasi aplikasi perangkat lunak. Model PaaS juga memungkinkan arsitektur multitenant sehingga banyak pengguna dapat menggunakan aplikasi web dengan cara yang aman, terukur, bersamaan, dan gagal-aman. Solusi PaaS yang canggih juga menyediakan lingkungan pengembangan aplikasi web terintegrasi, yang memfasilitasi pengkodean kolaboratif, kontrol sumber, dan penyebaran. Heroku (http://www.heroku.com) dan Google App Engine (http://cloud.google.com/AppEngine) adalah dua contoh platform PaaS yang sukses.

Model cloud SaaS menyediakan perangkat lunak yang dapat Anda konsumsi dari lingkup peramban web Anda. Tidak perlu untuk instalasi yang rumit dan memakan waktu. Buka browser, arahkan ke URL, dan gunakan aplikasi yang ditunjuk oleh URL. Apa yang terjadi di balik layar semuanya tersembunyi dari pengguna. SaaS telah berkembang pesat dalam dekade terakhir. Banyak penyedia SaaS telah membuat perangkat lunak desktop atau yang dihosting secara lokal menjadi usang. Yang Anda butuhkan hanyalah browser dan Anda siap menggunakan aplikasi apa pun untuk melakukan apa saja. Tidak ada sakit kepala karena peningkatan perangkat lunak, ketidakcocokan versi, atau portabilitas perangkat lunak. Server Gmail Google (http://gmail.com) adalah salah satu implementasi SaaS yang paling sukses dan dikenal luas. Komponen SaaS telah tumbuh secara eksponensial dengan perusahaan memanfaatkan

infrastruktur dan platform yang mendasari untuk membangun versi cloud dari sebagian besar penawaran produk perangkat lunak mereka. Pada tahun 2013, hampir setiap perusahaan telah memiliki versi SaaS dari aplikasi perangkat lunak populer yang tersedia untuk pelanggan online.

Apa itu pengembangan aplikasi cloud?

Salah satu cara menggambarkan pengembangan aplikasi cloud adalah kemampuan untuk membuat, membangun, dan menggunakan aplikasi perangkat lunak Anda dari browser web. Anda tidak perlu menginstal apa pun di mesin lokal Anda. Yang Anda butuhkan hanyalah browser web, koneksi Internet, dan kemampuan kode. Platform Cloud9 IDE adalah salah satu contoh lingkungan pengembangan aplikasi cloud berbasis web. Platform pengembangan IDE Cloud9 memungkinkan pengembang menulis, membuat, menguji, dan secara instan menggunakan aplikasi web mereka dari browser. Setelah digunakan, pengembang dapat mengakses aplikasi web menggunakan URL web.

Ada pendekatan lain meskipun hybrid untuk melakukan pengembangan aplikasi cloud. Dalam pendekatan ini, pengembang menulis kode pada mesin mereka sendiri, menggunakan alat pengembangan yang dipasang secara lokal, membangun kode mereka secara lokal terlebih dahulu, dan setelah diuji, mereka menyebarkan aplikasi web tersebut ke layanan platform di cloud. Platform sebagai layanan di cloud membuat kode lagi menggunakan alat bantu yang didukung dan menyebarkan aplikasi web ke server yang relevan. Kemudian, ia mengembalikan URL web ke pengembang untuk mengakses aplikasi web.

Keuntungan utama pengembangan aplikasi cloud

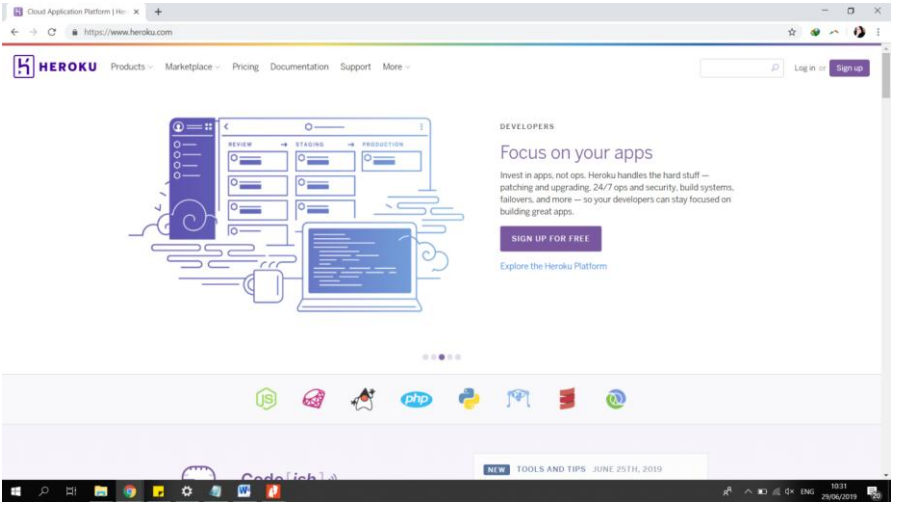
Mengembangkan aplikasi web menggunakan model pengembangan aplikasi cloud adalah keuntungan bagi semua pengembang. Ini adalah tongkat ajaib yang tidak bisa dibayangkan oleh para pengembang sebelum munculnya komputasi awan dan platform pengembangan berbasis cloud. Ada beberapa keuntungan menggunakan pengembangan aplikasi cloud:

- Salah satu keuntungan utama pengembangan aplikasi cloud adalah tidak diperlukan instalasi perangkat lunak padamesin lokal paradigma. Anda hanya perlu koneksi Internet, browser web, dan kemampuan membuat kode untuk menulis aplikasi web. Sebagai pengembang, Anda dapat fokus membangun fungsi aplikasi web dengan cara sebaik mungkin tanpa khawatir dengan editor kode, pengadu, atau alat bantu pembangunan / penyebaran yang akan digunakan. Semua ini bukan urusanmu lagi. Anda cukup menulis kode Anda dan sisanya akan diurus untuk Anda. Bahkan dalam pendekatan hibrida, beberapa platform yang memungkinkan pengembangan aplikasi cloud memberi Anda alat yang kompatibel (kompatibel dengan alat di lingkungan cloud) yang dapat Anda gunakan untuk membangun, menguji, dan menggunakan aplikasi Anda.
- Keuntungan praktis lain dari menggunakan pengembangan aplikasi cloud adalah kenyataan bahwa Anda tidak perlu mengelola infrastruktur perangkat lunak, penyimpanan, atau perangkat keras yang digunakan. Perangkat lunak pendukung (sistem operasi, alat pengembangan, dan sebagainya) secara otomatis ditingkatkan dan perangkat keras baru disediakan secara transparan untuk Anda; Anda tidak perlu khawatir tentang versi kompiler atau perpustakaan pihak ketiga yang Anda gunakan.
- Pengembangan aplikasi cloud juga berarti mengurangi biaya membangun dan memelihara aplikasi web Anda. Anda membayar saat Anda pergi dan mendapatkan tagihan hanya untuk apa yang Anda gunakan. Pengembang dapat mengoptimalkan investasi mereka dengan menggunakan perangkat yang tepat dan memanfaatkan perangkat sumber terbuka yang tersedia untuk membangun aplikasi web yang kuat, dapat diukur, dan berkinerja baik.
- Mobilitas adalah alasannya, yang dengan sendirinya memberikan keuntungan luar biasa bagi pengembang yang ingin membuat aplikasi saat bepergian. Saat Anda dapat mengembangkan, mengubah, atau menggunakan aplikasi web dari browser web perangkat seluler Anda, itu adalah gagasan

yang kuat. Anda bisa menyeruput kopi di kedai kopi lokal, menanggapi permintaan pelanggan melalui perangkat seluler Anda, menaikkan / menurunkan server Anda, menambahkan pengembang baru ke aplikasi Anda, atau menambah memori yang diperlukan untuk aplikasi web Anda. Kemungkinannya tidak terbatas.

Memperkenalkan Heroku

Heroku (http://www.heroku.com), diucapkan-oh-koo, adalah salah satu penyedia PaaS terkemuka dalam bisnis perangkat lunak cloud, yang membuktikan diri untuk menjadi solusi PaaS terkemuka untuk perusahaan kecil dan besar. Dengan peningkatan yang konsisten dan filosofi "kenyamanan" di atas "konfigurasi", Heroku telah menjadi platform pengembangan aplikasi cloud terkemuka untuk para pengembang, yang telah digunakan oleh lebih dari 40.000 situs web hingga saat ini. Filosofi Heroku adalah untuk membiarkan pengembang hanya berfokus pada penulisan aplikasi web dan melupakan server. Heroku secara ajaib menangani pembangunan, penggelaran, pengoperasian, dan penskalaan aplikasi untuk pengembang sesuai permintaan.



Heroku adalah polyglot platform aplikasi cloudyang memberikan fleksibilitas luar biasa dalam memilih bahasa pemrograman yang sesuai untuk mengembangkan aplikasi web. Heroku menyediakan dukungan platform untuk Ruby, Ruby on Rails, Java, Node.js, Clojure, Scala, Python, dan PHP pada awal 2013.

Arsitektur add-on Heroku memungkinkan pengembang untuk menyesuaikan penggunaan berbagai paket pihak ketiga berdasarkan kebutuhan. Anda memiliki fleksibilitas untuk memilih paket dasar atau premium berdasarkan persyaratan situs web Anda. Pengembang dapat menambah biaya aplikasi dengan sumber daya tambahan, seperti Memcached untuk caching data atau database NoSQL dan membuat aplikasi web yang benar-benar kuat dan kaya fitur.

Heroku menyediakan banyak fleksibilitas dalam mengelola aplikasi Anda begitu Anda telah menggunakannya. Dengan Heroku, pengembang dapat mengelola aplikasi menggunakan alat baris perintah Heroku yang berjalan di mesin klien atau dasbor yang berjalan di infrastruktur Heroku.

Heroku sangat bisa diukur. Ini berskala secara transparan karena lonjakan lalu lintas Anda dan dapat melayani aplikasi dengan lebih dari 103 permintaan berkelanjutan per detik hari ini.

Alur kerja Heroku Git yang terfokus memudahkan berbagi kode, berkolaborasi dengan pengembang lain, dan sering menggunakan kode, sehingga mengurangi waktu yang dibutuhkan aplikasi untuk menjangkau para penggunanya.

Sejarah Heroku

Sangatlah penting untuk memahami sejarah sesuatu untuk memahami bagaimana ia berevolusi ke kondisi saat ini. Selama bertahun-tahun, para desainer Heroku telah membuat beberapa pilihan untuk menjadikan Heroku seperti yang kita lihat hari ini. Heroku telah mengalami banyak iterasi evolusi. Semuanya berawal ketika beberapa insinyur web berkumpul dan membangun sebuah platform sebagai layanan yang sangat mirip dengan platform Unix. Pengembang dapat membangun aplikasi mereka di Ruby dan mendorongnya ke platform untuk hosting. Semua layanan bernilai

tambah seperti pemantauan, pencatatan, atau basis data dapat digunakan dan mudah digunakan. Itu adalah mimpi pengembang web menjadi kenyataan.

Berikut ini adalah ringkasan singkat dari sejarah Heroku yang menuntun Anda melalui evolusi Heroku selama beberapa tahun terakhir:

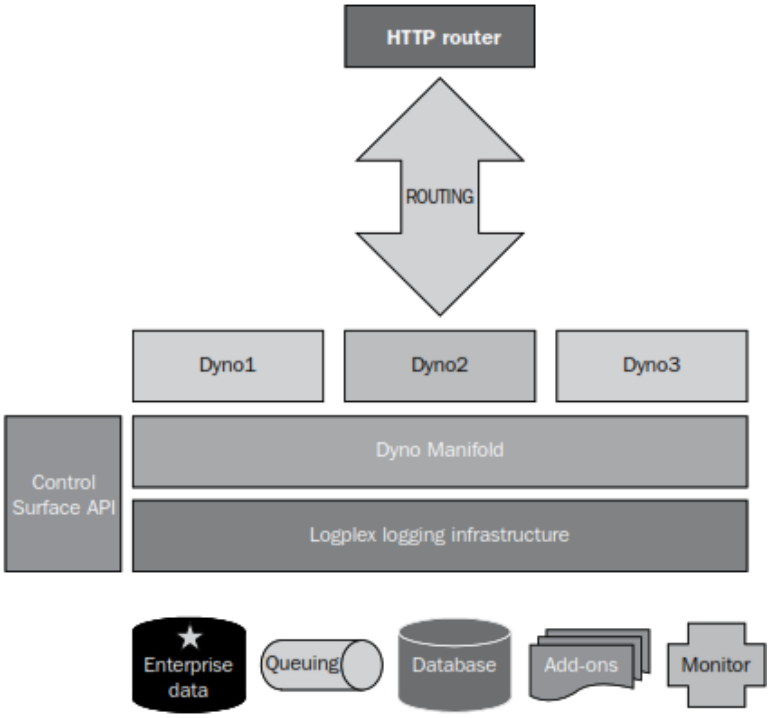
- Heroku diluncurkan pada 2007 oleh James Lindenbaum, Adam Wiggins, dan Orion Henry.
- Heroku adalah startup Y-combinator dan basisnya meningkat menjadi lebih dari 2.000 aplikasi dan pengguna hanya dalam enam bulan pertama.
- Heroku mulai dengan mengatasi dua masalah umum untuk pengembang web: penyebaran aplikasi dan produktivitas pengembang.
- Pada akhir 2007, Heroku datang dengan fitur menarik seperti "langsung hidup" (penyebaran), "buat dan edit online" (mengedit kode sumber online), dan "berbagi dan berkolaborasi" (berbagi kode).
- Beberapa waktu kemudian di 2008, Heroku diproyeksikan sebagai kerangka pengembangan web berbasis Rails dan lebih banyak fokus ditempatkan pada penyebaran dan penskalaan aplikasi web.
- Berbagi dan mengembangkan kode kolaboratif melalui Git membuka jalan untuk peningkatan minat pada Heroku oleh komunitas pengembang. Segera API terpadu menjadi bagian dari penawaran Heroku.
- Selama 2009, Heroku datang dengan Herokugarden – aplikasi tempat di mana pengembang dapat membuat, menyebarkan, dan mengelola aplikasi web. API (perintah Heroku) untuk mengelola aplikasi web adalah bagian dari rangkaian ini. Fitur ini segera dihapus.
- Heroku juga menambahkan konsep "add-on" –bagian perangkat lunak (perpustakaan) yang secara ajaib dapat Anda tambahkan ke aplikasi web dan hampir digunakan dengan cepat. Pada akhir 2009, Heroku memiliki hampir 20.000 aplikasi yang berjalan di berbagai platform termasuk ponsel.

- Heroku diakuisisi oleh Salesforce pada bulan Desember 2010.
- Dalam beberapa tahun terakhir, antara lain Heroku telah melakukan beberapa perubahan penting pada platform, termasuk membawa tumpukan Celadon Cedar baru dan mendukung Java, Clojure, Python, Node.js, dan Bahasa scala. Itu juga menambahkan dukungan lanjutan untuk database Postgres.

Tinjauan umum arsitektur Heroku

Arsitektur Heroku terdiri dari tumpukan platform yang terdiri dari runtime bahasa, berbagai perpustakaan, sistem operasi, dan infrastruktur yang mendasari untuk mendukung pengembangan aplikasi web yang dapat diskalakan.

Arsitektur tingkat tinggi dari platform Heroku ditampilkan sebagai berikut:



Manajemen proses

The dyno manifold (DM) adalah blok dasar untuk lingkungan eksekusi pada platform Heroku. Ini adalah lingkungan eksekusi yang terdistribusi, toleran terhadap kesalahan, dan dapat diskalakan secara horizontal untuk aplikasi. Ini mengelola seluruh keragaman contoh tipe proses melalui model proses dengan cara otomatis tanpa biaya perawatan.

Unit kerja dalam platform Heroku disebut dyno. Dyno adalah wadah yang sepenuhnya terisolasi, sangat tersedia (virtual) yang berjalan di dyno manifold .

Dynos menerima permintaan web dari routing mesh, menyambung ke sumber daya aplikasi seperti database, misalnya, menggunakan variabel lingkungan, dan menulis output ke pesan log yang disebut Heroku Logplex.

Jenis proses mendefinisikan template yang akan digunakan untuk instantiate proses tertentu. Itu adalah deklarasi sebuah perintah. Perintah dijalankan ketika sebuah dyno dari jenis proses itu dimulai. Setidaknya ada dua jenis proses yang tersedia di Heroku – web dan pekerja. Sebuah instance dari tipe proses web biasanya menangani permintaan klien HTTP. Router mengarahkan semua permintaan ke jenis proses web. Mesin virtual dari jenis proses pekerja digunakan untuk menjalankan tugas-tugas lain seperti pekerjaan kustom dari pekerjaan latar belakang yang sudah berjalan lama dan tugas-tugas antrian. Heroku juga menyediakan fleksibilitas untuk membuat jenis proses tambahan berdasarkan kebutuhan spesifik.

Suatu proses adalah turunan dari jenis proses tertentu. The Procfile menentukan berbagai jenis proses dan bagaimana menjalankannya pada platform Heroku.

Logging

Logplex Rutestream log yang berasal dari berbagai sumber, seperti tugas aplikasi, komponen sistem, dan layanan backend ke kumpulan output tunggal. Filter tambahan dapat digunakan untuk mencari pesan log tertentu, karenanya menyediakan fasilitas logging yang fleksibel.

Routing HTTP

Permintaan (dashed vertical lines) untuk sumber daya web (misalnya, halaman web) dialihkan ke dyno proses web yang sesuai menggunakan Routing mesh. Permintaan yang masuk diterima oleh penyeimbang beban yang secara otomatis merutekan permintaan HTTP ke dyno tertentu melalui mesh ini. Routing mesh bertanggung jawab untuk menentukan lokasi web dyno aplikasi dalam manifold dyno dan meneruskan permintaan HTTP ke salah satu dari dyno ini.

Interface Heroku

Heroku menyediakan berbagai permukaan kendali seperti manajemen proses, perutean, pencatatan, penskalaan, konfigurasi, dan penyebaran untuk membangun dan mengoperasikan aplikasi. Ini tersedia sebagai antarmuka baris perintah (CLI), konsol berbasis web, serta REST API lengkap. Permukaan kontrol ini memberikan pengembang aplikasi fleksibilitas untuk mengontrol berbagai aspek aplikasi melalui beberapa titik sentuh.

Kumpulan fitur Heroku

Heroku adalah PaaS berfitur lengkap yang menyediakan setumpuk lengkap fitur PaaS "benar". Berikut ini adalah beberapa fitur inti yang tersedia di Heroku:

1) Fitur runtime Heroku:
   - Memungkinkan kontrol proses yang fleksibel; proses web dan pekerja (jenis) yang dapat diskalakan (disebut dyno) dijalankan dari Procfile.
   - Hal ini memungkinkan jenis proses baru untuk didefinisikan seperti web atau pekerja jenis.
   - Proses isolasi – apa pun yang Anda simpan di proses web Anda akan diisolasi dari semua proses web lainnya.
   - Heroku bahkan memungkinkan Anda menjalankan proses selama mode pemeliharaan sambil melayani halaman statis pengguna aplikasi.
   - Ini memberikan dukungan multi bahasa yang berfungsi penuh.

2) Konfigurasi:
   - Heroku tidak menggunakan file properti ataukonfigurasi hardcoded variable untuk membaca parameter global spesifik sistem atau aplikasi; sebagai gantinya, ia menggunakan variabel konfigurasi atau variabel lingkungan.
   - Heroku lebih memilih kenyamanan daripada konfigurasi, oleh karena itu arsitektur konfigurasi sengaja disederhanakan untuk digunakan.
3) Effective releases:
   - Setiap kali ada kode baru, add-on, atau konfigurasi, Heroku membuat rilis baru secara otomatis.
   - Heroku menyediakan serangkaian kemampuan untuk mengelola rilis aplikasi mulai dari membuat rilis dan mendaftar rilis yang tersedia untuk memutar kembali ke rilis sebelumnya.
4) Logging :
   - Fasilitas Logplex Heroku memberi Anda pandangan agregat dari perilaku runtime aplikasi Anda.
   - Heroku juga memiliki ketentuan untuk memfilter catatan log berdasarkan sumber, proses, atau keduanya.
5) Keamanan :
   - Heroku menyediakan kemampuan untuk mengatur titik akhir SSL untuk aplikasi Anda untuk memungkinkan komunikasi yang aman ke aplikasi yang Anda gunakan.
   - Heroku juga mendukung kunci SSH untuk memungkinkan transferaman kode pengembang yangke / dari platform.
6) Status real-time :
   - Melalui URL web, Heroku memberi para penggunanya kemampuan untuk memverifikasi status aplikasi yang sedang berjalan berdasarkan rentang waktu. Anda dapat kembali pada waktunya untuk memeriksa kegagalan dan langkah-langkah yang diambil untuk resolusi mereka.

7) Penempatan berbasis Git :
   - Heroku menggunakan Git sebagai metode utama untuk menyebarkan aplikasi. Git adalah sistem kontrol revisi kode sumber terbuka yang memungkinkan akses bersama ke lingkungan kode sumber yang dikelola.

   Sementara mengelola aplikasi untuk platform Heroku menggunakan Git, pengembang dapat melakukan hal berikut :
   - Membangun dan melacak aplikasi
   - Buat repositori kode remote
   - Menyebarkan aplikasi di beberapa lingkungan (pengembangan, pementasan)
   - Gunakan sistem kontrol versi lain sisi dengan sisi dan Git hanya untuk penyebaran
8) Platform Polyglot :
   - Meskipun Heroku mulai menjadiaplikasi web khusus-Ruby platform, platform ini telah berkembang selama bertahun-tahun dan mulai mendukung banyak bahasa pemrograman populer lainnya termasuk Java, Node.js, Scala, Clojure, Python , dan kerangka kerja Play termasuk rilis terbaru.
9) Buildpacks :
   - Dukungan Heroku untuk bahasa diaktifkan dengan membuat buildpack untuk bahasa itu. Buildpack adalah sekumpulan skrip yang diperlukan untuk mengidentifikasi bahasa kode sumber dan memberikan instruksi untuk membuatnya menjadi kode yang dapat dieksekusi. Misalnya,Ruby (buildpackhttps://github.com/ heroku / heroku-buildpack-ruby) adalah salah satu buildpack yang paling umum digunakan di Heroku.
10) Add-ons :
   - Heroku menawarkan semakin banyak add-on melalui penyedia add-on program. Layanan tambahan, seperti pelacakan kesalahan, pelaporan, layanan email, database NoSQL yang di-host, dan pencarian teks lengkap di antaranya tersedia secara instan melalui beberapa klik atau perintah pada prompt CLI Heroku.

- Ada ketentuan untuk menambah dan menghapus add-on menggunakan API Heroku.

11) Alat baris perintah Heroku (CLI):
    - Alat baris perintah Heroku adalah antarmuka untukweb Heroku API. Alat ini menyediakan antarmuka baris perintah yang mudah digunakan untuk melakukan hal-hal seperti membuat / mengganti nama aplikasi, menjalankan proses satu kali, mengambil cadangan, dan mengonfigurasi pengaya.
    - Alat ini biasanya diinstal dengan program toolbelt. Ini juga memiliki arsitektur plugin yang memungkinkan pengembang untuk memperluas fungsionalitas perintah sesuai kebutuhan.
    - Hal ini sengaja disimpan mirip dengan UNIX shell perintah untuk mengurangi kurva belajar untuk administrator sistem atau pengguna API Heroku.
12) Platform API berfitur lengkap
    - Heroku Platform API adalah daftar fungsi standar yang memungkinkan Anda untuk memanggil backend Heroku secara terprogram dan melakukan berbagai operasi seperti membuat aplikasi atau menghapusnya. Ini membantu pengembang mendapatkan kontrol penuh atas aplikasi.
13) Dikelola, arsitektur multitenant:
    - Platform arsitektur Heroku menyediakanproses dikelola lingkungan eksekusidan tingkat tinggi isolasi antara aplikasi berjalan, yang semuanya transparan untuk aplikasi client.

Memulai Heroku

Sekarang setelah kita memiliki pemahaman dasar tentang apa itu Heroku dan apa yang ditawarkan untuk pengembang yang berencana membangun dan menggunakan aplikasi berbasis cloud, mari kita mulai gunakan Heroku. Anda akan menyukai bagian ini.
Anda perlu mempersiapkan syarat-syaratnya sebagai berikut:

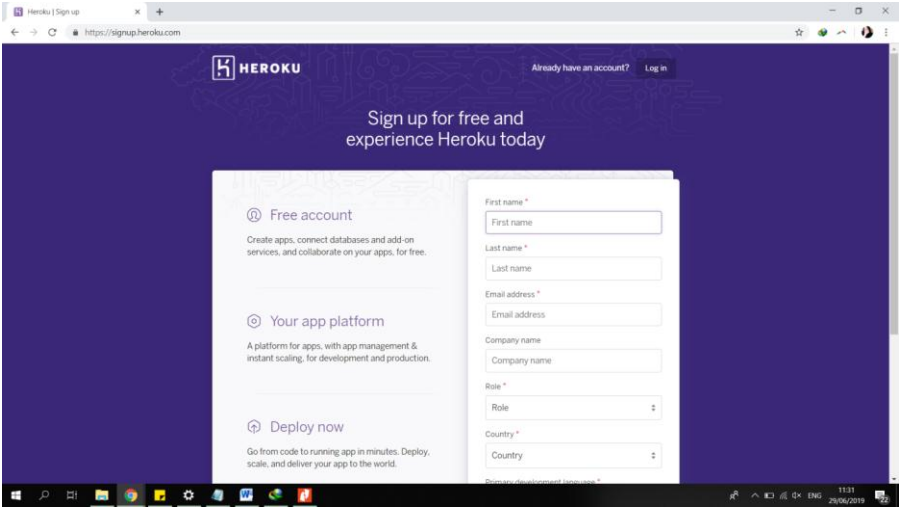
1. Dapatkan akun Heroku (https://www.heroku.com).
2. Instal klien Heroku toolbelt (https://toolbelt.heroku.com/).
3. Siapkan SSH untuk akun pengguna Anda.

Kunci SSH adalah token terenkripsi yang digunakan oleh mesin Anda dan akun Heroku Anda untuk memvalidasi keaslian pengguna yang meminta perintah pada platform Heroku.

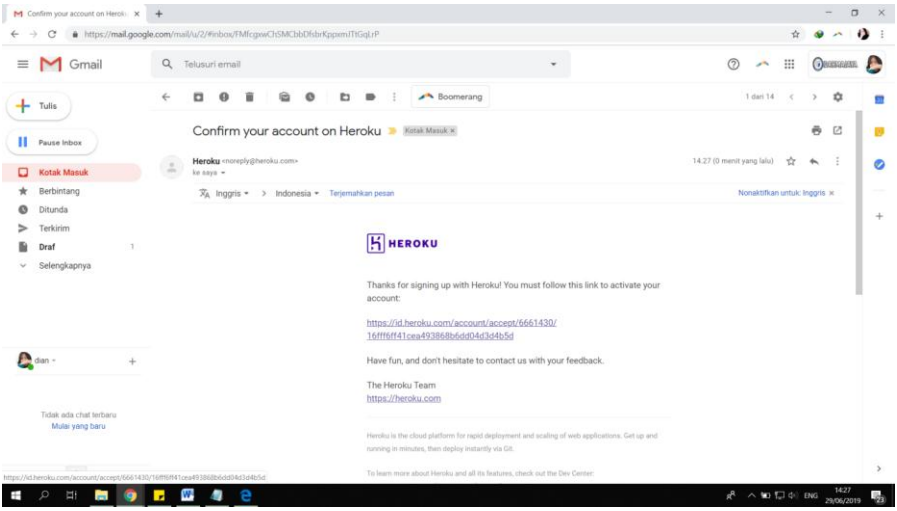
Di bagian selanjutnya, kita akan mempelajari tindakan yang diperlukan untuk menyiapkan klien Heroku di mesin lokal Anda.

Mendaftar

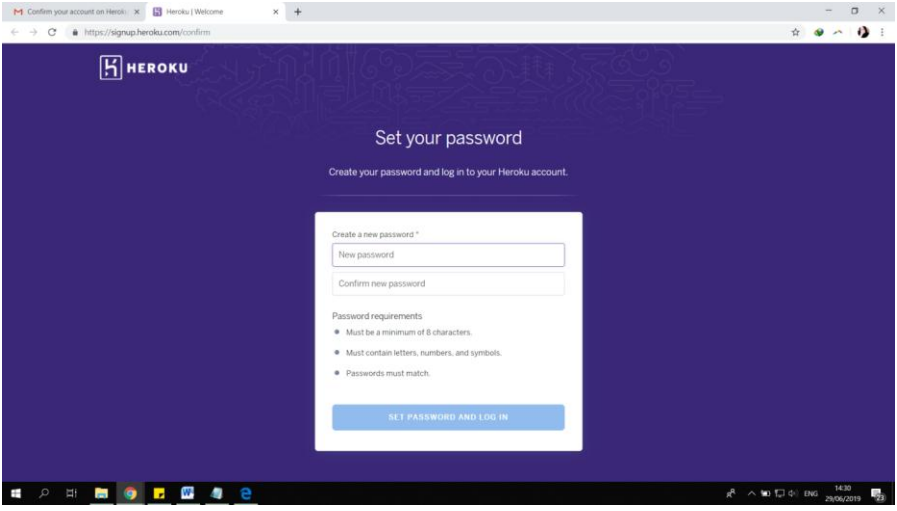
Untuk mulai menggunakan Heroku, Anda harus mendaftar untuk mendapatkan akun Heroku. Dibawah ini adalah tampilan pada website Heroku untuk mendaftarkan akun baru :



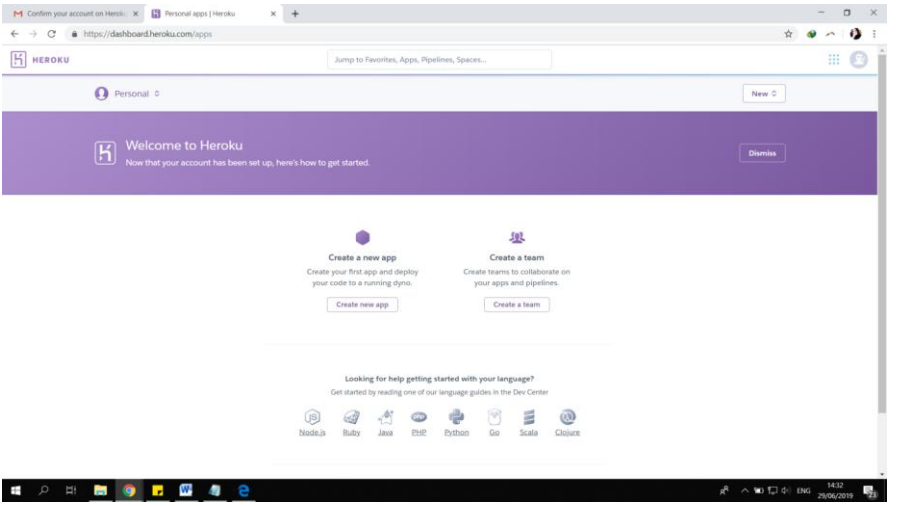
Setelah Anda memasukkan alamat email, Anda perlu memvalidasi akun Anda dengan masuk ke akun email Anda dan memverifikasi akun Anda untuk Heroku. Berikut tampilannya :



Setelah Anda memvalidasi akun Anda, Anda akan diarahkan ke halaman pengaturan kata sandi Heroku, di mana Anda harus memasukkan kata sandi yang valid dan mengonfirmasinya, seperti yang diperlihatkan dalam tampilan berikut :

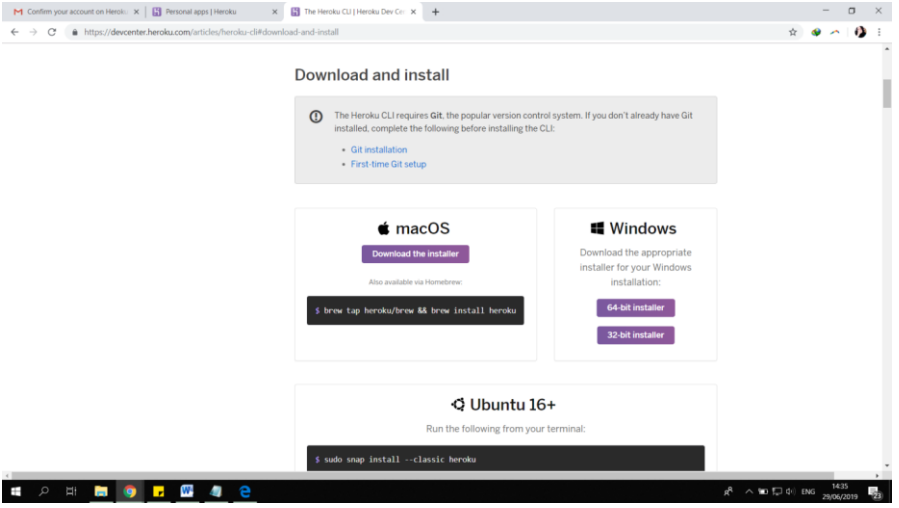


Setelah konfirmasi kata sandi berhasil, Anda akan dialihkan ke dasbor Heroku. Karena Anda adalah pengguna baru, dashboard belum akan berisi aplikasi yang digunakan. Tampilannya akan sebagai berikut ini :



Menginstal toolbelt Heroku

Toolbelt Heroku adalah perangkat lunak klien yang diperlukan untuk bekerja dengan platform Heroku. Ini dapat diunduh dari https://devcenter.heroku.com/articles/heroku-cli#download-and-install untuk platform Windows, Debian / Ubuntu dan Mac OS. Sebagai contoh, kami akan menggunakan versi Windows seperti yang ditunjukkan pada tangkapan layar berikut:



Instal toolbelt Heroku dengan mengklik dua kali pada executable yang diunduh dan mengikuti instruksi.
Toolbelt Heroku berisi komponen-komponen berikut:

1) Klien Heroku: Ini adalah alat baris perintah untuk membantu membuat dan mengelola aplikasi.
2) Foreman: Ini adalah utilitas untuk memberi Anda fleksibilitas menjalankan aplikasi secara lokal, terutama terkadang ketika Anda ingin memecahkan masalah.
3) Git: Ini adalah sistem kontrol revisi dan program utilitas yang relevan yang membantu Anda mendorong kode Anda ke Heroku atau mengunduhnya dari repositori jarak jauh. Anda juga dapat mengunduh Git UI khusus klien ke sistem operasi Anda jika Anda ingin antarmuka pengguna mengeluarkan perintah Git.

Logging dan membuat kunci SSH baru

Setelah menginstal toolbelt, Anda dapat mengeluarkan perintah Heroku dari prompt perintah klien Heroku. Perintah (atau kode) ini dikirim ke server Heroku dan dieksekusi pada platform Heroku. Namun, Heroku perlu memverifikasi bahwa perintah dan kode yang dikirim kepadanya berasal dari pengguna asli. Karenanya, Anda perlu membuat pasangan kunci publik / pribadi untuk mendorong kode ke Heroku.

Ikuti langkah-langkah ini untuk membuat kunci dan mengunggahnya ke akun Heroku Anda:

1) Buat kunci pada mesin lokal menggunakan alat ssh-keygen (tersedia di http://www.openssh.com/).
2) Tambahkan kunci ini ke akun Heroku terkait menggunakan kunci: add heroku command.

Uji Coba Heroku

Apa asyiknya melihat Ferrari dan tidak mengendarainya? Jadi, mari kita kenakan sabuk pengaman dan ambil Heroku. Kami berasumsi bahwa kami menjalankan klien Heroku pada mesin Windows 7. Kami berasumsi bahwa kami memiliki versi Windows dari toolbelt Heroku Kami juga telah membuat dan menambahkan kunci SSH ke akun Heroku kami.

Di bagian ini, kita akan membuat aplikasi barebones Rails 3 pada platform Heroku. Anda dapat menggunakan bahasa lain yang didukung, namun langkah-langkahnya tidak akan berubah kecuali untuk detail khusus bahasa (buildpack digunakan). Pertama dan terutama, kita perlu beberapa hal dan berjalan sebelum kita dapat membuat aplikasi pertama kita di Heroku. Persyaratan ini khusus untuk bahasa yang kami gunakan di sini.
Jadi, sebelum kita mulai:

1. Pastikan Anda memiliki prasyarat tertentu untuk contoh Rails 3. Anda membutuhkan:
   - Ruby 1.9.2 (toolbelt runtime Ruby dibundel dengan satu) ° permata Ruby (http://rubygems.org)
   - Bundler (http://rubygems.org/gems/bundler) ° Rails 3 (http : //rubyonrails.org/)
2. Periksa apakah klien Heroku diinstal dan di jalur buka command prompt dan ketik:
   $ heroku
   Anda akan mendapatkan bantuan penggunaan Heroku sebagai berikut:

```
Usage: heroku COMMAND [--app APP] [command-specific-options]

Primary help topics, type "heroku help TOPIC" for more details:

  addons     #  manage addon resources
  apps       #  manage apps (create, destroy)
  auth       #  authentication (login, logout)
  config     #  manage app config vars
  domains    #  manage custom domains
  logs       #  display logs for an app
  ps         #  manage processes (dynos, workers)
  releases   #  manage app releases
  run        #  run one-off commands (console, rake)
  sharing    #  manage collaborators on an app

Additional topics:

  account       #  manage heroku account options
  certs         #  manage ssl endpoints for an app
  db            #  manage the database for an app
  drains        #  display syslog drains for an app
  git           #  manage git for apps
  help          #  list commands and display help
  keys          #  manage authentication keys
  labs          #  manage optional features
  maintenance   #  manage maintenance mode for an app
  pg            #  manage heroku-postgresql databases
  pgbackups     #  manage backups of heroku postgresql databases
  plugins       #  manage plugins to the heroku gem
  ssl           #  manage ssl certificates for an app
  stack         #  manage the stack for an app
  status        #  check status of heroku platform
  update        #  update the heroku client
  version       #  display version
```

3. Masuk menggunakan alamat email dan kata sandi yang Anda gunakan saat membuat Anda akun Heroku:
   $ heroku login
   Masukkan kredensial Heroku Anda.
   Email: xxxx@yyyy.com
   Kata Sandi:
   Tidak dapat menemukan kunci publik yang ada.
   Apakah Anda ingin membuatnya? [Yn]
   Menghasilkan kunci publik SSH baru.
   Mengunggah kunci publik ssh /users/xxxx/.ssh/id_rsa.pub
   Tekan enter saat diminta untuk mengunggahAnda ssh kunciatau buat yang baru.
4. Tulis aplikasi Rails 3 sederhana:
   $ rails sampleapp_rails3 baru
   $ cd sampleapp_rails3
5. Edit Gemfile Anda (Gemfile adalah file yang berisi daftar nama pustaka / file gem aplikasi Anda perlu berfungsi dengan benar) dan ubah gem 'sqlite3' untuk permata 'pg'. Perubahan ini memungkinkan aplikasi untuk menggunakan database Heroku PostgreSQL yang baru diperkenalkan, bukan SQLite.
6. Instal ulang dependensi Anda (untuk menghasilkanbaru Gemfile.lock):
   $ bundle install
7. Buat aplikasi Anda di Git seperti yang ditunjukkan pada gambar berikut :

```
git init
git add
git commit -m “init”
•   Deploy the application on Heroku
•   Create the app on Heroku using:
heroku create
Creating pink-poppies-786... done, stack is cedar
http://pink-poppies-786.herokuapp.com/ | git@heroku.com:pink-poppies-786.git
Git remote heroku added
•   Deploy you code:
git push heroku master
counting objects: 67, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (52/52), done.
Writing objects: 100% (67/67), 86.33 KiB, done.
Total 67 (delta 5), reused 0 (delta 0)

-----> Heroku receiving push
-----> Rails app detected
-----> Installing dependencies using Bundler version 1.1
       Checking for unresolved dependencies.
       Unresolved dependencies detected.
       Running: bundle install --without development: test --path vendor / bundle – deployment
       Fetching source index for http: //rubygems.org/
       Installing rake (0.8.7)
        ...
       Installing rails (3.0.5)
       You bundle is complete! It was installed into ./vendor/bundle
-----> Rails plugin injection
       Injecting rails_log_stdout
       Injecting rails3_serve_static_assets
-----> Discovering process types
       Proc?le declares types -> (none)
       Default types for Rails -> console, rake, web, worker
-----> Compiled slug size is 8.3MB
-----> Launching...done, v5
       http.//pink-poppies-786.herokuapp.com deployed to heroku

To git@heroku.com:pink-poppies-786.git
* [new branch]          master -> master
```

8. Untuk membuka aplikasi Heroku, ketikkan terbuka perintahpada Heroku CLI:
   $ heroku open
   Membuka pink-poppies-786 ... selesai
9. Periksa status menjalankan proses Heroku, ketik perintahps (status proses) sebagai berikut:
   $ heroku ps
   === web: `bundle exec rails server –p $ PORT`web.1: up for 5s
10. Untuk mendukung lebih banyak bersamaan pengguna, Anda perlu meningkatkan aplikasi Anda. Tambah hitungan web dyno dari baris perintah Heroku. Gunakan perintah ps: scale untuk meningkatkan proses web menjadi 2 sebagai berikut :
    $ heroku ps: scale web = 2

11. Jika Anda menemukan masalah saat menjalankan aplikasi Heroku, Anda dapat menggunakan log perintahuntuk mendapatkan detail pesan terbaru dan menggunakan informasi tersebut untuk memecahkan masalah terkait seperti yang ditunjukkan pada gambar berikut:

```
$ heroku logs
2012-12-10T11:10:34.08:00 heroku[web.1]: State changed from created to starting
2012-12-10T11:10:37.08:00 heroku[web.1]: Running process with command: 'bundle exec rails server -
p 42383'
2012-12-10T11:10:40.08:00 app[web.1]: [2012-12-10 11:23:40] INFD WEBrick 1. 3. 1
2012-12-10T11:10:40.08:00 app[web.1]: [2012-12-10 11:23:40] INFD ruby 1. 9. 2. (2010-12-25) [x86_64-
linux]
```

Kami membuat akun Heroku, menambahkan kunci SSH kami ke Heroku, dan menulis aplikasi barebones Rails3 dan menggunakannya di Heroku. Akhirnya, untuk memecahkan masalah, kami menggunakan log perintah untuk melihat apa yang terjadi di balik layar kalau-kalau ada sesuatu yang perlu diperhatikan.

Ringkasan

Dalam bab ini, kami membangun di atas pemahaman kami tentang komputasi awan, model layanan cloud yang berbeda dan mengapa pengembangan berbasis cloud sangat masuk akal di lingkungan bisnis saat ini. Kami diperkenalkan ke platform Heroku ajaib sebagai layanan (PaaS), yang menyediakan platform build, deploy, dan operating terintegrasi untuk pengembang web untuk digunakan dan mengelola aplikasi mereka dengan mudah. Kami meninjau arsitektur platform Heroku dan memahami rangkaian fitur yang kaya dari platform tersebut. Akhirnya, kami menguji mengendarai platform Heroku, membahastoolset yang diperlukan untuk terhubung ke Heroku dan membangun / menggunakan aplikasi barebones di Heroku dalam proses.

Pada bab selanjutnya, kita akan mempelajari sedikit lebih dalam tentang arsitektur platform Heroku dan memahami bagaimana Heroku bekerja di bawah selimut. Secara khusus, kami fokus pada detail clonning Heroku, arsitektur prosesnya, kerangka kerja eksekusi, dan arsitektur logging. Dan masih ada lagi

BAB 2
ISI HEROKU

Sekarang setelah kami menguji Heroku dan mengetahui sedikit tentang set fitur-fiturnya, sekarang saatnya untuk memahami beberapa cara kerja dalam platform Heroku.

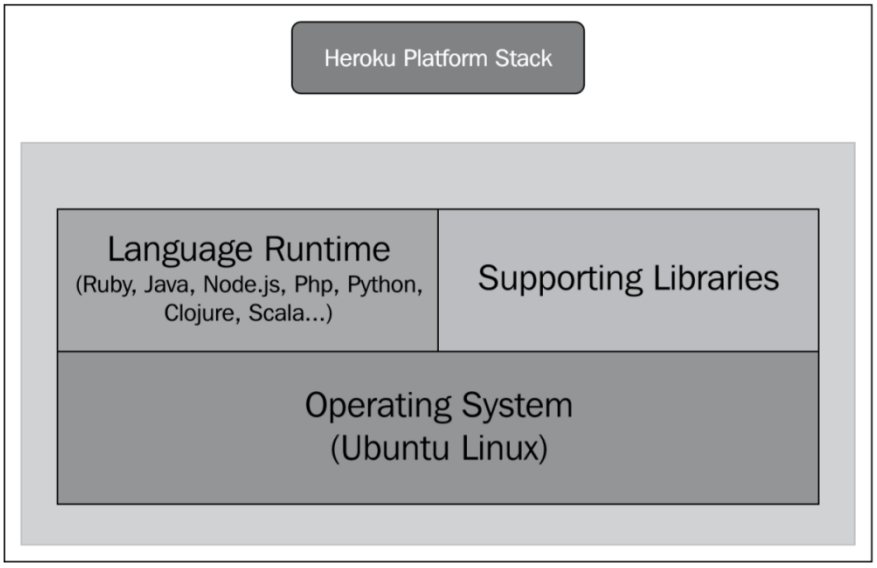
Platform pengembangan cloud Heroku memberi Anda dasar untuk mengembangkan, menyebarkan, dan memecahkan masalah aplikasi berbasis cloud Anda. Ini terdiri dari komponen-komponen berikut:

- Tumpukan platform: Ini menyediakan komponen inti seperti sistem operasi, runtime bahasa, dan pustaka pendukung.
- Mekanisme perutean permintaan: Ini memberikan dukungan untuk menerima permintaan klien dan merutekannya dengan andal ke proses yang sedang berjalan.
- Lingkungan eksekusi: Ini memberikan dukungan runtime yang diperlukan untuk menjalankan aplikasi Anda.
- Kerangka penebangan Logplex: Ini adalah sistem pencatatan peristiwa untuk menyusun peristiwa aplikasi yang berasal dari berbagai proses untuk diproses lebih lanjut.
- Ekosistem add-on: Ini menyediakan kemampuan untuk dengan mudah menyambungkan perpustakaan pihak ketiga untuk memberikan layanan bernilai tambah, seperti pemantauan kinerja, caching, atau penyimpanan data untuk aplikasi Anda.
- API Platform Heroku: Ini adalah antarmuka layanan web untuk melakukan operasi Heroku secara terprogram dari kode Anda.

Dalam bab ini, kami meninjau setiap aspek dasar dari platform Heroku dengan sedikit lebih detail. Kita juga akan mendapatkan pemahaman yang lebih dalam tentang arsitektur proses Heroku – komponen inti dari lingkungan runtime Heroku.

Lapisan Platform Heroku

Heroku adalah sebuah platform, yaitu area yang memberi Anda fondasi untuk menjalankan aplikasi Anda tanpa khawatir tentang seluk-beluk cara kerjanya. Anda membangun aplikasi, mengkoneksikan ke Heroku, dan voila, kompiler slug deployment stackAnda menjadi aplikasi yang siap dijalankan. Terlebih lagi, itu juga membuat URL dapat digunakan aplikasi. Sementara semua proses ini bekerja dalam waktu singkat, ada dasar yang kuat, sangat tersedia, dapat diukur, dan bukti kegagalan di bawahnya yang memungkinkan semua ini terjadi. Elemen inti dari platform Heroku adalah tumpukan Heroku – kombinasi dari sistem operasi, runtime bahasa, dan pustaka terkait yang menyediakan lingkungan eksekusi lengkap bagi pengguna untuk menjalankan aplikasi dari berbagai beban kerja dan skala dengan baik.



Pada deployment stack, sistem operasi menentukan teknologi dasar yang digunakan oleh platform Heroku , dan telah bervariasi di berbagai rilis tumpukan Heroku. Setiap rilis ini telah mendukung serangkaian runtime bahasa pemrograman yang berbeda. Citarasa Heroku Ubuntu 10.04 yang tersedia saat ini berdasarkan sistem operasi disebut tumpukan Celadon Cedar. Dengan diadopsinya tumpukan Celadon Cedar, Heroku telah matang menjadi platform polyglot padat yang secara asli mendukung runtime bahasa paling populer untuk pengembangan aplikasi. Dari mendukung Ruby dalam rilis awalnya, Heroku sekarang secara asli mendukung enam bahasa pemrograman, dan daftarnya terus bertambah.

Selama bertahun-tahun, Heroku telah mendukung tiga deployment stack berikut :

Stack	Underlying platform	Supported runtimes
Argent Aspen	Debian Etch 4.0	MRI* 1.8.6
Badious Bamboo	Debian Lenny 5.0	REE** 1.8.7, MRI 1.9.2
Celadon Cedar	Ubuntu 10.04	REE 1.8.7, MRI 1.9.2, Node.js, Clojure, Java, Python, Scala

* MRI: Matz Ruby Interpreter

** REE: Ruby Enterprise Edition Detail lebih lanjut dari bahasa Heroku support dapat ditemukan di https://devcenter.heroku.com/categories/language-support.

The Celadon Cedar Stack

Celadon Cedar Stack adalah stack penyebaran default untuk platform Heroku ke depannya. Banyak aplikasi yang digunakan pada Heroku mungkin masih menjalankan tumpukan Badious Bamboo karena Heroku mendukungnya untuk kompatibilitas versi lama. Namun, disarankan agar pengguna bermigrasi ke Cedar stack untuk memanfaatkan fitur-fitur terbaru. Juga disarankan agar pengguna terlebih dahulu menguji aplikasi dengan runtime bahasa yang didukung pada area pengujian sebelum memindahkan seluruh aplikasi ke bagian produksi di Cedar.

Cedar stack membawa banyak peningkatan drastis pada deployment stack Heroku termasuk stack berikut :

- Dukungan multi-bahasa: Cedar deployment stackmemiliki tujuan umum, artinya ia tidak memberikan dukungan orisinil untuk salah satu bahasa pemrograman. Ini memberikan kemampuan untuk menambahkan dukungan untuk bahasa yang secara virtual sesuai permintaan melalui mekanisme adaptor waktubangun yang disebut buildpack. Buildpack dapat mengkompilasi aplikasi yang ditulis dalam bahasa pemrograman tertentu menjadi executable biner yang dapat dieksekusi pada runtime yang disediakan oleh stack deployment Cedar. Secara resmi, Cedar menyediakan dukungan untuk REE 1.8.7, MRI 1.9.2, Node.js, Clojure, Java, Python, dan Scala; meskipun Anda bisa menambahkan buildpack baru untuk bahasa pilihan dan menjalankan aplikasi yang ditulis di sebagian besar bahasa pemrograman di platform Heroku.
  Kemudahan menggunakan buildpacks kustom ini telah menyebabkan pengembang memberikan dukungan untuk banyak bahasa populer seperti PERL, Go, dan Common LISP. Menjalankan aplikasi dengan buildpack kustom semudah menentukan variabel lingkungan BUILDPACK_URL dan meneruskan URL buildpack pada waktu pembuatan aplikasi seperti berikut:
  $ heroku create --buildpack <BUILDPACK_URL>
- Model proses baru: Cedar stack memanfaatkan Model proses UNIX untuk menyediakan area eksekusi proses yang mulus yang dapat meningkatkan permintaan dan menyediakan layanan aplikasi yang toleran terhadap kesalahan. Kerangka model proses memberikan perincian terperinci untuk mengelola aplikasi dalam lingkungan eksekusi proses yang dapat menjangkau beberapa mesin. Model proses memperkenalkan konsep Procfile (formulir pendek untuk file proses), yang merupakan file konfigurasi untuk pembentukan proses Anda. Berbagai jenis proses dan cara menjalankannya dapat ditentukan di Procfile. Keberadaan Procfile memberikan fleksibilitas besar dalam memilih ukuran yang tepat dari proses aplikasi Anda. Model proses juga memungkinkan Anda untuk

menjalankan proses satu kali pada baris perintah, karenanya memberikan fleksibilitas yang lebih besar saat menjalankan utilitas atau aplikasi hanya sekali ini saja. Anda dapat menskalakan jenis proses tertentu untuk menjalankan lebih banyak proses dengan cepat, dan lingkungan eksekusi yang mendasari mengurus memulai instance baru secara otomatis. Fleksibilitas yang disediakan oleh Procfile juga membantu dalam mengganti yang dapat dieksekusi dengan yang lain, jika diperlukan peningkatan, misalnya, jika Anda ingin mengganti satu server web dengan yang lain karena persyaratan aplikasi.

- Pemecahan masalah yang lebih baik: Salah satu fitur platform Heroku yang sangat cerdas adalah referensi perintah intuitif untuk menjalankan, memantau, dan mengelola tugas aplikasi. Tidak seperti banyak penawaran PaaS lainnya, Heroku menyediakan referensi perintah seperti UNIX yang lebih mudah dipelajari dan mulai digunakan. Perintah Heroku CLI yang paling umum pada tumpukan Cedar Celadon diberi nama setelah perintah shell UNIX diawali dengan Heroku untuk memberikan perbedaan. Misalnya, perintah sederhana untuk memeriksa status proses Heroku Anda ditampilkan dalam tangkapan layar berikut:

```
$ heroku ps
=== web: `bundle exec thin start -p $PORT -e production`
web.1: up for 3h
web.2: up for 2m
```

Outputnya menunjukkan bahwa saat ini dua proses web sedang berjalanselama 3 jam dan 2 menit masing-masing. Baris dengan === menunjukkan perintah yang dieksekusi di lingkungan eksekusi Heroku. Selain dekat dengan sintaks UNIX kapan pun bisa, Heroku juga menyediakan sintaks langsung untuk mengelola proses. Misalnya, untuk mengurangi jumlah jenis proses web dan menambah jenis proses pekerja, semua yang diperlukan adalah satu liner berikut: Gunakan + (kenaikan) dan - indikator(penurunan) untuk menambah dan mengurangi jumlah masing-masing proses Heroku , seperti yang ditunjukkan pada tangkapan layar berikut:

```
$ heroku ps:scale web-1 worker+2
Scaling web processes... done, now running 1
Scaling worker processes... done, now running 3
```

Memeriksa log untuk aplikasi pemecahan masalah atau memverifikasi aliran proses sesederhana yang ditunjukkan tangkapan layar berikut:

```
$ heroku logs
←[36m2013-01-27T10:48:08+00:00 heroku[api]:←[0m Starting process with command `date` by
ahanjura@gmail.com
←[33m2013-01-27T10:48:12+00:00 heroku[run.2988]:←[0m Awaiting client
←[33m2013-01-27T10:48:12+00:00 heroku[run.2988]:←[0m Starting process with command `date`
←[33m2013-01-27T10:48:14+00:00 heroku[run.2988]:←[0m Process exited with status 0
←[33m2013-01-27T10:48:14+00:00 heroku[run.2988]:←[0m State changed from starting to complete
```

Beberapa pernyataan log (log tail) terakhir dapat dilihat oleh perintah heroku logs –t . Terlebih lagi, Anda dapat memfilter pesan log untuk jenis proses tertentu. Untuk memfilter pesan log dari pekerja proses.1 dan daftar pesan log yang sesuai, gunakan perintah heroku log --ps pekerja.1 –t . Di sini, pengguna mencoba memfilter pesan log untuk pekerja.1 prosessaja. Fleksibilitas ini sangat kuat dalam mengatasi masalah pembentukan proses besar dengan contoh proses tertentu.

• Manajemen rilis: Pada penerapan kode baru, apakah itu perubahan konfigurasi atau modifikasi (menambah / menghapus) sumber daya, Heroku membuat rilis baru dan memulai ulang aplikasi secara otomatis. Heroku memberikan fleksibilitas untuk membuat daftar riwayat rilis dan menggunakan rollback untuk kembali ke rilis sebelumnya, jika penyebaran atau konfigurasi salah. Untuk membuat daftar rilis Anda, ketikkan apa yang ditunjukkan pada tangkapan layar berikut:

```
$ heroku releases
=== gentle-mesa-5445 Releases
v2  Enable Logplex   ahanjura@gmail.com  2012/12/03 03:39:37
v1  Initial release  ahanjura@gmail.com  2012/12/03 03:39:36
```

Untuk menunjukkan, cara memodifikasi aplikasi Anda, saya telah menambahkan variabel konfigurasi baru, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku config:add TEST_ENV_VAR_EDHA=45
Setting config vars and restarting gentle-mesa-5445... done, v3
TEST_ENV_VAR_EDHA: 45
```

Verifikasi bahwa rilis baru ditambahkan oleh Heroku, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku releases
=== gentle-mesa-5445 Releases
v3  Add TEST_ENV_VAR_EDHA config  ahanjura@gmail.com  2013/01/27 20:16:21 (~ 17s ago)
v2  Enable Logplex                ahanjura@gmail.com  2012/12/03 03:39:37
v1  Initial release               ahanjura@gmail.com  2012/12/03 03:39:36
$ heroku releases:info v3
=== Release v3
By:     ahanjura@gmail.com
Change: Add TEST_ENV_VAR_EDHA config
When:   2013/01/27 20:16:21 (~ 57s ago)

=== v3 Config Vars
TEST_ENV_VAR_EDHA: 45
```

Akhirnya, saya kembalikan perubahan ke Versi 2 seperti yang ditunjukkan pada tangkapan layar berikut :

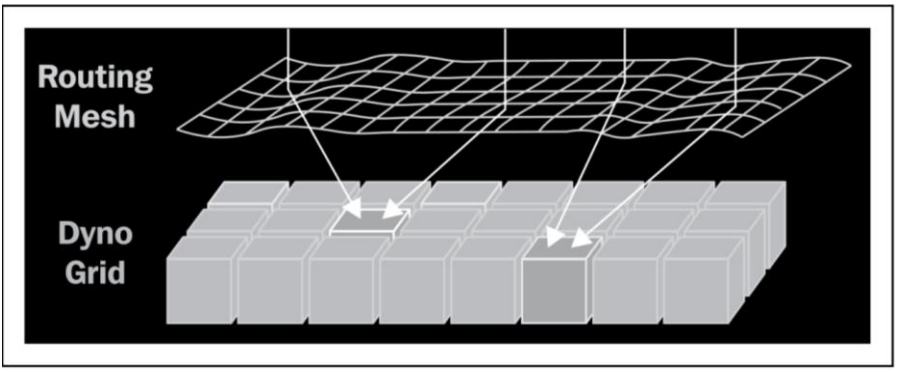
```
$ heroku rollback
Rolling back gentle-mesa-5445... done, v2
!    Warning: rollback affects code and config vars; it
  doesn't add or remove addons. To undo, run: heroku
  rollback v3
```

Meminta perutean di Heroku

Di tumpukan penyebaran saat ini, aplikasi bernama myapp akan memiliki nama host default dari myapp.herokuapp.com. herokuapp.com Rutedomainsetiap permintaan untuk Heroku melalui routing mesh dan menyediakan jalur routing langsung ke proses web yang sesuai. Ini memungkinkan penggunaan HTTP tingkat lanjut seperti respons terpotong, pemungutan suara

panjang, dan menggunakan server web asinkron untuk menangani banyak respons dari satu proses web tunggal.

Routing mesh bertanggung jawab untuk mengarahkan permintaan ke masing-masing dyno di manifold dyno. Routing mesh menggunakan algoritma round robin untuk melakukan distribusi permintaan ke dino seperti yang ditunjukkan pada diagram berikut:



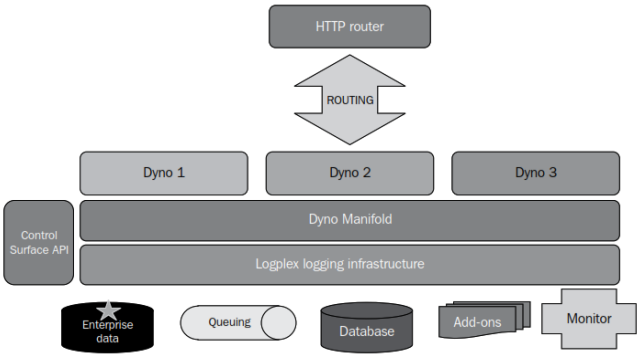
Permintaan HTTP mengikuti batas waktu (30 detik) untuk proses web untuk mengembalikan data responsnya ke klien . Jika proses web gagal mengembalikan respons dalam waktu ini, log proses yang sesuai mencatat kesalahan. Setelah komunikasi awal, jendela waktu yang lebih besar (55 detik) ditetapkan untuk klien atau server untuk mengirim data. Jika tidak satu pun dari mereka yang mengirimnya, koneksi diakhiri dan kesalahan dicatat lagi. Skema timeout bergulir ini memfasilitasi pembebasan koneksi jaringan ketika lebih mungkin bahwa komunikasi tidak akan terjadi dalam periode waktu yang cukup praktis. Koneksi yang kurang terbuka berarti manajemen sumber daya dan kinerja yang lebih baik.

Deployment stack Heroku juga mendukung menjalankan aplikasi multithreaded dan / atau asinkron, yang dapat menerima banyak koneksi jaringan untuk memproses acara klien. Misalnya, semua aplikasi Node.js biasanya menangani beberapa koneksi jaringan sesuai proses saat menangani permintaan klien.

Cedar tidak lagi memiliki cache proxy terbalik seperti Varnish, karena lebih suka menawarkan fleksibilitas bagi pengembang untuk memilih solusi CDN pilihan mereka. Milik Herokuarsitektur add-on mendukung pengaktifan opsi semacam itu.

Lingkungan eksekusi - dyno dan bermacam-macam dyno

Dyno seperti UNIX virtual yang dapat menjalankan semua jenis proses pada manifold dyno. Manifold dyno adalah lingkungan proses eksekusi yang memfasilitasi eksekusi berbagai dino, yang mungkin melayani permintaan klien yang berbeda. Diagram berikut menunjukkan berbagai komponen lingkungan eksekusi Heroku:



Infrastruktur berjenis dyno menyediakan manajemen proses yang diperlukan, isolasi, penskalaan, perutean, distribusi, dan redundansi yang diperlukan untuk menjalankan web tingkat produksi dan proses pekerja dengan cara yang benar.

Manifold adalah lingkungan yang toleran terhadap kesalahan, terdistribusi agar proses Anda berjalan dalam aliran penuh. Jika Anda merilis versi aplikasi baru, manifold mengatur restart secara otomatis, sehingga menghilangkan banyak kesulitan maintenance. Dyno digunakan ulang setiap hari atau kendala sumber daya migrasi atau untuk mengubah ukuran dyno. Manifold bertanggung jawab untuk memulai ulang secara otomatis setiap dyno yang gagal. Jika dyno gagal lagi dan lagi, manifold tidak segera memulai kembali tetapi menunggu delta dan kemudian mencoba untuk memulai kembali. Anda dapat mencoba memulai proses secara manual menggunakan perintah restart heroku.

Anda dapat mengatur parameter global dan variabel konfigurasi tertentu di lingkungan. File profil saat dipanggil sebelum manifold memulai dyno. Setiap dyno mandiri dalam arti bahwa ia menjalankan contoh tertentu dari proses web atau pekerja dan memiliki akses ke sumber daya yang cukup untuk menangani permintaan klien.

Sistem Logplex tidak menyediakan penyimpanan untuk data log yang seluruhnya dihasilkan tetapi hanya menyimpan data terbaru yang cukup baik untuk mengekstrak informasi yang relevan dari aplikasi dijalankan. Biasanya, data ini berukuran sekitar 1.500 pesan log gabungan. Jika Anda memiliki bisnis perlu menyimpan lebih dari itu batas yang telah ditentukan, Anda harus memilih layanan penyimpanan alternatif, misalnya, saluran **Syslog** untuk mengarsipkan semua pesan.

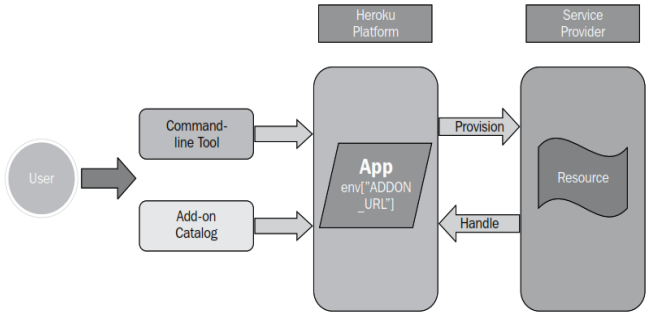
Logplex dirancang dengan kemampuan untuk bekerja dengan alat dan layanan eksternal seperti Loggly. Jika salah satu dari alat atau layanan eksternal ini tidak dapat menggunakan pesan yang dihasilkan oleh Logplex, mereka dapat kehilangan sebagian dari pesan-pesan ini. Namun, Logplex tidak memasukkan entri peringatan kapan pun ia harus meninggalkan pesan karena konsumsi downtime layanan.

Arsitektur tambahan Heroku

Add-on adalah salah satu cara untuk memperluas kemampuan aplikasi Heroku. Pengguna dapat menambahkan layanan baru seperti caching atau penyimpanan data ke aplikasi dengan menyediakan add-on dan mengonsumsi layanan yang ditawarkan. Pengguna Heroku memiliki dua cara untuk menyediakan pengaya :

- Pasar add-on
- Klien sabuk alat Heroku

Seluruh proses penyediaan dan konsumsi add-on ditunjukkan didiagram berikut:



Langkah-langkah berikut menjelaskan bagaimana add-on baru disediakan dan dikonsumsi oleh
Aplikasi Heroku :

1. Pengguna Heroku meminta add-on menggunakan salah satu dari dua cara yang ditunjukkan dalam paragraf sebelumnya.
2. Heroku mengirimkan permintaan penyediaan untuk akun saat ini ke penyedia layanan untuk layanan yang diberikan.
3. Penyedia layanan memvalidasi permintaan dan membuat sumber daya pribadi untuk pemanggil dan mengembalikan URL yang menentukan lokasi yang tepat dan kredensial yang diperlukan untuk mengakses layanan. Misalnya, penyedia layanan Amazon RDS mungkin kembali: MYSQL_URL = mysql: // user: pass@mydbhost.com/database.
4. Pada titik ini, Heroku menambahkan URL yang dikembalikan ke lingkungan aplikasi yang meminta, membangun kembali aplikasi, dan me-restart dino terkait untuk aplikasi.

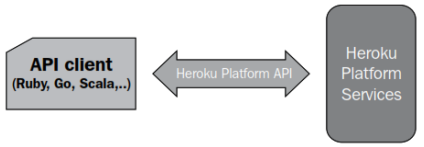
5. Aplikasi siap untuk mengkonsumsi layanan dari add-on baru.
6. Aplikasi mengekstrak URL sumber daya dari lingkungan dan mengeluarkan permintaan baca / tulis ke layanan, melewati parameter yang diperlukan.
7. Sumber daya yang diminta dalam penyedia layanan memvalidasi keaslian permintaan dan memberikan hak istimewa untuk melaksanakan operasi jika permintaan tersebut valid dan pengguna tersebut asli.

8. Sumber daya membuat respons dan mengirimkannya kembali ke aplikasi, yang pada gilirannya menggunakan hasilnya untuk membuat halaman atau menyimpan respons dalam memori aplikasi untuk digunakan nanti dan melanjutkan.

Perhatikan bahwa permintaan layanan dari aplikasi dapat berasal dari pembuatan pengguna permintaan di browser web atau dyno pekerja yang meminta sumber daya sementara melakukan pekerjaan latar belakang yang berjalan lama.

Secara terprogram mengonsumsi Heroku jasa

Dengan Heroku Platform API, Heroku memberi Anda kemampuan untuk memanggil layanan platform Heroku untuk melakukan hampir semua hal yang perlu Anda lakukan di Heroku seperti membuat aplikasi, memeriksa riwayat rilis aplikasi Anda, atau memasukkan add-on baru ke aplikasi Anda . Semua ini dan lebih banyak lagi hanya dengan menggunakan HTTP sederhana. Diagram berikut menggambarkan interaksi antara klien Heroku API sederhana dan layanan platform Heroku. Klien API dapat menggunakan API Heroku untuk berinteraksi dengan layanan platform inti Heroku menggunakan antarmuka yang terdefinisi dengan baik.



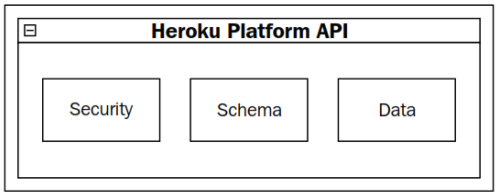
API Platform Heroku memungkinkan Anda melakukan hampir semua yang dapat Anda lakukan dengan baris perintah atau dasbor Heroku. API ini adalah alat yang ampuh bagi pengembang aplikasi untuk memanfaatkan kemampuan fitur Heroku dan mendapatkan kontrol penuh atas aplikasi mereka, mulai dari membuat hingga memantau mereka.

API Platform Heroku

Setidaknya ada tiga komponen penting dari API Platform Heroku seperti yang ditunjukkan pada diagram berikut. Komponen-komponen ini menentukan perilaku API ke dunia eksternal.

Komponennya adalah:

- Keamanan
- Skema
- Data



Keamanan

Aplikasi apa pun yang ingin mengkonsumsi layanan yang disediakan oleh platform Heroku melalui Platform API harus mengotentikasi pengguna yang mencoba mengakses layanan. Meskipun otentikasi HTTP dasar akan berfungsi, jika Anda menggunakan skrip Anda sendiri dan tidak berharap untuk mengeksposnya ke khalayak yang lebih luas, OAuth adalah cara yang disarankan untuk memberikan dukungan otentikasi bagi pihak ketiga untuk mengakses layanan Heroku. Anda dapat menggunakan token OAuth untuk memberikan akses ke akun Anda kepada setiap pengguna termasuk pihak ketiga. Dengan mendapatkan akses melalui OAuth, pihak ketiga kemudian dapat menggunakan aplikasi Anda untuk menyediakan banyak layanan bernilai tambah seperti pemantauan, skalabilitas, atau manajemen aplikasi.

Skema

Untuk menggunakan Platform API, klien perlu mengetahui beberapa hal. Misalnya, sumber daya apa yang tersedia, apa lokasi sumber daya yang sesuai, bagaimana sumber daya ini diwakili, dan operasi apa yang diizinkan pada sumber daya itu? Hanya dengan demikian klien dapat membuat permintaan untuk mengakses detail sumber daya yang tersedia di platform Heroku melalui API. Sebagian

besar informasi ini disediakan oleh skema API. Skema ini disimpan dalam format JSON dapat dibaca oleh mesin dan menjelaskan berbagai aspek sumber daya. Skema API, seperti sumber daya API lainnya, juga dapat diakses menggunakan curl utility. Heroku juga mendukung ekstensi validasi dan hiperteks untuk skema JSON dasar untuk memungkinkan manipulasi data JSON yang lebih kuat.

Data

Kapan pun klien menerima respons dari API Platform Heroku, header respons berisi tag Entity (ETag). Tag ini digunakan untuk mengidentifikasi versi tepat dari sumber daya yang diakses, kalau-kalau sumber daya diperlukan lagi nanti. ETag ini dapat disematkan di dalam bidang tajuk If-Not-Match dalam permintaan di masa mendatang untuk memverifikasi bahwa sumber daya memang telah berubah. Jika sumber daya tidak berubah, status pengembalian tidak diubah dikirim kembali ke klien. Jika sumber daya telah berubah sejak permintaan sebelumnya, panggilan berkinerja normal dan mengembalikan karakteristik baru sumber daya tersebut.

Anda dapat mengoptimalkan akses sumber daya dengan caching respons sumber daya di sisi klien dan mengeluarkan permintaan API untuk sumber daya hanya ketika sumber daya memang diubah di sisi platform.

Mengakses API

Klien yang menggunakan platform API memanggil metode standar yang ditentukan untuk HTTP dan meneruskan parameter di badan permintaan ke API yang melayani. API bertindak atas permintaan, melakukan operasi spesifik, dan mengembalikan hasil operasi kepada klien melalui respons yang merupakan representasi JSON dari detail sumber daya yang diminta.

Klien API

Sejak Heroku Platform API baru-baru ini dirilis dalam Beta, klien API yang ditulis dalam berbagai bahasa yang didukung Heroku perlahan-lahan muncul di kancah pengembangan. Heroku mendukung klien API yang disebut Heroics for Ruby dan ada banyak lagi yang muncul. Ada klien API yang dikenal untuk Node.js, Scala,

dan Go yang dapat digunakan berdasarkan persyaratan aplikasi Anda.

Memanggil API

Klien API mengatasi permintaan untuk api.heroku.com menggunakan HTTPS dan tentukan Accept: application / vnd.heroku + json; versi = 3 Terima tajuk HTTPS dalam permintaan. Klien juga dapat menentukan header Agen-Pengguna untuk membantu mengatasi masalah dan melacak permintaan jika diperlukan.

Untuk sebagian besar, Anda dapat menggunakan curl (http://curl.haxx.se/) sebagai klien API dalam skrip atau utilitas untuk memanggil Heroku Platform API. Konvensi panggilan untuk mengakses Platform API sederhana dan hasil yang dikembalikan dalam format JSON, jadi parsing dan menggunakannya sangat mudah.

Untuk bahasa pemrograman lain, yang Anda butuhkan adalah perpustakaan HTTP dan metode untuk memanggil API dari kode Anda dan Anda sudah siap. Panggil API, terima data yang dikembalikan, parsing, dan gunakan dalam aplikasi Anda. Di bagian ini, kami menggunakan curl untuk menunjukkan beberapa sampel untuk memberi Anda pemahaman dasar tentang cara menjalankan API.

Tanggapan

Berhasil atau tidaknya panggilan Platform API ditunjukkan oleh kode status yang dikembalikan oleh API setelah selesai. Panggilan API dapat menghasilkan berbagai tindakan, misalnya, pembuatan sumber daya seperti dalam kasus aplikasi baru yang dibuat atau meninjau riwayat sumber daya, misalnya, melihat riwayat revisi aplikasi.

Respons API terdiri dari bagian header HTTP yang lebih pendek dan relevan dan detail respons di badan JSON jika diperlukan. Header respons mencakup hitungan saat ini dari permintaan API yang tersisa untuk akun.

Batasi panggilan API

API platform Heroku memungkinkan pengembang untuk mengontrol berbagai aspek pengembangan aplikasi Heroku dari dalam aplikasi. Pengembang dapat mengelola konfigurasi aplikasi, runtime (dyno), penyebaran, dan manajemen akun pengguna, di samping beragam fungsi lainnya. Namun, platform membatasi jumlah panggilan API yang dapat dilakukan pengguna atau pengembang per jam. Ini pada dasarnya untuk melindungi dari penggunaan berlebihan yang tidak perlu, memastikan penanganan yang adil dari klien yang tak terhitung, dan menegakkan langkah-langkah keamanan. Dengan setiap panggilan API, jumlah token ini dikurangi.

Arsitektur proses Heroku

Aplikasi Heroku adalah kumpulan proses yang berjalan di manifold dyno Heroku. Apakah Anda menjalankan proses lokal atau yang jauh pada sekelompok mesin terdistribusi, aplikasi Heroku pada dasarnya adalah serangkaian proses, masing-masing memakan sumber daya seperti proses UNIX normal.

Untuk menambah fleksibilitas dan memiliki kontrol yang lebih baik tentang bagaimana Anda mendefinisikan konfigurasi proses Anda, Heroku mendefinisikan konsep tipe proses. Setiap proses yang Anda jalankan sebagai dyno dapat diklasifikasikan sebagai proses web atau jenis proses pekerja tergantung pada apakah ia menangani permintaan HTTP atau melakukan beberapa pemrosesan latar belakang. Anda juga dapat menentukan jenis proses khusus untuk menambah fleksibilitas pada definisi aplikasi Anda.

Procfile

Heroku memiliki metode sentris agnostik bahasa untuk menentukan jenis proses Anda melalui file konfigurasi yang disebut Procfile – format untuk mendeklarasikan jenis proses yang menjelaskan bagaimana aplikasi Anda akan berjalan.

Procfile adalah file teks bernama Procfile ditempatkan di root aplikasi Anda yang berisi daftar jenis proses dalam aplikasi. Setiap jenis proses adalah deklarasi dari sebuah perintah yang dieksekusi ketika suatu proses dari tipe proses itu dieksekusi. Semua bahasa

dan kerangka kerja pada tumpukan Cedar menyatakan jenis proses web, yang memulai server aplikasi.
Berikut adalah isi dari contoh Procfile untuk aplikasi Sinatra:
web: bundle exec rackup config.ru -p $ PORT

Bagian sebelum: menentukan jenis proses, dan bagian sesudah: menentukan perintah untuk meluncurkan jenis proses ketika dyno menjalankan proses boot up.

Seseorang dapat menggunakan variabel lingkungan yang dihuni oleh Heroku dalam perintah seperti yang ditunjukkan pada perintah sebelumnya.

Contoh lain dari perintah Procfile adalah :
web: sh path/bin/webapp

Dalam contoh ini, ketika proses web dimulai, itu akan meluncurkan server aplikasi web yang terletak di path / bin / webapp di shell baru. Perhatikan betapa miripnya dengan menjalankan proses UNIX pada baris perintah.

Mendeklarasikan jenis proses

Tipe proses menyatakan namanya dan perintah baris perintah – ini adalah prototipe yang dapat dipakai dalam satu atau lebih proses yang sedang berjalan. Biasanya, Procfile akan berada di root aplikasi. Jika tidak disediakan atau dibuat, Heroku secara default membuat proses web untuk menjalankan aplikasi default.

Tujuan utama dari Procfile adalah untuk mendeklarasikan jenis proses yang berbeda dan mengaitkan suatu tindakan dengan setiap jenis proses. Pada dasarnya, ini adalah spesifikasi deklaratif tentang apa tipe proses individu yang tersedia untuk aplikasi ini dan bagaimana instance dari tipe proses tersebut harus dimulai atau dioperasikan ketika diperlukan. Jenis proses dapat mewakili berbagai jenis proses pekerja, pekerjaan jam, atau server Node.js yang menjalankan dan memproses acara yang masuk. Saat mendeklarasikan jenis proses baru, kami merekomendasikan agar kami membedakan antara jenis proses untuk pekerjaan sementara (berjalan untuk periode waktu kecil) dan jenis proses untuk pekerjaan yang berjalan lama. Ini membantu ketika Anda membuat contoh dari masing-masing jenis proses ini untuk dijalankan. Anda

dapat dengan mudah mengukur pekerjaan sementara secara terpisah dari pekerjaan yang sudah berjalan lama.

Format Procfile

Format Procfile adalah satu jenis proses per baris, dengan setiap baris berisi <proses ketik>: <perintah>.
Sintaks didefinisikan sebagai berikut :

- <tipe proses>: Ini adalah string alfanumerik; itu adalah nama untuk proses Anda, seperti web, pekerja, jam, atau myproc
- <command>: Baris perintah untuk meluncurkan proses, seperti sh path / bin / webapp

Jenis proses web adalah khusus dan merupakan satu-satunya jenis proses yang akan menerima lalu lintas HTTP dari jala perutean. Jenis proses lainnya dapat dinamai secara sewenang-wenang.

Sampel Procfile

Berikut adalah contoh Procfile untuk aplikasi Node.js dengan dua jenis proses:

- web : Ini menangani permintaan HTT
- worker : Ini menjalankan pekerjaan latar belakang;

  # Procfile
  web: node server.js
  pekerja: node bkgrnd.js

Saat Anda menjalankan aplikasi yang terkait dengan Procfile ini di Heroku, platform mendeteksi jenis proses yang diminta dan mem-bootnya, seperti yang ditunjukkan pada tangkapan layar berikut:

```
$ heroku create --stack cedar
$ git push heroku master
...
-----> Heroku receiving push
-----> Node.js app detected
...
-----> Discovering process types
       Procfile declares types -> web, worker
```

Menambahkan Procfile ke Heroku

Procfile belum tentu digunakan untuk menyebarkan aplikasi. Platform dapat mendeteksi jenis aplikasi dan runtime terkait untuk digunakan. Meskipun Heroku secara otomatis membuat jenis proses web default untuk mem-boot server aplikasi, disarankan untuk membuat Procfile eksplisit karena memberi Anda kontrol yang lebih besar tentang bagaimana proses Anda dapat dijalankan di lingkungan eksekusi Heroku. Untuk Heroku untuk digunakan

Procfile Anda, tambahkan Procfile ke root aplikasi Anda dan kemudian dorong ke Heroku, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ git add .
$ git commit -m "Procfile"
$ git push heroku
...
-----> Procfile declares process types: web, worker
       Compiled slug size is 10.8MB
-----> Launching... done
       http://gentle-mesa-5445.herokuapp.com deployed to Heroku

To git@heroku.com:gentle-mesa-5445.git
 * [new branch]      master -> master
```

Menjalankan aplikasi secara lokal

Seberapa sering kita menghadapi masalah di lingkungan produksi yang gagal mereproduksi di lingkungan pengembangan? Oleh karena itu, sangat penting bahwa ketika mengembangkan dan mendebug aplikasi, kode di lingkungan pengembangan lokal dieksekusi dengan cara yang sama seperti lingkungan jarak jauh. Ini memastikan bahwa setiap perbedaan antara dua lingkungan dan sulit ditemukan dicegat sebelum menyebarkan kode ke produksi.

Di lingkungan pengembangan lokal, Anda dapat menjalankan versi kecil aplikasi Anda dengan meluncurkan satu proses untuk masing-masing dari dua jenis proses: web dan pekerja.

Foreman adalah alat baris perintah yang digunakan untuk menjalankan aplikasi berbasis Procfile secara lokal. Itu diinstal secara otomatis oleh toolbelt Heroku (paket klien Heroku), dan juga tersedia sebagai permata.

Jika Anda tidak memiliki mandor yang diinstal, gunakan perintah mand install mandor untuk menginstalnya dari prompt.

Foreman pemula mudah seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ foreman start
09:27:10 web.1     | started with pid 22298
09:27:10 worker.1  | started with pid 22299
09:27:11 web.1     | Listening on port 5000
09:27:11 worker.1  | Worker ready to do work
```

Karena Procfile memiliki proses web dan pekerja, mandor akan memulai salah satu dari setiap jenis proses dengan output yang disisipkan di terminal Anda. Proses web Anda menggunakan port 5000, karena inilah yang disediakan mandor sebagai default dalam variabel lingkungan $ PORT. Proses web harus menggunakan nilai ini, karena digunakan oleh platform Heroku pada penyebaran juga. Anda dapat menguji aplikasi sekarang. Tekan Ctrl + C untuk mengirim sinyal untuk menutup aplikasi setelah selesai.

Mengatur variabel lingkungan local

Variabel yang disimpan dalam file .env dari direktori proyek akan ditambahkan ke lingkungan ketika dijalankan oleh mandor.

Misalnya, kita dapat mengatur SRCPATH_ENV untuk pengembangan di lingkungan Anda, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ echo " SRCPATH_ENV=SMS" >>.env #Put the SRCPATH_ENV variable into the .env file

$ foreman run irb
> puts ENV["SRCPATH_ENV "]
> SMS
```

Perintah dalam tangkapan layar sebelumnya menjalankan shell interaktif dan menetapkan nilai variabel lingkungan SRCPATH_ENV ke SMS.

Anda dapat menggunakan file .env untuk konfigurasi lokal karena memiliki pengaturan khusus untuk lingkungan lokal Anda. Itu tidak perlu diperiksa atau dilakukan untuk Heroku.

Proses pembentukan

Setiap set aplikasi proses yang berjalan pada manifold dyno dikenal sebagai proses pembentukannya.

Formasi default untuk aplikasi sederhana adalah proses web tunggal, di mana aplikasi yang lebih kompleks dapat terdiri dari beberapa salinan web dan jenis proses pekerja. Anda dapat menskalakan jenis proses sesuai permintaan karena kebutuhan aplikasi Anda meningkat.

Misalnya, jika aplikasi sudah memiliki proses pembentukan tiga proses web, dan Anda menjalankan heroku ps: skala pekerja = 3 perintah, Anda sekarang akan memiliki total enam proses: tiga proses web dan tiga proses pekerja.

Proses penskalaan

Dalam lingkungan produksi, aplikasi Anda mungkin perlu meningkatkan kapasitas yang jauh lebih besar. Ini bukan lagi beberapa proses web dan pekerja yang biasanya Anda perlukan untuk aplikasi yang digunakan di lingkungan lokal.

Heroku mengikuti arsitektur proses share-nothing. Ini memungkinkan instantiasi dari setiap jenis proses beberapa kali. Setiap proses dengan jenis proses yang sama memiliki perintah dan tujuan yang sama, tetapi dijalankan sebagai proses yang terpisah dan terisolasi di lokasi fisik yang berbeda.

Perintah heroku ps:scale memungkinkan Anda mengatur skala aplikasi ke ukuran apa pun karena tuntutan aplikasi Anda meningkat seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku ps:scale web=20 worker=40
Scaling web processes... done, now running 20
Scaling worker processes... done, now running 40
```

Perintah scale membuat 20 web dan 40 proses pekerja tambahan untuk menangani peningkatan aplikasi.

Seperti halnya heroku run, skala Heroku meluncurkan proses. Tetapi alih-alih meminta proses tunggal, sekali tembak yang melekat pada terminal, ia meluncurkan seluruh kelompoknya, mulai dari prototipe yang ditentukan dalam Procfile.

Dalam contoh sebelumnya, formasi proses adalah 20 proses web dan 40 proses pekerja. Setelah scaling out, Anda dapat melihat status formasi baru Anda dengan perintah heroku ps, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku ps
Process       State               Command
------------  ------------------  ------------------------------
web.1         up for 3s           node server.js
web.2         up for 2s           node server.js
...
…
web.20……
worker.1      starting for 4s     node bkgrnd.js
worker.2      up for 2s           node bkgrnd.js
...
…
worker.40…..
```

Manifold dyno akan menjaga proses ini tetap berjalan dalam formasi yang tepat ini, sampai Anda meminta perubahan dengan perintah skala Heroku lainnya. Menjaga proses Anda berjalan tanpa batas dalam formasi yang Anda minta adalah bagian dari ketahanan erosi Heroku.

Kuantitas penskalaan kuantitas juga dapat ditentukan sebagai kenaikan dari jumlah dino atau proses yang ada sebagai berikut:

```
$ heroku ps:scale web+2
Scaling web processes... done, now running 22
```

Menghentikan jenis proses

Untuk berhenti menjalankan jenis proses tertentu sepenuhnya, cukup skala ke nol seperti yang ditunjukkan pada tangkapan layar berikut:

```
$ heroku ps:scale worker=0
Scaling worker processes... done, now running 0
```

Memeriksa Proses Anda

Gunakan heroku ps untuk menentukan jumlah proses yang dieksekusi. Daftar menunjukkan tipe proses di kolom kiri, dan perintah yang sesuai dengan jenis proses di kolom kanan, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku ps
=== web: `bundle exec rails server -p $PORT`
web.1: up for 3m
```

Log proses

Untuk mendapatkan daftar agregat pesan log dari semua jenis proses, ketik kode seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku logs
2013-01-26T01:24:20-07:00 heroku[slugc]: Slug compilation finished
2013-01-26T01:24:22+00:00 heroku[web.1]: Running process with command: `bundle exec
rails server mongrel -p 46999`
2013-01-25T18:24:22-07:00 heroku[web.1]: State changed from created to starting
2013-01-25T18:24:29-07:00 heroku[web.1]: State changed from starting to up
2013-01-26T01:24:29+00:00 app[web.1]: => Booting Mongrel
```

Gunakan heroku log --ps pekerja untuk melihat hanya pesan dari jenis proses pekerja, seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku logs --ps worker
2013-01-25T18:33:25-07:00 heroku[worker.1]: State changed from created to starting
2013-01-26T01:33:26+00:00 heroku[worker.1]: Running process with command: `env
QUEUE=* bundle exec rake resque:work`
2013-01-25T18:33:29-07:00 heroku[worker.1]: State changed from starting to up
```

Perhatikan bahwa perintah telah memfilter pesan log dan ditampilkan pada pesan pekerja.1 di log.

Menjalankan proses satu kali

Arsitektur proses Heroku memungkinkan Anda untuk menjalankan sekelompok proses (proses informasi) atau proses satu kali yang berdiri sendiri. Di mesin lokal Anda, Anda dapat melakukan cd ke direktori dengan aplikasi Anda, lalu ketik perintah untuk menjalankan suatu proses. Di tumpukan Cedar Heroku, Anda dapat menggunakan heroku run untuk meluncurkan proses terhadap kode aplikasi yang digunakan di manifold dyno Heroku.

Untuk menjalankan perintah tanggal pada Heroku sebagai dyno mandiri, ketik kode seperti yang ditunjukkan pada tangkapan layar berikut:

```
$ heroku run date
Running `date` attached to terminal... up, run.2963
Thu Jan 12 08:25:21 UTC 2013
```

Perintah menampilkan tanggal pada terminal dengan menjalankan dyno mandiri dengan ID proses 2963. Masing-masing perintah ini dijalankan pada dyno baru yang berdiri sendiri berjalan di lokasi fisik yang berbeda. Setiap dino sepenuhnya terisolasi, dimulai dengan salinan bersih dari sistem file sementara aplikasi, dan seluruh dino (termasuk proses, memori, dan sistem file) dibuang ketika proses yang diluncurkan oleh perintah keluar atau diakhiri.

Menjalankan apa saja

Anda bisa menjalankan apa saja di platform Heroku. Jika Anda memerlukan fitur tambahan untuk aplikasi web Anda, Anda dapat menggunakan add-on yang didukung dan menggunakan layanan add-on dari aplikasi Anda dan memberikan kemampuan yang lebih baru seperti pemantauan (aplikasi) atau penyimpanan data dalam aplikasi Anda. Kemungkinannya tidak terbatas.

Katakanlah Anda ingin menukar server web dan sistem pekerja yang digunakan untuk aplikasi Rails Anda dan masing-masing menggunakan Unicorn dan Resque.

Anda hanya perlu mengubah Gemfile dan Procfile, dan hanya itu. kamu siap untuk pergi!

Perubahan Gemfile adalah sebagai berikut :

gem 'unicorn'
gem 'resque'
gem 'resque-scheduler'

Perubahan Procfile adalah sebagai berikut :

web: bundle exec unicorn -p $ PORT -c ./config/unicorn.rb
pekerja: bundle exec rake resque: work QUEUE = *
urgworker: bundle exec rake resque: work QUEUE = urgent

The urgworker adalah tipe proses kustom baru yang didefinisikan untuk menandakan tugas dengan tipe QUEUE yang mendesak. Anda mendapatkan fleksibilitas penskalaan jenis proses ini independen satu sama lain ketika dibutuhkan. Jadi, jika Anda ingin lebih banyak proses pemroses urgensi, yang perlu Anda lakukan hanyalah memanggil heroku ps: skala urgworker + 1 perintah dan voila, Anda siap dan menjalankan lebih banyak tugas pemroses antrian yang mendesak.

Perhatikan bagaimana Procfile menentukan dua proses pekerja, satu untuk antrian mendesak dan satu lagi untuk pemrosesan normal. Fleksibilitas semacam itu adalah kunci untuk menjalankan penyebaran aplikasi berskala besar.

Ringkasan

Dalam bab ini, kami menggali lebih dalam berbagai komponen yang membentuk platform Heroku. Kami memahami dasar dari platform Heroku, tumpukannya, dan bagaimana Celadon Cedar yang saat ini didukung menyediakan kemampuan berbeda yang membantu membangun dan skala aplikasi web dengan mulus. Kami juga melihat bagaimana permintaan pengguna dialihkan dan akhirnya dieksekusi di lingkungan eksekusi Heroku. Kami telah membahas secara singkat infrastruktur Logplex logging Heroku yang membantu melacak proses internal aplikasi web Anda. Kami juga melihat arsitektur tambahan Heroku yang membantu pengembang memperluas platform Heroku dan membangun kemampuan yang lebih baru untuk aplikasi mereka.

Kami belajar bagaimana menjalankan operasi Heroku secara terprogram dengan memanfaatkan API Platform Heroku dan akhirnya kami meninjau arsitektur proses Heroku satu-satunya subsistem terpenting dari platform Heroku.

Sekarang setelah kita memiliki pemahaman mendasar yang diperlukan dari platform Heroku, kita akan mulai belajar bagaimana membangun aplikasi web yang nyata, kuat, dan dapat diukur pada Heroku di bab berikutnya.

BAB 3
MEMBANGUN APLIKASI HEROKU

Pada bab sebelumnya, kami mengambil langkah pertama dalam memahami Heroku. Kami mempelajari berbagai komponen platform Heroku, dimulai dengan tumpukan platform dan diakhiri dengan API Platform yang digunakan untuk secara terprogram menjalankan operasi Heroku. Kami juga menggali lebih dalam ke dalam arsitektur proses Heroku, yang merupakan inti dari arsitektur platform Heroku.

Sekarang, kita akan mengambil langkah selanjutnya. Kami akan mempelajari lebih lanjut tentang Heroku, termasuk cara membuat aplikasi web yang dapat diskalakan pada platform Heroku termasuk mekanisme yang digunakan untuk konfigurasi dan menyediakan dukungan bahasa untuk aplikasi tersebut. Tetapi sebelum kita melakukan itu, kita juga akan melihat bahan-bahan utama atau prinsip-prinsip yang (harus) mengatur cara aplikasi web SaaS harus dikembangkan. Seperangkat prinsip ini disebut metodologi App Dua Belas-Faktor.

Pada akhir bab ini, Anda akan memiliki pemahaman yang baik tentang faktor-faktor berikut :

Prinsip-prinsip desain aplikasi untuk platform Heroku

- Proses pembuatan dan konfigurasi aplikasi pada platform Heroku

- Sistem peran yang dimainkan oleh sistem buildpack dalam mengaktifkan dukungan runtime bahasa di Heroku
- Bagaimana kode ditransformasikan menjadi siput dan bagaimana siput dioptimalkan untuk membangun lebih cepat, lebih efisien.

Ketika kita mengeksplorasi berbagai aspek Heroku dalam buku ini, kita akan menyadari betapa pentingnya pengaruh metodologi ini terhadap arsitektur Heroku.

Metodologi TFA terdiri dari pedoman berikut untuk membantu Anda mengembangkan aplikasi web SaaS yang dapat disesuaikan dan terukur.

Basis kode selalu diversi dan dapat memiliki banyak penyebaran

Menurut metodologi TFA, basis kode aplikasi selalu dilacak melalui sistem versi kode sumber. Database pelacakan revisi untuk basis kode disebut repositori atau repo singkatnya. Basis kode adalah repositori tunggal (kontrol kode sumber terpusat) atau satu set repositori yang berbagi basis atau versi root atau komit (seperti halnya dengan sistem Git SCC yang didesentralisasi).

Selalu ada korelasi satu-ke-satu antara basis kode dan aplikasi web. Jika ada beberapa basis kode, ini adalah sistem terdistribusi dengan setiap komponen dalam sistem terdistribusi menjadi aplikasi. Metodologi TFA kemudian berlaku untuk setiap aplikasi individual yang merupakan bagian dari aplikasi terdistribusi.

Sebagai konsekuensi wajar dari prinsip sebelumnya, beberapa aplikasi yang memiliki basis kode yang sama melanggar metodologi TFA. Jika perlu, bagian umum dari berbagai aplikasi harus diabstraksi menjadi perpustakaan yang kemudian dapat dimanfaatkan oleh masing-masing aplikasi melalui manajer dependensi.

Ini menyiratkan bahwa aplikasi web Anda seharusnya hanya memiliki satu basis kode tetapi dapat memiliki banyak penyebaran. Basis kode adalah sama di beberapa penyebaran. Bahkan jika versi berbeda dari aplikasi aktif di setiap penyebaran, aplikasi berbagi basis kode yang sama. Ada kemungkinan bahwa pengembang memiliki perubahan yang tidak dikomit atau lingkungan pementasan

telah melakukan belum mendorong untuk produksi, tetapi mereka semua berbagi kode yang sama, sehingga membuat ini (aplikasi pengembang atau aplikasi pementasan) muncul sebagai penyebaran terpisah dari aplikasi web yang sama.

Deklarasikan dan pisahkan dependensi secara eksplisit (selalu)

Metodologi TFA merekomendasikan untuk mendeklarasikan semua dependensi aplikasi secara eksplisit dan lengkap, tanpa bergantung pada keberadaan implisit dari pustaka atau paket di seluruh sistem. Jika Anda membutuhkan pustaka sistem atau alat, Anda harus menjualnya di aplikasi Anda, mengandalkan sistem di sekitarnya.

Aplikasi TFA harus menggunakan manifes pernyataan ketergantungan yang menyatakan semua dependensi yang mungkin dimiliki aplikasi Anda. Selain itu, metodologi TFA mengharuskan Anda untuk menggunakan alat isolasi dependensi selama eksekusi untuk memastikan bahwa tidak ada dependensi implisit merembes masuk dari luar.

Spesifikasi ketergantungan yang diletakkan berlaku untuk semua lingkungan yang berbeda seperti pengembangan dan produksi secara seragam. Contoh umum untuk deklarasi dependensi dalam aplikasi Ruby adalah manifes Gemfile. Bagian isolasi ketergantungan untuk aplikasi Ruby dicapai melalui bundle exec. Deklarasi ketergantungan dan isolasi keduanya harus dan harus digunakan bersama-sama agar sesuai dengan metodologi TFA. Manfaat yang jelas dari menggunakan keduanya secara bersamaan adalah bahwa ketika pengembang memulai proyek baru, semua yang diperlukan untuk memulai adalah runtime bahasa dan manajer dependensi. Pengembang tidak perlu mengasumsikan keberadaan perpustakaan di luar ini dan tidak perlu menginstal paket apa pun selain runtime dan dependency manager. Lingkungan pengembang menjadi mandiri dalam arti tertentu. Lingkungan memiliki segala yang dibutuhkan untuk membuat aplikasi, membangun, dan menyebarkannya.

Konfigurasi harus disimpan di lingkungan

Kalau dipikir-pikir, satu-satunya perbedaan antara dua penyebaran aplikasi adalah konfigurasi apakah itu URL basis data atau basis data atau nama host tempat aplikasi tersebut digunakan. Metodologi TFA mendasarkan prinsip ketiganya pada observasi ini – konfigurasi harus dipisahkan dari kode.

Sebuah aplikasi tidak boleh menyimpan konfigurasi di dalam kode karena berbagai jaminan lingkungan penyebaran mengubah ini dan yang secara langsung menyiratkan mengubah kode setiap waktu. Ini sama sekali tidak efisien. Namun, ini tidak berlaku untuk konfigurasi yang tetap sama di antara lingkungan penempatan. Biasanya, aplikasi yang mengikuti prinsip ini dapat dengan mudah dibuka bersumber tanpa harus membuat perubahan apa pun dalam kode.

Metodologi TFA merekomendasikan menyimpan konfigurasi aplikasi dalam variabel lingkungan. Pilihan ini tidak memerlukan perubahan kode ketika Anda berpindah antar deploys. Juga, variabel lingkungan dapat mengontrol aplikasi pada tingkat paling granular dan harus independen dari variabel lingkungan lainnya. Model pendefinisian skala konfigurasi ini secara linier karena jumlah penyebaran bertambah tanpa ketergantungan silang pada kode atau lingkungan penyebaran tertentu. Atau, banyak pengembang mencoba mengelompokkan konfigurasi menjadi nama seperti pengembangan atau pentahapan, tergantung pada lingkungan yang digunakan. Pendekatan ini tidak berskala juga karena selama periode waktu tertentu, jenis-jenis pengelompokan konfigurasi ini cenderung memadati sistem dan menjadi sangat sulit untuk dikelola.

Layanan backend harus diperlakukan sebagai sumber daya terlampir (digabungkan secara longgar).

Metodologi TFA menyarankan memperlakukan layanan dukungan sebagai layanan yang digabungkan secara longgar yang dapat dicolokkan ke dalam atau keluar dari aplikasi Anda tanpa mengubah kode. Ini tidak hanya membuat mengelola aplikasi lebih mudah, tetapi juga membantu meningkatkan skala aplikasi Anda lebih baik saat Anda membutuhkannya. Metodologi tidak

menganggap layanan pihak ketiga berbeda dari layanan lokal. Semua yang dibutuhkan aplikasi adalah pencari layanan melalui mand konfigurasi yang sudah diatur. Jika penyedia layanan berubah, memindahkan aplikasi ke penyedia layanan baru mudah dan transparan bagi pengguna. Yang perlu Anda lakukan adalah mengganti pencari lokasi dalam konfigurasi. Aplikasi tidak perlu berubah sama sekali.

Dengan demikian, layanan atau sumber daya dilampirkan ke aplikasi melalui konfigurasi dan dapat dihapus atau terlepas ketika diperlukan. Tidak ada ikatan.

Pemisahan yang ketat dari tahap build, release, dan run dari suatu aplikasi

Basis kode mengalami banyak tahap evolusi dari ditulis hingga dieksekusi selama runtime. Basis kode pertama kali dibangun, lalu dirilis, dan akhirnya dijalankan dalam lingkungan eksekusi.

Tahap build mengacu pada proses kompilasi kode sumber, menurunkan dependensi, dan membangunnya menjadi yang dapat dieksekusi. Tahap rilis menggabungkan executable dengan konfigurasi spesifik aplikasi (untuk penyebaran tertentu) dan membuat aplikasi siap untuk dijalankan. Tahap menjalankan adalah tempat aplikasi menemukan kehidupan dan dieksekusi di lingkungan runtime. Aplikasi memanifestasikan dirinya ke dalam serangkaian proses sistem yang berjalan khusus untuk penyebaran itu.

Metodologi TFA secara ketat mengisolasi tiga tahap ini dari satu sama lain. Setiap kode yang dapat dieksekusi dengan konfigurasi spesifiknya mendefinisikan rilis dan jika pengembang membuat perubahan pada kode atau konfigurasi, rilis baru dibuat. Setiap rilis secara unik ditentukan melalui stempel waktu atau urutan yang terus meningkat nomor (nomor rilis). Rilis tidak dapat dimodifikasi setelah dibuat. Perubahan pada rilis dapat berupa rilis baru atau rilis lama melalui proses rollback yang terdefinisi dengan baik dan diimplementasikan.

Tahap menjalankan dapat diotomatisasi melalui alat karena dapat dipicu kapan saja tanpa perubahan pada aplikasi itu sendiri. Dua tahap lainnya memanifestasikan diri lebih sering karena pengembang terus mengubah kode dan / atau mengkonfigurasi untuk memicu rilis baru. Tahap menjalankan harus memiliki kompleksitas minimum karena masalah yang dapat menghentikan proses dari menjalankan dapat muncul kapan saja dan mungkin perlu beberapa saat sebelum ada yang melihatnya.

Aplikasi dalam eksekusi adalah suatu proses atau banyak proses

Aplikasi web dalam eksekusi dapat memanifestasikan dirinya sebagai satu atau banyak proses yang berjalan bersamaan di lingkungan eksekusi. Metodologi TFA merekomendasikan bahwa proses tidak berbagi apa pun di antara mereka dan tidak memiliki kewarganegaraan. Jika proses perlu bertahan data untuk digunakan nanti, data harus ditulis ke toko persisten seperti database. Cache memori atau sistem file hanya dapat digunakan untuk transaksi berdurasi pendek dan data yang terkandung di dalamnya tidak dianggap ada untuk melayani permintaan di masa mendatang yang dapat dialihkan ke proses yang sama sekali baru. Setiap proses restart umumnya akan menghapus semua cache dan memori lokal. Oleh karena itu, tidak ada jaminan kegigihan yang diberikan untuk keadaan sementara proses ini. Pada baris yang sama, metodologi TFA juga memperlakukan penggunaan sesi tempel sebagai pelanggaran prinsip-prinsipnya karena aplikasi tampaknya mengasumsikan bahwa keadaan sebelumnya tersedia untuk permintaan di masa mendatang. Semua jenis kompilasi (misalnya, kompilasi aset) dilakukan selama tahap pembuatan tanpa dampak pada runtime dari sistem eksekusi proses yang direkomendasikan TFA.

Layanan harus diekspor melalui pengikatan pelabuhan

Aplikasi web yang mematuhi TFA mengekspor layanan, misalnya, layanan HTTP dengan mengikat ke port dan aplikasi mendengarkan permintaan apa pun yang tiba di alamat port itu. Menurut metodologi TFA, aplikasi web yang sesuai adalah mandiri

dan tidak tergantung pada injeksi runtime dari server web ke dalam lingkungan eksekusi untuk membangun layanan web untuk dikonsumsi oleh pengguna.

Bergantung pada lingkungan penyebaran, layanan ini dapat memiliki pelacak yang berbeda tetapi pelacak ini tidak bergantung pada kode. Misalnya, pengembang dapat menggunakan layanan dalam lingkungan pengembangan dari localhost melalui URL http: // localhost: 1234, sedangkan layanan yang sama ketika diekspos di lingkungan produksi dapat diakses melalui URL yang berbeda (termasuk port). Dalam produksi, lapisan perutean berpotensi menangani permintaan yang masuk dari nama inang publik dan mengikatnya ke layanan web terikat port.

Layanan web terikat port diimplementasikan dengan mendeklarasikan ketergantungan pada server web di aplikasi web Anda, sehingga server web adalah bagian dari build Anda dan seluruh proses adalah bagian dari kode pengguna. Selama runtime, layanan terikat ke port dan pengguna akhir kemudian dapat meminta layanan. Satu manifestasi penting dari mengekspos layanan seperti ini adalah bahwa satu layanan dapat dikonsumsi oleh yang lain hanya dengan mengakses URL layanan.

Aplikasi harus ditingkatkan melalui model prosesnya

Ada banyak cara untuk menjalankan aplikasi web. Salah satu caranya adalah ketika server web mem-fork proses anak baru untuk setiap permintaan klien baru (misalnya, Apache memalsukan proses untuk pemrosesan sisi-server PHP). Cara lain adalah ketika bahasa runtime / mesin virtual mengalokasikan sejumlah besar sumber daya sistem pada startup dan kemudian mengelola sumber daya tersebut secara internal ketika permintaan klien baru masuk (misalnya, JVM dalam aplikasi web Java). Dalam salah satu dari kasus ini, pengembang memiliki visibilitas yang sangat terbatas dari proses yang berjalan di latar belakang melayani permintaan klien. Metodologi TFA memberikan panduan tentang bagaimana skala proses aplikasi Anda ketika basis pengguna bersamaan dari aplikasi web meningkat dan meningkatkan skala web Anda aplikasi adalah satu-satunya cara untuk mendukung peningkatan beban.

Metodologi TFA merekomendasikan penggunaan **jenis proses** untuk menandakan jenis pekerjaan /pemrosesan yang dilakukan. Dengan menggunakan jenis proses, aplikasi web dapat mendistribusikan beban kerjanya berdasarkan sifat bawaan dari permintaan yang masuk. Misalnya, jika ini adalah permintaan HTTP, maka jenis proses tertentu yang disebut web dapat melayani permintaan jenis ini. Proses dari jenis ini dapat ditingkatkan secara independen dari proses jenis proses lainnya. Mungkin ada tipe proses yang disebut pekerja yang dapat digunakan untuk mengklasifikasikan proses yang berjalan lama dan dapat berjalan di latar belakang sementara sistem Anda siap memproses permintaan lainnya. Tipe-tipe proses yang didukung dan semua instance dari tipe-tipe tersebut bersama-sama membentuk apa yang disebut **formasi proses**. Seperti yang Anda lihat, contoh dari setiap jenis proses diisolasi dari contoh dari jenis lainnya dan dapat diskalakan secara independen. Prinsip-prinsip ini dapat diterapkan dengan sangat baik pada entitas eksekusi yang lebih terperinci yang disebut utas. Utas ada dalam suatu proses dan lakukan tugas tertentu dengan tumpang tindih minimum sumber daya bersama dengan utas lain dalam proses yang sama.

Kemampuan partisi horizontal dan tidak berbagi karakteristik apa pun dari proses ini membantu penskalaan dan memungkinkan menambahkan lebih banyak konkurensi ke aplikasi dengan mudah. Menurut metodologi TFA, aplikasi dapat terdiri dari beberapa proses yang berjalan pada beberapa node fisik sesuai dengan filosofi proses yang secara horizontal dapat diskalakan. Aplikasi yang sesuai harus selalu bergantung pada operasi manajer proses sistem untuk menangani logistik terkait proses seperti menanggapi contoh macet dan menangani proses reboot dan shutdown yang diprakarsai pengguna. Aplikasi yang sesuai dengan TFA tidak mengubah proses dan juga tidak menggunakan **pengidentifikasi proses** (**PID** cara unik untuk mengidentifikasi contoh proses dalam sistem) sebagai mekanisme untuk menyimpan kondisi pelaksanaan proses. Model ini secara signifikan dipengaruhi oleh model eksekusi sistem operasi UNIX untuk daemon layanan.

Startup lebih cepat dan shutdown yang anggun adalah cara untuk kelincahan dan skalabilitas aplikasi

Menurut metodologi TFA, proses suatu aplikasi adalah sekali pakai, yaitu, mereka dapat dimulai atau dimatikan tanpa pemberitahuan sama sekali. Akibatnya, menjadi sangat jelas bahwa proses harus dirancang agar memiliki waktu startup yang minimal dan shutdown dengan anggun ketika diperlukan. Manfaat nyata adalah bahwa proses dapat ditingkatkan sesuai permintaan dan perpindahan dari pengembangan ke produksi untuk aplikasi akan lebih cepat.

Waktu startup menandakan bahwa suatu proses dibutuhkan untuk mulai melayani permintaan. Waktu startup yang lebih singkat berarti rilis yang lebih cepat dan skalabilitas yang lebih baik. Juga, manajer proses yang bertanggung jawab dapat dengan mudah mengalihkan proses ke mesin fisik lain bila diperlukan tanpa menahan permintaan baru.

Shutdown anggun berarti suatu proses menerima permintaan penghentian dari manajer proses dan itu mematikan, membersihkan setiap kondisi sementara yang mungkin telah dibuat untuk sementara. Untuk proses web yang menangani permintaan HTTP, shutdown yang anggun dapat berarti menyelesaikan permintaan dalam penerbangan dan berhenti mendengarkan permintaan baru pada port yang ditunjuk. Dalam beberapa kasus, klien dari proses server dapat mencoba menghubungkan kembali ketika koneksi ke proses penutupan terputus. Untuk pekerjaan yang berjalan lama atau proses pekerja, shutdown yang anggun dapat berarti mengembalikan permintaan dalam penerbangan ke keadaan sebelumnya seperti keadaan pos pemeriksaan di mana pemrosesan dapat dimulai setelah proses dimulai ulang. Teknik ini juga berguna jika tiba tiba, shutdown mendadak, misalnya, dalam kasus kegagalan perangkat keras di mana TFA dapat menggunakan mekanisme antrian yang kuat untuk mengembalikan pekerjaan yang tertunda ke antrian. Metodologi TFA juga merekomendasikan untuk membuat kesalahan aplikasi toleran dan mampu menangani shutdown yang tidak diketahui dan tidak lincah.

Pengembangan dan produksi (dan segala sesuatu di antaranya) harus semirip mungkin.

Salah satu masalah utama dan sumber penderitaan tim operasi dalam pengembangan kode hari modern adalah perbedaan antara pengembangan dan lingkungan produksi. Pengembang terus mengubah aplikasi, kadang-kadang membutuhkan waktu berbulan-bulan untuk check-in versi baru dari kode (kesenjangan waktu) diikuti oleh tim operasi menyebarkan sesuatu yang paling tidak mereka kenal, dalam rentang waktu yang sangat singkat (kesenjangan keterampilan) dan akhirnya aplikasi harus menggunakan setumpuk alat yang sama sekali baru sangat berbeda dari lingkungan pengembang (kesenjangan teknis). Ini adalah contoh klasik mengapa segala sesuatu bekerja di satu lingkungan dan tidak di lingkungan lain meskipun semuanya terlihat sama, hampir juga.

Metodologi TFA berupaya menyelesaikan perbedaan ini dengan merumuskan prinsip bahwa kesenjangan antara pengembangan dan produksi harus sekecil mungkin. Prinsip ini memungkinkan penerapan berkelanjutan dengan meminta pengembang tidak hanya menulis kode tetapi juga menyebarkannya lebih cepat dan secara aktif terlibat dengan operasi dalam memantau perilaku kode dalam produksi. Selain itu, metodologi TFA merekomendasikan penggunaan metode untuk menjaga kesenjangan teknis antara dua lingkungan menjadi minimum.

Dalam banyak kasus, perbedaan antara pengembangan dan produksi muncul dari penggunaan layanan backend yang digunakan oleh kedua lingkungan. Sangat umum bagi pengembang untuk menggunakan layanan backend yang ringan dalam pengembangan apakah itu database (SQLite) atau cache (custom cache berbasis sistem file). Namun, ketika sistem yang sama digunakan dalam produksi, aplikasi harus menggunakan basis data skala produksi (**PostgreSQL**) atau cache terdistribusi (**Memcached**) untuk melayani beban yang diharapkan dari lingkungan produksi. Meskipun ini sangat masuk akal mengingat kompleksitas dan skala yang diperlukan untuk didukung dalam lingkungan yang relevan, metodologi TFA menyarankan sebaliknya.

Aplikasi yang sesuai dengan TFA harus menggunakan layanan dukungan yang sama dalam lingkungan pengembangan dan produksi, sehingga tidak ada kejutan yang tidak diinginkan muncul ketika pindah ke lingkungan produksi meskipun kode tetap sama.

Selama bertahun-tahun, pengembang telah menggunakan adaptor untuk memberikan fleksibilitas untuk dengan mudah memindahkan aplikasi dari satu lingkungan ke lingkungan yang lain meskipun layanan dukungan yang sesuai mungkin berbeda. Adaptor memberikan fleksibilitas tetapi mereka tidak harus menghindari masalah yang muncul dari penggunaan layanan dukungan yang berbeda. Juga, insentif untuk menggunakan berbagai jenis layanan backend dalam pengembangan dan produksi telah menjadi minimal. Banyak layanan dukungan (yang biasanya hanya digunakan untuk produksi) sekarang menjadi mudah digunakan pada lingkungan pengembangan melalui alat manajemen instalasi yang lebih baik.

Singkatnya, untuk masa pakai aplikasi secara keseluruhan, mempertahankan paritas dev/prod dapat berjalan lebih baik dan memberikan banyak manfaat dibandingkan dengan menggunakan berbagai rasa dari layanan dukungan di lingkungan yang berbeda penyebaran berkelanjutan dan tim operasi yang bahagia adalah beberapa di antaranya.

Aplikasi seharusnya hanya mencatat acara yang tidak mengelolanya.

Data pencatatan adalah jendela ke berbagai karakteristik perilaku aplikasi Anda. Log adalah aliran berkelanjutan dari peristiwa aplikasi yang dipesan waktu yang dihasilkan oleh aplikasi Anda saat berjalan. Dalam kebanyakan kasus, log yang dihasilkan akan dialihkan ke output standar untuk ditinjau pengembang atau tetap ke sistem file sebagai file log.

Metodologi TFA merekomendasikan bahwa aplikasi yang sesuai tidak boleh bertanggung jawab untuk mengelola log, apakah itu tentang merutekan acara log atau menyimpannya. Dalam pengembangan, pesan log dapat dialihkan ke output standar pengembang untuk pemecahan masalah. Sedangkan dalam pementasan atau produksi, aplikasi hanya menghasilkan log dan

memungkinkan lingkungan eksekusi mengambil peristiwa log dari berbagai proses yang berjalan, menyusunnya dengan cara yang dipesan waktu, dan kemudian merutekannya ke terminal yang akan dilihat atau sistem file untuk penyimpanan jangka panjang.

Lingkungan eksekusi sepenuhnya mengelola internal tempat log disimpan dan bagaimana dialihkan dan bertahan. Aplikasi ini tidak memiliki visibilitas atau kemampuan untuk mengkonfigurasi detail sistem logging. Pendekatan ini memberikan fleksibilitas besar dalam hal pemrosesan lebih lanjut dari data log aplikasi. Data dapat dikirim secara real time ke alat analisis untuk memperoleh informasi penting tentang aplikasi atau untuk menemukan kejadian peristiwa tertentu, selain memicu peringatan ketika kondisi tertentu terpenuhi pada formulir agregat atau data log yang berkelanjutan.

Tugas administrasi atau manajemen aplikasi harus dijalankan sebagai proses satu kali

Sebagai pengembang atau anggota tim operasi, Anda kadang-kadang perlu menjalankan tugas administrasi atau manajemen aplikasi tertentu. Tugas semacam itu dapat mencakup migrasi basis data atau skrip untuk menjalankan perbaikan instalasi satu kali. Tugas-tugas ini, meskipun berbeda dari tugas-tugas reguler aplikasi dalam hal frekuensi menjalankannya, tidak berbeda dalam hal bagaimana mereka dijalankan. Tugas-tugas ini adalah bagian dari aplikasi yang sama, berjalan melawan rilis tertentu dan menggunakan konfigurasi dan kode yang sama yang digunakan tugas berjalan lama pada sistem. Aturan isolasi ketergantungan yang sama harus berlaku untuk tugas-tugas tersebut sesuai dengan metodologi TFA.

Aplikasi yang sesuai dengan TFA menjalankan tugas dengan cara standar di lingkungan yang sama dengan tugas biasa, tidak membedakan antara yang berjalan berkali-kali versus yang berjalan sekali. Misalnya, proses startup proses web (berjalan lebih sering) dan migrasi database (berjalan satu kali) menggunakan bundle exec directive / commandfollowed oleh tugas tertentu untuk melakukan tugas yang tampaknya berbeda.

Metodologi TFA juga merekomendasikan agar kode untuk tugas-tugas administrasi atau manajemen dikirimkan bersama aplikasi agar menjaga basis kode konsisten dan menghindari perbedaan antara berbagai versi aplikasi.

Pengembang dapat menggunakan prinsip TFA untuk membangun aplikasi web yang kuat di platform Heroku. Di sepanjang buku ini, Anda akan menghargai banyak fitur Heroku yang mendukung pembentukan prinsip-prinsip ini.

Sekarang kita memahami prinsip-prinsip yang mengatur untuk mengembangkan aplikasi web, mari kita buat satu dan lihat cara kerjanya.

Membuat aplikasi Heroku

Heroku memulai dengan mendukung Ruby on Rails (RoR) sebagai kerangka kerja default untuk mengembangkan aplikasi web. Dengan RoR lahir filosofi inti dari platform Heroku - konvensi tentang konfigurasi. Filosofi ini berarti bahwa seorang pengembang tidak harus membuat terlalu banyak keputusan saat mengembangkan suatu aplikasi dan pada saat yang sama tidak boleh kehilangan fleksibilitas mengembangkannya dengan cara yang dianggap tepat. Selama periode waktu tertentu, Heroku menambahkan dukungan untuk bahasa lain seperti Java, Node.js, dan lainnya sambil mengaktifkan filosofi dasar yang sama untuk mengembangkan aplikasi pada platform Heroku.

Mari kita lihat manifestasi nyata dari filosofi ini dengan membuat aplikasi web. Proses pembuatan aplikasi Heroku dapat dipecah (untuk kesederhanaan) menjadi langkah-langkah berikut :

1. Mengetahui dan memuaskan prasyarat untuk aplikasi.
2. Menulis aplikasi inilah yang idealnya paling harus dikhawatirkan oleh pengembang.
3. Menambahkan aset yang sudah ada dan dapat digunakan kembali ke aplikasi melalui add-on.
4. Mengkonfigurasi aplikasi untuk mengatur lingkungan aplikasi.
5. Menyebarkan aplikasi di Heroku.
6. Memantau dan meningkatkan skala aplikasi jika perlu.

Di bagian ini, mari kita membangun aplikasi RoR dan sambil melakukannya, pahami langkah-langkah yang terlibat dalam mengembangkan aplikasi web dunia nyata di Heroku.

Untuk memulai, Anda memerlukan prasyarat berikut :

- Akun Heroku
- Klien Heroku (toolbelt – https: //toolbelt.heroku.com) diinstal pada mesin Anda
- Ruby, Rails 3, Rubygems, dan Bundler diinstal pada mesin Anda
- Kemampuan untuk mengembangkan aplikasi Rails dasar
- Pemahaman dasar tentang bagaimana mendorong kode Anda ke Git
-

Berikut ini adalah langkah-langkah selanjutnya yang perlu Anda ambil untuk membuat aplikasi RoR :

1. Dengan anggapan Anda telah menginstal Heroku toolbelt, Anda akan memiliki akses ke antarmuka baris perintah (CLI) Heroku dari mesin Anda. Anda dapat menggunakan Heroku CLI untuk mengeluarkan perintah yang akan dijalankan pada platform Heroku.
2. Buka jendela shell perintah Windows (atau Unix).
3. Terhubung ke platform Heroku dengan menggunakan perintah login heroku sebagai berikut :

```
$ heroku login
Enter your Heroku credentials.
Email: abcdef@ghijklm.com
Password:
Could not find an existing public key.
Would you like to generate one? [Yn]
Generating new SSH public key.
Uploading ssh public key /Users/abcdef/.ssh/id_rsa.pub
```

   Ketik Y saat diminta untuk membuat dan mengunggah kunci SSH publik baru ke akun Heroku. Untuk detail tentang cara kerjanya, silakan merujuk ke Bab 8, Keamanan Heroku.
4. Tulis aplikasi Rails tanpa tulang atau gunakan aplikasi Rails yang ada untuk penyebaran.
5. Kami akan membuat aplikasi Rails dasar untuk memenuhi persyaratan minimum langkah ini sebagai berikut:

```
$ rails new herokuclouddevapp --database=postgresql
$ cd herokuclouddevapp
```

- Argumen --database dilewatkan untuk menggunakan database Heroku PostgreSQL sebagai penyimpan data untuk aplikasi Anda. Dianjurkan agar Anda menggunakannya bukan SQLite, yang merupakan penyimpanan data pengembangan default sebelumnya. Memilih database ini akan membantu menjaga konfigurasi pengembangan sedekat mungkin dengan konfigurasi produksi aplikasi, sehingga mengurangi kemungkinan memiliki masalah produksi yang terkait dengan perbedaan jenis database.
- Pastikan file config / database.yml untuk aplikasi Rails Anda menunjuk ke adaptor PostgreSQL dan berisi pengaturan yang tepat dan kredensial pengguna untuk lingkungan aplikasi yang didukung.

```
development:
  adapter: postgresql
  encoding: unicode
  database: herokuclouddevapp_development
  pool: 3
  username: herokucda
  password:
production:
  adapter: postgresql
  encoding: unicode
  database: herokuclouddevapp_production
  pool: 3
  username: herokucda
  password:
...
```

- Perbarui dependensi proyek Anda dengan menjalankan perintah instal bundel.

6. Jika Anda menggunakan Rails 4, Anda mungkin tidak perlu melakukan ini tetapi jika Anda menggunakan Rails 3 (seperti kami), Anda perlu melakukan konfigurasi berikut dalam file config / application.rb dari proyek rails Anda:

```
config.assets.initialize_on_precompile = false
```

Sekarang mengapa ini diperlukan? Nah, Rails 3 memiliki konsep pipeline aset, yang merupakan metode untuk menggabungkan

dan mengkompresi aset JavaScript dan CSS yang tersedia dalam proyek. Selama proses pembuatan, Heroku mengkompilasi aset Anda ke siput agar mudah tersedia untuk ditempatkan. Untuk mempercepat langkah pra-kompilasi aset, praktik terbaik adalah memberi tahu Rails untuk memuat aplikasi Anda hanya sebagian. Heroku tidak menyediakan seluruh lingkungan aplikasi untuk proses pembangunan, karenanya membuat pengaturan ini wajib.

7. Inisialisasi repositori Git Anda dan simpan kode Anda di sana seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ git init
$ git add .
$ git commit -m "first version"
```

8. Buat aplikasi Heroku Anda seperti yang ditunjukkan pada tangkapan layar berikut :

```
$ heroku create
Creating herokuclouddevapp... done, stack is cedar
http://herokuclouddevapp.herokuapp.com/ | git@heroku.com:herokuclouddevapp.git
Git remote heroku added
```

9. Sebarkan kode Anda ke Heroku. Langkah yang memicu seluruh proses pembuatan untuk aplikasi Anda adalah sebagai berikut :
   - Pengguna harus berada dalam direktori aplikasi untuk mulai mendorong kode. Pengguna mengeluarkan dorongan git untuk mulai membangun aplikasi sebagai berikut:

```
$ git push heroku master
Counting objects: 63, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (49/49), done.
Writing objects: 100% (63/63), 26.03 KiB, done.
Total 63 (delta 2), reused 0 (delta 0)
-----> Ruby/Rails app detected
```

   - Bundler alat manajemen ketergantungan untuk Ruby, mencoba menyelesaikan dependensi pada permata / paket lain dan menginstalnya seperti yang diperlihatkan dalam baris perintah berikut :

```
Bundler will do a full resolve so native gems are handled properly.
       This may result in unexpected gem versions being used in your app.
-----> Installing dependencies using Bundler version 1.3.0.pre.5
       Running: bundle install --without development:test --path vendor/bundle --binstubs vendor/bundle/bin
       Fetching gem metadata from https://rubygems.org/.....
       Fetching gem metadata from https://rubygems.org/..
       Installing rake (10.0.3)
       Installing i18n (0.6.1)
       Installing multi_json (1.5.0)
       Installing activesupport (3.2.9)
      Your bundle is complete! It was installed into ./vendor/bundle
```

- Memuat konfigurasi basis data dari lingkungan (DATABASE_URL), menyelesaikan prakompilasi aset Rails dan penyisipan plugin sebagai berikut :

```
-----> Writing config/database.yml to read from DATABASE_URL
-----> Preparing app for Rails asset pipeline
       Running: rake assets:precompile
       Asset precompilation completed (11.90s)
-----> Rails plugin injection
       Injecting rails_log_stdout
       Injecting rails3_serve_static_assets
```

- Akhirnya, proses build mendeteksi tipe proses yang didukung (baik dengan membaca Procfile atau menggunakan default). Juga, ini menghitung ukuran siput dan meluncurkan aplikasi dari http: // <nama aplikasi> .herokuapp.com sebagai berikut :

```
-----> Discovering process types
       Procfile declares types      -> (none)
       Default types for Ruby/Rails -> console, rake, web, worker
-----> Compiled slug size: 9.0MB
-----> Launching... done, v9
       http://herokuclouddevapp.herokuapp.com deployed to Heroku

To git@heroku.com:herokuclouddevapp.git
 * [new branch]      master -> master
```

Siput yang dihasilkan dari proses build ini sekarang dapat di-boot pada manifold dyno bersama aplikasi yang ditulis dalam bahasa lain.

Jadi, kita selesai membangun aplikasi Rails 3 sederhana. Sekarang, mari kita lihat konsep variabel konfigurasi, juga disebut vars konfigurasi. Konfigurasi konfigurasi adalah cara untuk mempengaruhi pelaksanaan aplikasi Anda tergantung pada fase penyebaran, misalnya, pengembangan, pengujian, pementasan, atau produksi. Config vars adalah fokus dari bagian selanjutnya.

Mengkonfigurasi aplikasi Heroku Anda

Sebagian besar penerapan aplikasi bisa berantakan; beberapa penyebaran situs produksi, beberapa test bed, lingkungan pementasan, dan siapa yang tahu berapa banyak lingkungan pengembangan yang dipelihara secara lokal oleh pengembang. Selain itu, mungkin ada ribuan penyebaran untuk paket sumber terbuka standar yang paling populer. Ini menimbulkan tantangan besar untuk menggunakan informasi konfigurasi aplikasi Anda yang mungkin berubah tergantung pada lingkungan tempat seseorang bekerja. Meskipun kode sumber yang dijalankan adalah sama, hampir setiap kali konfigurasi terdiri dari sejumlah besar informasi yang spesifik untuk lingkungan. Contoh umum termasuk kredensial pengguna, konfigurasi basis data, dan sebagainya. Penggunaan file properti cukup umum untuk mendukung persyaratan seperti itu. Namun, pendekatan seperti itu memiliki jebakan karena file properti dengan cepat mereplikasi diri mereka sendiri dengan cukup tidak terkelola.

Heroku menyediakan API terkait konfigurasi berbasis CLI yang kaya untuk mengelola informasi konfigurasi aplikasi.

API konfigurasi aplikasi Heroku

Heroku menggunakan konsep variabel konfigurasi untuk konfigurasi aplikasi banyak pada baris variabel lingkungan, seperti yang digunakan dalam rasa Unix.

API konfigurasi aplikasi Heroku terdiri dari perintah berikut :

- config: add: Perintah ini menambahkan parameter konfigurasi
- config: get: Perintah ini mendapatkan nilai dari variabel konfigurasi
- config: remove: Perintah ini menghapus variabel konfigurasi

Variabel konfigurasi menggantikan variabel lingkungan untuk aplikasi. Variabel lingkungan ini tetap ada di seluruh deploys dan restart aplikasi dan hanya perlu ditetapkan satu kali. Jika Anda ingin mengubah variabel konfigurasi, Anda dapat menggunakan konfigurasi: tambahkan API dan atur variabel konfigurasi ke nilai baru. Setiap kali Anda menambah atau memperbarui variabel

konfigurasi (termasuk menghapus variabel konfigurasi), aplikasi akan dihidupkan ulang.

Kode dapat mengekstraksi nilai variabel konfigurasi berdasarkan nama variabel konfigurasi (yang bertindak sebagai kunci) dan menjalankan aliran kode tertentu; misalnya, jika pengembang ingin menggunakan variabel konfigurasi bernama TRY_COUNT, yang menandakan jumlah upaya yang harus dilakukan suatu aplikasi untuk menyambungkan ke database sebelum menghitung waktu dan melontarkan kesalahan, config: add API dapat digunakan untuk mengatur konfigurasi variabel TRY_COUNT ke beberapa nilai, misalkan 10. Kode akan mencoba untuk terhubung ke database TRY_COUNT beberapa kali sebelum melemparkan kesalahan pada output standar.

Contoh penggunaan konfigurasi aplikasi

Mari kita lihat beberapa contoh menggunakan API konfigurasi. Kami akan menunjukkan operasi yang paling umum digunakan saat mengembangkan aplikasi web. Mereka adalah, menambahkan, memverifikasi, mengambil, dan menghapus variabel konfigurasi seperti yang diberikan pada langkah-langkah berikut :

1. Untuk menambahkan variabel konfigurasi, ketikkan perintah berikut :
   $ heroku config:add GITHUB_USERNAME=whizkid
   Menambahkan var konfigurasi dan me-restart aplikasi ... selesai, v3
   GITHUB_USERNAME: whizkid
2. Untuk memeriksa semua variabel konfigurasi yang tersedia untuk aplikasi, ketikkan perintah berikut:
   $ heroku config
   GITHUB_USERNAME: whizkid
   OTHER_VAR: produksi
3. Untuk mengambil variabel konfigurasi, ketikkan perintah berikut:
   $ heroku config:get GITHUB_USERNAME
   whizkid

4. Untuk menghapus variabel konfigurasi, ketikkan perintah berikut:
   $ heroku config:remove GITHUB_USERNAME
   Membatalkan pengaturan GITHUB_USERNAME dan memulai ulang aplikasi saya ... selesai, v4

Kegigihan variabel konfigurasi

Variabel konfigurasi yang dibahas di bagian sebelumnya bersifat persisten - variabel tersebut akan tetap berada di seluruh deploys dan aplikasi restart. Kecuali jika Anda perlu mengubah ini, Anda dapat membuatnya sekali dan menggunakannya berulang kali. Data variabel konfigurasi (kumpulan semua kunci dan nilai) dibatasi hingga 16 KB untuk setiap aplikasi.

Mengakses variabel konfigurasi saat runtime

Anda dapat membaca variabel saat runtime menggunakan metode khusus bahasa. Misalnya, untuk mengakses variabel lingkungan yang disebut DATABASE_URL di Ruby, ketikkan perintah berikut :

```
ENV['DATABASE_URL']
```

Setelah penggelaran ke Heroku, aplikasi akan menggunakan kunci yang diatur dalam direktori config.

Batasan pada data konfigurasi

Ada batas atas pada ukuran data variabel konfigurasi (termasuk kunci dan nilai) untuk setiap aplikasi pada platform Heroku. Batas saat ini pada data variabel konfigurasi diatur ke 16 KB per aplikasi.

Plugin Heroku menyediakan cara untuk mengambil semua variabel konfigurasi dari aplikasi dan menempatkannya di lingkungan lokal pengembang. Kebalikannya juga dimungkinkan jika diperlukan.

Menggunakan plugin konfigurasi Heroku

Untuk menarik konfigurasi Anda secara interaktif, meminta setiap nilai ditimpa dalam file lokal Anda, ketikkan perintah berikut :

$ heroku config:pull --overwrite –interactive

Untuk mendorong konfigurasi lokal Anda ke Heroku, gunakan perintah config: push sebagai berikut :

$ heroku config:push

Jadi, kami menulis aplikasi Rails 3 sederhana dan belajar cara mengatur variabel konfigurasi untuk aplikasi di platform Heroku. Satu pertanyaan yang mungkin Anda miliki adalah, bagaimana cara Heroku memetakan kode yang kami kirim menggunakan perintah git push ke runtime bahasa yang akan digunakan untuk membangun kode? Yah, Heroku menggunakan konsep buildpacks untuk mengaktifkan runtime bahasa yang dibutuhkan untuk aplikasi yang ditulis dalam bahasa yang didukung.

Memperkenalkan buildpacks

Untuk mendukung bahasa di Heroku, Anda perlu membuat aplikasi Anda yang ditulis dalam bahasa itu di atas adaptor build-time yang dapat mengompilasi aplikasi ke dalam program yang dapat dieksekusi yang cocok untuk dijalankan pada tumpukan Cedar. Adaptor build-time ini dikenal sebagai buildpack dalam bahasa Heroku. Cedar stack menyediakan runtime universal di mana dukungan bahasa dicolokkan melalui buildpack. Buildpack bertanggung jawab untuk membangun lingkungan runtime kerja yang lengkap di sekitar aplikasi. Ini mungkin termasuk bahasa VM dan ndependencies runtime lain yang diperlukan oleh aplikasi. Buildpack Anda perlu menyediakan binari ini dan menggabungkannya dengan kode aplikasi.

Heroku, secara default, menyediakan dukungan buildpack untuk semua bahasa yang didukung secara asli. Buildpack default ini tersedia untuk semua aplikasi Heroku selama kompilasi dan dapat diunduh dari GitHub. Ketika Anda mendorong aplikasi Anda ke Heroku, buildpack ini dicari untuk menemukan runtime yang sesuai untuk digunakan untuk aplikasi tersebut.

Anda dapat menemukan buildpack apa pun di GitHub dengan mengikuti format URL berikut :
https://github.com/heroku/heroku-buildpack-<Language>
Misalnya, buildpack umum dapat ditemukan di tautan berikut :

Languange	URL
Ruby	https://github.com/heroku/heroku-buildpack-ruby
Node.js	https://github.com/heroku/heroku-buildpack-nodejs
Clojure	https://github.com/heroku/heroku-buildpack-clojure
Python	https://github.com/heroku/heroku-buildpack-python
Java	https://github.com/heroku/heroku-buildpack-java

Namun, jika Anda ingin menggunakan buildpack berbeda untuk bahasa yang didukung atau ingin menambahkan dukungan untuk bahasa baru, Anda dapat menggunakan / menulis buildpack kustom.

Menggunakan buildpack khusus

Untuk menggunakan buildpack kustom, yang perlu Anda lakukan adalah mengganti buildpack default Heroku dengan menentukan buildpack kustom di variabel konfigurasi BUILDPACK_URL seperti berikut pada CLI Heroku :

$ heroku config:add BUILDPACK_URL=https://github.com/heroku/heroku-buildpack-mylang

Perintah ini mengarahkan BUILDPACK_URL ke custom buildpack menggunakan perintah Heroku config: add. Anda sekarang dapat menggunakan buildpack untuk bahasa yang ditentukan oleh mylang.

Menentukan buildpack khusus pada tahap pembuatan aplikasi

Kami dapat menentukan buildpack khusus selama langkah pembuatan aplikasi sebagai berikut :

$ heroku create myapp --buildpack https://github.com/ heroku/ heroku-
buildpack-mylang

Anda dapat menentukan versi persis buildpack dengan menggunakan revisi Git di BUILDPACK_URL sebagai berikut :

git://repo.git#master git://repo.git#v1.2.0

URL Buildpack dapat mengarah ke repositori atau tarbal Git.

Buildpack pihak ketiga

Buildpack pihak ketiga berisi perangkat lunak yang tidak berada di bawah kendali Heroku. Jika Anda ingin menggunakan buildpacks ini, Anda perlu meninjau kode sumber buildpack dengan hati-hati dan memastikannya aman untuk digunakan dan tidak menyebabkan kerentanan keamanan apa pun. Juga, buildpacks ini mungkin tidak up-to-date dengan versi terbaru atau rilis runtime bahasa.

API buildpack

Anda dapat membuat paket build khusus untuk menggunakan aplikasi Heroku yang ditulis dalam bahasa yang tidak didukung secara default. Heroku menawarkan buildpack API untuk membantu pengembang membuat buildpack untuk bahasa dan kerangka kerja yang lebih baru. Setelah ditulis, buildpacks ini dapat diperiksa ke repositori kode sumber umum seperti GitHub dan digunakan oleh pengembang untuk membangun aplikasi Heroku mereka.

Pengembang bahkan dapat mengesampingkan buildpacks default dengan memberikan URL alternatif untuk buildpack utama selama proses build. Anda dapat melakukan ini untuk menggunakan buildpack yang ditingkatkan yang mungkin menyediakan dukungan yang lebih baru untuk bahasa yang relevan sementara buildpack yang ada mungkin bertanggal.

Komponen API buildpack

Buildpack terdiri dari tiga skrip; mereka adalah sebagai berikut :

Script name Purpose

bin/detect Mendeteksi apakah paket ini dapat diterapkan ke aplikasi

bin/compile Skrip ini bertanggung jawab untuk mengonversi ke status runnable diHeroku

bin/release Skrip ini memberi informasi mendata tertentu seperti add-on untuk menginstal

Nampan / deteksi skrip

Penggunaan skrip ini adalah bin / deteksi <build_dir>. Berikut ini adalah uraiannya :

- Skrip / bin / deteksi menerima satu argument
- Skrip ini menerima <build_dir> sebagai argument
- Script pada eksekusi harus mengembalikan kode keluar 0 jika aplikasi yang ada di <build_dir> dapat ditangani oleh buildpack ini
- Jika kode keluar adalah 0, skrip harus mencetak nama kerangka kerja yang dapat dibaca manusia ke stdout

Berikut ini adalah kode sampel :

```
#! / bin / sh
# paket ini valid untuk aplikasi dengan Makefile di root
jika [-f $ 1 / Makefile]; kemudian
 gema "aplikasi GNU C"
 keluar 0
lain
 keluar 1
fi
```

Skrip memeriksa apakah Makefile ada di <build_dir> tempat kode aplikasi berada, dan begitu Makefile aplikasi terdeteksi, ia mencetak pesan kepada pengguna di stdout dan keluar dengan kode keluar 0 (sukses). Jika Makefile tidak terdeteksi, skrip shell keluar dengan kode kesalahan bukan nol. Kode kesalahan bukan nol

menandakan kondisi kesalahan yang memerlukan tindakan dari pihak pengembang. Pengembang harus memastikan dalam hal ini bahwa Makefile yang benar ada di direktori yang ditentukan sebagai argumen baris perintah.

Nampan / kompilasi skrip

Penggunaan skrip ini adalah bin / kompilasi <build_dir> <cache_dir>. Berikut ini adalah uraiannya :

- <build_dir> adalah lokasi aplikasi
- <cache_dir>adalah lokasi yang digunakan buildpack untuk menyimpan cache artefak di antara beberapa proses proses build

Aplikasi di <build_dir> bersama dengan semua perubahan yang dibuat oleh skrip kompilasi akan dikonversi menjadi siput yang dapat dieksekusi pada sebuah dyno di lingkungan Heroku.

Konten <cache_dir> dipertahankan antara build untuk meningkatkan kinerja build karena direktori ini digunakan untuk men-cache hasil dari tugas yang rumit dan memakan waktu seperti resolusi dependensi. Direktori <cache_dir> hanya tersedia selama kompilasi slug, dan khusus untuk aplikasi yang sedang dibangun. Konten lengkap dari direktori <cache_dir> disimpan dengan repositori Git dan harus dikirimkan melalui jaringan setiap kali aplikasi dikerahkan; karenanya, direktori ini harus berukuran tepat untuk menghindari menyebabkan keterlambatan yang signifikan untuk proses pembangunan. Untuk menggunakan cache, aplikasi harus membuat direktori <cache_dir> jika tidak ada.

Semua output yang diterima pada stdout dari skrip ini akan ditampilkan kepada pengguna. Berikut ini adalah kode sampel :

```
#!/usr/bin/env bash
# bin/compile <build-dir> <cache-dir>
set -e
set -o pipefail
BUILD_DIR=$1
CACHE_DIR=$2
function indent() {
```

```
 c='s/^/ /'
 case $(uname) in
 Linux) sed -u "$c";;
 *) sed -l "$c";;
 esac
}
cd $BUILD_DIR
# configure
if [ -f configure ]; then
 echo "-----> Configuring"
 ./configure 2>&1 | indent
fi
# make
echo "-----> Compiling with Make"
make 2>&1 | indent
```

Script ini mengubah direktori ke direktori build tempat aplikasi disimpan dan menjalankan skrip configure bernama .configure untuk mengatur aplikasi. Script configure memeriksa apakah prasyarat yang diperlukan untuk membangun aplikasi tersedia. Setelah bagian konfigurasi kompilasi selesai, skrip menjalankan kemampuan untuk membuat executable untuk aplikasi Anda.

Nampan / lepaskan skrip

Penggunaan skrip ini adalah bin / release <build_dir>, di mana <build_dir> adalah lokasi aplikasi.

Skrip bin / rilis memberikan informasi metadata tertentu kembali ke runtime.

Berikut ini adalah kode sampel :

```
#!/bin/sh
cat << EOF
--addons:
 - heroku-postgresql:dev
default_process_types:
 web: bin/node server.js
EOF
```

Skrip ini mengembalikan informasi tentang add-on yang akan diinstal dan tipe default process , yang merupakan hasil dari entri Procfile default. Sekarang, mari kita lihat apa yang diperlukan untuk menulis buildpack baru.

Menulis buildpack

Menulis buildpack terdiri dari langkah-langkah berikut :

1. Menulis tiga skrip: deteksi, kompilasi, dan lepaskan. Setiap skrip ini memiliki tujuan yang berbeda untuk prosedur pembuatan.
2. Check-in ketiga skrip ke GitHub dalam struktur folder standar. Misalnya, jika nama pengguna GitHub adalah johndoe, maka buildpack dapat disimpan di johndoe / heroku-buildpack-mylang / bin.
3. Gunakan konfigurasi berikut: tambahkan perintah untuk menambahkan BUILDPACK_URL baru untuk digunakan dengan aplikasi Anda:
   $ heroku config: tambahkan
   BUILDPACK_URL = https: //github.com/heroku/heroku-buildpack- mybuildpack.
4. Anda juga dapat menentukan buildpack selama pembuatan aplikasi sebagai berikut:
   $ heroku buat myapp --buildpack
   https://github.com/heroku/heroku-buildpack-mybuildpack

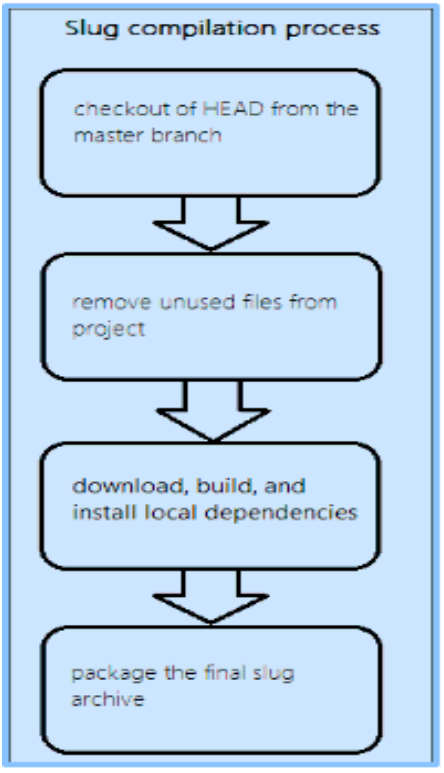
Setelah Anda menulis, mengonfigurasi, dan menautkan aplikasi Anda ke buildpack yang tepat, sekarang saatnya untuk membangun aplikasi dan mengubahnya menjadi executable yang dapat dijalankan pada platform Heroku. Alat compiler siput membantu kita dengan itu.

Kompiler siput

Compiler slug bertanggung jawab untuk membuat aplikasi yang dapat dieksekusi dari kode mentah dengan membangun aplikasi, menautkan binari yang diperlukan, dan mengompres executable untuk penyebaran yang lebih cepat. Produk akhir dari proses kompilasi siput adalah apa yang disebut sebagai siput.

Setelah siput dihasilkan, mereka dikompres untuk mengurangi ukurannya. Slug adalah salinan pra-paket dari aplikasi yang dioptimalkan untuk distribusi cepat di manifold dyno – lingkungan proses eksekusi Heroku. Ketika kita git push ke Heroku, kode dikirim ke kompiler slug yang mengubah repositori pengguna menjadi slug. Penskalaan aplikasi kemudian menyebabkan siput diunduh dan diperluas untuk dieksekusi pada sebuah dyno.

Compiler slug dipanggil oleh hook pre-accept Git – skrip khusus yang dipanggil saat kode didorong ke platform Heroku oleh pengembang. Kompiler siput menggunakan langkah-langkah berikut selama menjalankannya :



Berikut ini adalah deskripsi langkah kompilasi siput :

1. Buat checkout HEAD dari cabang utama.
2. Hapus file yang tidak digunakan, termasuk direktori .git, apa pun di direktori log dan tmp, dan apa pun yang ditentukan dalam file .slugignore tingkat atas.
3. Unduh, bangun, dan instal dependensi lokal seperti yang ditentukan dalam file build Anda dengan alat manajemen dependensi yang didukung oleh bahasa (misalnya, Bundler untuk Ruby).
4. Kemas arsip slug terakhir.

Mengoptimalkan siput

Jika repositori Anda berisi file yang tidak perlu untuk menjalankan aplikasi Anda, Anda harus mempertimbangkan membenarkan atau mengoptimalkan siput Anda dengan melakukan beberapa optimasi. Anda dapat mencegah file dalam repositori Anda agar tidak disertakan dalam slug dengan memasukkan nama file yang akan dikecualikan dalam file .slugignore. File-file yang dikecualikan ini biasanya file yang berisi tes unit, dokumen (PDF), atau data uji proyek yang tentu tidak diperlukan dalam aplikasi yang digunakan.

Berikut ini adalah contoh konten file .slugignore :

```
*.pdf
/test
/spec
```

File .slugignore menyebabkan file dihapus setelah Anda mendorong kode ke Heroku dan sebelum buildpack berjalan. Ini memungkinkan Anda mencegah file besar agar tidak dimasukkan dalam slug terakhir. Anda selanjutnya dapat mengurangi jumlah file yang tidak perlu (sementara atau file log) dengan memastikan bahwa mereka tidak dilacak oleh Git, dalam hal ini mereka juga tidak akan digunakan untuk Heroku.

Batas ukuran

Ukuran siput ditampilkan setelah kompilasi berhasil. Anda juga bisa mengetikkan info heroku: aplikasi untuk menemukan ukuran siput begitu siput dibuat. Ukuran siput maksimum adalah 200 MB; aplikasi idealnya jauh lebih kecil dari ini untuk membantu penyebaran dan eksekusi yang lebih cepat. Siput yang lebih kecil dapat ditransfer melintasi dyno manifold lebih cepat sehingga memungkinkan penskalaan yang relatif instan.

Slug size	Remarks
<15 MB	Small size, fastest to deploy and run
<50 MB	Average size, moderate deployment time
>90MB	Large size, best avoided, slower deploys and execution

Jika ukuran aplikasi melampaui ukuran terbesar, disarankan ukuran siput dioptimalkan dengan memindahkan file yang lebih besar seperti PDF atau mp3 atau video ke penyimpanan aset, katakan CDN, dan hapus file yang tidak perlu dan dependensi dari aplikasi.

Untuk memeriksa ukuran siput, Anda dapat menggunakan aplikasi: perintah info sebagai berikut:

```
$ heroku apps:info
=== herokuclouddevapp
Addons:      heroku-postgresql:dev
Git URL:     git@heroku.com:herokuclouddevapp.git
Owner Email:XXXXXX@yyyy.com
Repo Size:   6M
Slug Size:   8M
Stack:       cedar
Web URL:     http://herokuclouddevapp.herokuapp.com/
```

Ukuran repositori (repo) tidak memiliki batas yang telah ditentukan tetapi repositori yang sangat besar tidak disarankan karena menyebabkan batas waktu transfer dan dorongan kode yang lambat. Cache build juga disimpan di dalam repositori, karenanya repositori mungkin lebih besar dari jarak jauh secara lokal.

Biasanya, memeriksa file biner ke dalam repositori atau log pengembangan aplikasi berukuran besar menyebabkan repositori tumbuh dalam ukuran kadang-kadang ke level yang tidak direkomendasikan. File yang diperiksa secara tidak sengaja dapat dihapus dengan memfilternya dengan utilitas cabang Git filter. Pastikan Anda memaksakan mendorong kode lain kali setelah menjalankan utilitas ini setelah berkoordinasi dengan pengembang lain.

Ringkasan

Bab ini adalah tentang membangun aplikasi web di Heroku. Dengan bantuan contoh sederhana, kami mempelajari langkah-langkah yang terlibat dalam membangun aplikasi web. Kami juga meninjau prinsip-prinsip panduan untuk mengembangkan aplikasi web SaaS pada platform Heroku. Kami melihat API konfigurasi yang disediakan oleh Heroku CLI untuk mengatur dan menghapus variabel lingkungan untuk aplikasi Anda di platform Heroku. Kami juga mempelajari sedikit lebih dalam tentang buildpacks – cara Heroku mengidentifikasi runtime bahasa yang akan digunakan untuk membangun aplikasi Anda. Akhirnya, kami melihat proses kompilasi siput yang membangun kode Anda menjadi sesuatu yang dapat dieksekusi, menghubungkannya dengan perpustakaan yang diperlukan dan aset lainnya.

Sekarang kita memiliki pemahaman yang masuk akal tentang fitur-fitur Heroku, tumpukan platform, dan proses pembuatan, ini adalah saat yang tepat untuk mempelajari cara menjalankan aplikasi di Heroku; pokok bahasan bab selanjutnya.

BAB 4
MENYEBARKAN HEROKU APLIKASI

Pada bab sebelumnya, kami belajar tentang membangun aplikasi Heroku. Secara khusus, kami mengeksplorasi bagaimana proses build bekerja, bagaimana sistem Buildpack yang mendasarinya memungkinkan runtime yang tepat untuk membangun aplikasi Anda, dan akhirnya, bagaimana parameter konfigurasi, juga disebut config vars, memainkan peran penting dalam semantik eksekusi aplikasi Anda.

Sekarang, kita akan mengambil langkah logis berikutnya dan memahami fase penyebaran siklus hidup pengembangan aplikasi Heroku.

Dalam bab ini, antara lain, kita akan melihat hal-hal berikut :

- Memahami persyaratan penyebaran Heroku
- Memperoleh keakraban dengan kontrol versi yang didistribusikan Git yang digunakan secara luas dengan manajemen kode sumber Heroku
- Menyebarkan aplikasi Anda ke Heroku dan mempelajari cara mengoptimalkan siput
- Menguraikan konsep kloning dan forking aplikasi Heroku Anda
- Mengoptimalkan langkah penyebaran aplikasi
- Memperkenalkan dan memahami konsep daerah untuk penempatan
- Melacak perubahan aplikasi dengan kait penempatan
- Mengelola rilis aplikasi Heroku Anda

Penempatan di Heroku

Sementara banyak platform saat ini menawarkan serangkaian alat dan aksesori yang kaya untuk menjalankan aplikasi web dengan mulus, satu area di mana Heroku menonjol adalah model penyebarannya yang sederhana dan efisien. Model penyebaran Heroku sederhana fokuslah menulis aplikasi Anda dan serahkan sisanya pada Heroku.

Penempatan Heroku terdiri dari yang berikut ini :

- Mendapatkan akun Heroku
- Menginstal toolbelt clientLogging ke akun Heroku menggunakan klien Heroku
- Menulis aplikasi Anda
- Mendorong aplikasi Anda ke Heroku menggunakan kontrol versi Git

Kode yang terdorong akan dibangun menjadi slug atau self-contained yang dapat dieksekusi oleh compiler slug dan dieksekusi sebagai dyno (proses Heroku) dalam manifold dyno (lingkungan eksekusi Heroku).

Tidak peduli apa bahasa pemrogramannya, penyebaran aplikasi Heroku mengikuti urutan kejadian yang hampir sama, dari hanya menjadi sepotong kode hingga menjadi aplikasi web produksi yang lengkap. Fakta bahwa seorang pengembang hanya perlu membuat aplikasi dan Heroku mengurus peluncuran aplikasi dari sana adalah aspek yang sangat kuat dari model penyebaran Heroku.

Dalam bab ini, demi kelengkapan dan konsistensi, kami akan membahas masing-masing elemen sebelumnya dan memfokuskan lebih dalam pada elemen yang belum pernah dibahas sebelumnya.

Mendapatkan akun Heroku

Langkah pertama untuk mulai menggunakan Heroku adalah mendaftar ke akun Heroku. Pergi ke http://www.heroku.com dan daftar untuk akun Heroku.

Anda dapat memilih salah satu dari yang berikut :

- Gratis 750 jam dyno per aplikasi, per penggunaan bulanan
- Beli jam dino yang diperlukan sesuai permintaan aplikasi Anda

Saat berlangganan berhasil, Anda dapat meninjau detail akun Anda di dasbor Heroku.

Menginstal kit klien toolbelt

Selanjutnya, unduh kit peralatan Heroku dari situs web Heroku https://toolbelt.heroku.com.

Toolbelt kit berisi komponen-komponen berikut :

- Klien Heroku: Ini adalah alat baris perintah yang membantu membuat dan mengelola aplikasi
- Mandor: Ini adalah utilitas yang memberi Anda fleksibilitas menjalankan aplikasi secara lokal, terutama pada saat Anda ingin memecahkan masalah sistem kontrol revisi Git dan program utilitas yang relevan yang membantu Anda mendorong / mengunduh kode Anda ke / dari Heroku

Masuk ke akun Heroku

Anda dapat masuk ke akun Heroku Anda dengan mengatur sesi SSH aman terlebih dahulu.

Menyiapkan SSH

Jika Anda belum menggunakan SSH, Anda harus membuat pasangan kunci publik-pribadi untuk mendorong kode ke Heroku. Pasangan kunci ini digunakan untuk menjaga saluran komunikasi antara pengembang dan Heroku aman.

Membuat kunci publik sesederhana memasukkan perintah berikut :

$ ssh-keygen -t rsa
Menghasilkan pasangan kunci publik / swasta.
Masukkan file untuk menyimpan kunci (/ Pengguna/dianra/.ssh/id_rsa):
Masukkan frasa sandi (kosong tanpa frasa sandi):
Masukkan frasa sandi yang sama lagi:
Identifikasi Anda telah disimpan di / Pengguna/dianra/.ssh/id_rsa.
Kunci publik Anda telah disimpan di / Pengguna/dianra/.ssh/id_rsa.pub.
Sidik jari kunci adalah:
a1: 84: 0a: 08: 72: 90: c6: d9: d5: d6: e3: 04: d5: 6c: 3e dianra@edha.pc

Dalam perintah ini, kami menggunakan metode enkripsi **RSA** untuk mengamankan kredensial kami. Dengan metode RSA, Anda dapat menandatangani / memverifikasi dan mengenkripsi / mendekripsi kredensial Anda untuk tujuan otentikasi. Opsi keamanan lain yang dapat diandalkan tetapi lebih lambat yang dapat Anda gunakan adalah algoritma DSA yang memungkinkan Anda untuk menandatangani dan memverifikasi kredensial. Dikatakan bahwa lebih sulit untuk memecahkan kunci DSA daripada kunci RSA untuk kunci dengan panjang yang sama.

Menekan Enter pada kedua prompt membuat kunci tanpa kata sandi aman. Selama Anda menyimpan rahasia ~ / .ssh / id_rsa, kunci Anda akan aman bahkan tanpa kata sandi.

Anda dapat membaca detail lebih lanjut tentang alat ssh-keygen di http: //en.wikipedia. org / wiki / Ssh-keygen.

Pertama kali Anda menjalankan perintah Heroku, Anda akan diminta untuk kredensial Anda. Kunci publik Anda kemudian akan diunggah secara otomatis ke Heroku, memungkinkan Anda untuk menyebarkan kode ke semua aplikasi Anda.

Kunci publik Anda harus tersedia di Heroku agar Anda dapat terhubung.

Anda dapat menghapus kunci lama, yang mungkin tidak digunakan untuk sementara waktu, menggunakan perintah berikut :

$ heroku keys:remove dianra@edha.pc
Removing dianra@edha.pc SSH key... done

Nama kuncinya adalah bit pengguna @workstation yang muncul di akhir baris kunci dalam file kunci publik Anda.

Anda dapat melihat daftar semua kunci, termasuk kemudian nama, menggunakan perintah berikut :

$ heroku keys
=== joe@example.com Keys
ssh-dss AAAAB8NzaC...DVj3R4Ww== joe@edha.pc

Opsi baris perintah panjang untuk perintah kunci Heroku menampilkan bentuk detail kunci yang lebih panjang.

Setelah selesai, Anda dapat mengeluarkan perintah pada Heroku CLI :

$ heroku login

Sambungkan ke server Heroku setelah Anda memasukkan kredensial akun Anda (email dan kata sandi). Sekarang, Anda siap untuk mulai mendorong aplikasi Anda ke Heroku.

Menulis aplikasi Anda

Langkah ini benar-benar perlu diperhatikan oleh pengembang. Anda dapat membuat aplikasi dalam bahasa pilihan Anda Ruby on Rails, Node.js, Clojure, Java, atau bahasa lain yang didukung lalu dorong ke Heroku.

Mendorong aplikasi Anda ke Heroku

Langkah ini adalah jantung penyebaran sejauh menyangkut aplikasi Heroku. Menyebarkan aplikasi Heroku sesederhana dan tidak rumit seperti mengeluarkan perintah berikut :

git push heroku master

Dan seperti yang mereka katakan, sisanya adalah sihir!

Dalam bab ini, kita akan mempelajari lebih dalam tentang Git sistem kontrol versi kolaboratif yang digunakan untuk pengembangan aplikasi Heroku. Kami juga akan menyentuh pada topik yang merupakan kunci untuk memahami penyebaran pada Heroku dan fasilitas di platform, topik yang membantu menjadikan Heroku platform penyebaran yang mudah dan efisien.

Kosa kata Git

Heroku menggunakan Git sebagai sistem kontrol revisi kode sumber pilihan untuk aplikasi yang digunakan pada platform Heroku. Git adalah sistem kontrol revisi desentralisasi yang sangat kuat untuk mengelola kode di lingkungan pengembangan

terdistribusi. Git telah berkembang menjadi sistem kontrol revisi pilihan untuk ribuan proyek perangkat lunak terdistribusi di seluruh dunia.

Untuk bekerja dengan Heroku, seseorang harus sedikit terbiasa dengan berbagai perintah Git. nKarena itu, kami akan membahas beberapa konsep dan perintah Git untuk membantu Anda bekerja dengan Heroku secara efisien.

Memulai dengan Git

Ada dua skenario yang relevan sehubungan dengan pelacakan proyek dengan Git.
Mereka adalah sebagai berikut :

- Melacak proyek baru
- Menggunakan proyek Git yang ada

File yang belum ditambahkan ke Git memiliki status tidak terlacak, karena repositori tidak mengetahui keberadaannya dan tidak memiliki referensi untuk itu.

File yang sudah ada di repositori memiliki status yang dilacak dan berada di snapshot terakhir repositori Anda. Mereka dapat dibalik ke status sebelumnya, diubah ke status baru, atau dipentaskan dan dikomit ke repositori Git.

Melacak proyek baru

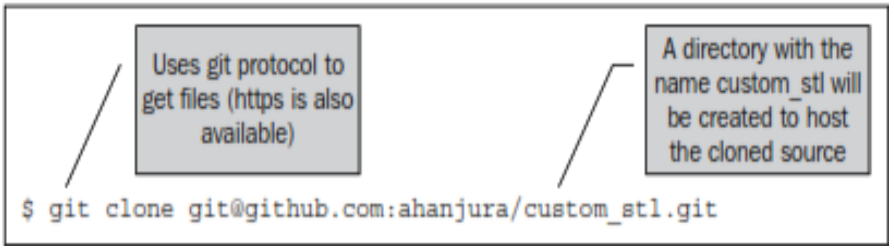
Saat melacak proyek baru dengan Git, Anda perlu menambahkan direktori tempat sumber Anda berada di repositori Git. Ini disebut menginisialisasi repositori proyek Git Anda. Anda harus pergi ke direktori masing-masing pada sistem file tempat kode Anda berada dan ketik perintah berikut :

$ git init

Ini menciptakan direktori bernama .git pada sistem file lokal Anda yang berisi struktur repositori git tulang kosong di dalamnya dengan semua file repositori yang diperlukan. Untuk mulai melacak kode Anda dengan Git, Anda perlu menambahkan kode ke repositori Git menggunakan perintah git add seperti yang ditunjukkan pada gambar berikut:

Menggunakan proyek Git yang ada

Anda dapat menggunakan proyek Git yang ada dan melakukan perubahan, kemudian melacak perubahan ini dengan menggunakan perintah Git. Untuk menggunakan proyek Git yang ada, Anda perlu mengkloning atau menyalin proyek Git ke direktori pada sistem file lokal dan kemudian bekerja secara lokal. Setelah selesai, Anda dapat mengkomit perubahan Anda ke repositori Git, dan Git memastikan bahwa perubahan Anda disimpan dalam repositori secara konsisten. Untuk mengkloning repositori Git, Anda perlu menggunakan perintah git clone dengan cara berikut :



Untuk menggunakan nama direktori khusus untuk memeriksa sumbernya, Anda dapat menambahkan nama direktori target di akhir perintah di tangkapan layar sebelumnya, yang dipisahkan oleh karakter spasi, dan jalankan. Sumber akan dikloning di direktori yang Anda tentukan dalam perintah.

Satu hal yang patut dicatat adalah bahwa klien Git mendukung protokol yang berbeda untuk berinteraksi dengan repositori. Repositori Git dapat menggunakan format http:: atau user @ server: /path.git, yang pada gilirannya menggunakan protokol transfer SSH.

Siklus hidup artefak di Git

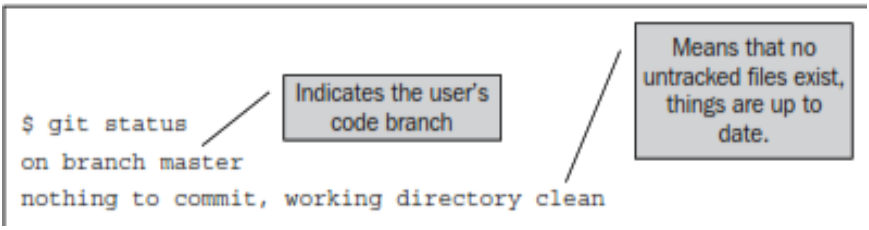
File sumber mengikuti alur kerja yang khas saat beralih dari artefak yang tidak dilacak (ketika tidak ada yang ditambahkan ke Git) ke item yang dilacak dan dikomit (ketika file berkomitmen menggunakan git commit) dalam sistem kontrol versi Git.
Alur kerja ini dapat diringkas sebagai berikut :

1. Pengembang pertama mulai dengan file yang tidak diketahui oleh Git dan karenanya tidak terlacak.
2. Pengembang menambahkan file ke Git menggunakan perintah add git. Sekarang file dilacak tetapi tidak dimodifikasi.
3. Pengembang mengedit file untuk membuat perubahan yang diperlukan dalam aplikasi. File mencapai kondisi yang dimodifikasi.
4. Anda menyiapkan file untuk dikomit dengan mementaskannya secara opsional. File siap untuk dikomit sekarang.
5. Pengembang melakukan file ke Git menggunakan perintah git commit.

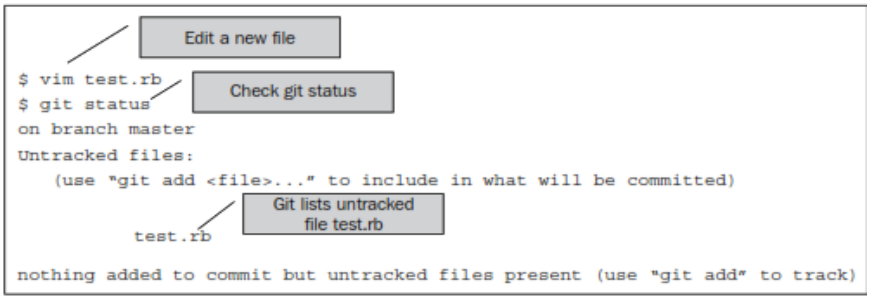
Anda selanjutnya dapat menghapus komit (membatalkan perubahan) atau menghapus artefak itu sendiri tergantung pada kebutuhan spesifik. Anda mungkin perlu mengedit file berkali-kali sebelum mengambil bentuk akhirnya, karenanya bagian dari alur kerja ini dapat diulang berkali-kali.

Melacak file dalam proyek Git

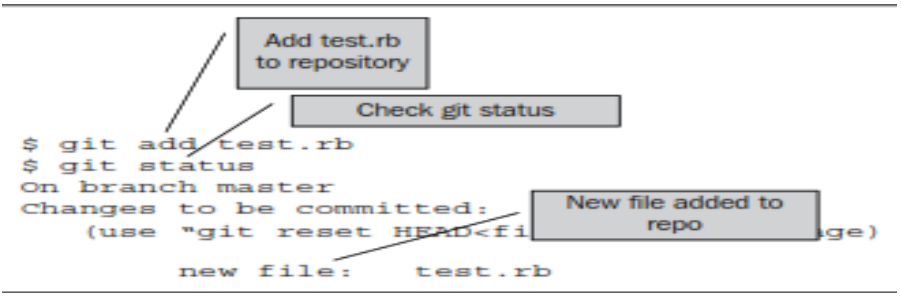
Cukup mudah untuk melacak status file sumber Anda. Gunakan perintah git status untuk mendapatkan status repositori Anda saat ini dengan cara beriku :



Jika Anda mengedit file test.rb baru dan mencoba perintah status git, Git akan melaporkan file yang tidak dilacak sebagai berikut :



Mari kita tambahkan file test.rb dan verifikasi bahwa itu dilacak sekarang. Jalankan perintah git status sebagai berikut :



Ketika Anda tidak perlu Git untuk melacak file Anda

Anda dapat mengabaikan file yang Anda tidak ingin Git menambahkan secara otomatis ke repositori atau yang muncul sebagai dilacak, dengan menggunakan file .gitignore. File .gitignore mencantumkan semua file (diwakili oleh nama file atau pola nama pencocokan ekspresi reguler) yang ingin Anda tambahkan ke daftar yang dikecualikan. File seperti file log atau file yang dihasilkan selama proses build termasuk dalam kategori ini.

File .gitignore mengikuti konvensi tertentu sehubungan dengan ditafsirkan oleh sistem Git. Jika berisi baris kosong atau yang dimulai dengan karakter hash (#), baris tersebut diabaikan.

Anda dapat menggunakan pola standar global dan pola akhir dengan garis miring (/) untuk menentukan direktori atau meniadakan pola dengan menggunakan operator negasi (!) Sebelum pola.

Contoh file .gitignore terlihat seperti berikut :

```
# ignore all files in the temp/directory
temp/

# but do track setup.xml, even through you're ignoring .xml files
!setup.xml

# ignore the root SYSTEM file but not subdir/SYSTEM
/SYSTEM

# ignore man/ls.txt, but not man/server/ls.txt
man/*.txt

# ignore all .txt files in the win/directory
win/**/*.txt

# no .xml files
*.xml
```

Perintah git diff - mengetahui apa yang berubah

Ketik git diff dengan cara berikut tanpa argumen lain untuk memeriksa apa yang telah Anda ubah tetapi belum dipentaskan :

```
$ git diff
diff --git a/test.rb b/test.rb
index abcdef0..ef65585 100644
--- a/test.rb
+++ b/test.rb
@@ -27,6 +26,10 @@ def main
             @commit.parents[0].parents[0].parents[0]
           end

+          foo_run(x, 'commits 1') do
+            git.commits.size
+          end
+
           foo_run(x, 'commits 2') do
             log = glt.commits('master', 7)
             log.size
```

Perintah git diff menunjukkan garis di mana perubahan dipengaruhi (tanda + di awal baris).

Melakukan perubahan Anda

File baru atau yang diubah yang tidak ditambahkan lagi menggunakan git add karena mereka terakhir diubah tidak akan masuk ke dalam komit. Mereka akan terus ada sebagai file yang dimodifikasi pada disk Anda. Setelah semuanya dipentaskan, kami siap untuk melakukan perubahan.

Untuk melakukan perubahan, gunakan perintah git commit dari direktori sumber Anda dengan cara berikut :

$ git commit –m "this is a sample commit"

Perintah ini akan mengkomit isi direktori sumber Anda yang telah Anda ubah sejak checkout terakhir ke git.

Menghapus file

Untuk menghapus file dari Git, Anda harus menghapusnya dari daftar file yang dilacak dan kemudian melakukan perubahan menggunakan perintah git commit. Gunakan perintah git rm untuk menghapus file dari daftar yang dilacak dan juga dari direktori yang berfungsi, sehingga kita tidak melihat file sebagai file yang tidak terlacak saat menjalankan perintah status git nanti.

Jika Anda menghapus file secara fisik dari sistem file, menjalankan perintah status git akan menunjukkan file itu sebagai perubahan yang tidak dilakukan untuk komit.

Anda perlu menjalankan perintah git rm untuk menampilkan file untuk dihapus:

```
$ git rm test.rb
rm 'test.rb'
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    test.rb
```

Jika Anda komit sekarang, file tersebut akan hilang untuk selamanya.

Memindahkan file

Git juga menyediakan fungsi kenyamanan git mv untuk mengganti nama file, yaitu sebagai berikut :

$ git mv src dest

Perintah ini adalah jalan pintas untuk mengubah nama file di luar Git, menghapus file lama dari repositori, dan menambahkan

yang diubah namanya ke Git. Ini lebih merupakan fungsi utilitas. Satu hal yang menarik tentang perpindahan file adalah bahwa Git tidak menyimpan metadata apa pun tentang perpindahan file untuk memberi tahu bahwa file tersebut diubah namanya. Namun, jika Anda memeriksa status setelah mengganti nama file, Git memiliki mekanisme setelah kejadian untuk mengidentifikasi bahwa file tersebut diubah namanya.

Melihat riwayat komit

Salah satu pertemuan paling umum yang dimiliki pengembang dengan Git adalah untuk memeriksa riwayat kode yang dikomit atau modifikasi pada repositori selama periode waktu tertentu. Pengembang mungkin ingin tahu bagaimana dan kapan artefak berubah dari penambahan pertama ke repositori. Perintah git log berguna dalam situasi seperti itu. Log git memberikan fleksibilitas besar dalam menampilkan hanya sebagian daftar atau penjelasan terperinci tentang perubahan artefak berdasarkan opsi baris perintah yang diteruskan ke sana.

Anda dapat menggunakan perintah git log untuk meninjau riwayat commit file yang dimaksud sebagai berikut :

```
$ git log
commit aa82a6dff817ea66f44342007202620a23763242
Author: Anubhav Hanjura <ahanjura@abcdef.com>
Date:   Mon Mar 24 11:52:10 2008 -0700

    changed the version number

commit e77bef06e7f659402fe7567ebf99ed00de2209e6
Author: Anubhav Hanjura <ahanjura@abcdef.com>
Date:   Fri Mar 7 11:27:13 2008 -0700

    first commit
```

Membatalkan perubahan

Untuk mengembalikan perubahan file Anda ke versi yang dikomit terakhir, Anda dapat menggunakan variasi ini dari perintah git checkout sebagai berikut :

$ git checkout -- <filename>

Perintah ini mengembalikan nama file ke status komitmen terakhir dan membatalkan semua perubahan yang telah Anda lakukan untuk sementara.

Anda dapat menggunakan bantuan Git

Untuk mendapatkan bantuan pada perintah Git, Anda dapat menggunakan salah satu dari perintah berikut :

- Pilihan 1 :
  $ git bantuan <verb>
- Pilihan 2 :
  $ git <verb> --help
- Pilihan 3 :
  $ man git- <verb>

Parameter <verb> adalah perintah khusus yang Anda perlukan bantuan.

Repositori lokal

Sebelum Anda bisa mendorong kode Anda ke Heroku, Anda harus membuat repositori Git lokal dan memeriksa file-file Anda di sana. Repositori ini berfungsi sebagai repositori kode sumber lokal. Kode ini akan didorong ke Heroku untuk dibuat dan dijalankan oleh aplikasi Anda nanti.

Repositori jarak jauh

Perintah yang kami lihat sejauh ini adalah perintah umum untuk melacak kode sumber Anda di repositori Git Anda. Namun, dalam konteks pengembangan aplikasi cloud Heroku, kita harus berurusan dengan repositori kode yang di-host di platform Heroku untuk pembuatan dan penyebaran. Oleh karena itu, penting untuk memahami konsep-konsep seputar repositori jarak jauh.

Repositori jarak jauh (atau remote) adalah versi proyek Anda yang dihosting pada mesin yang berbeda dari mesin lokal Anda, biasanya, pada sistem terdistribusi yang dapat diakses melalui serangkaian fungsi untuk mengambil dan memperbarui artefak sumber.

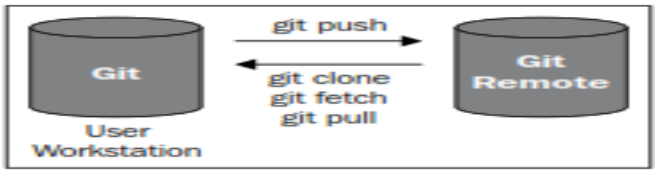
Untuk melihat repositori jarak jauh proyek Anda, Anda dapat menggunakan perintah git remote sebagai berikut :

```
$ git remote -v
origin git://github.com/ahaniura/custom_stl.git (fetch)
origin git://github.com/ahanjura/custom_stl.git (push)
```

Membuat remote Heroku

Setelah repositori Git lokal dibuat, Anda perlu membuat remote Heroku untuk menjaga kode sumber didorong ke lingkungan Heroku (menggunakan Git push). Aplikasi Heroku mengharapkan struktur direktori aplikasi pada akar repositori. Gagal memiliki file aplikasi di direktori yang benar akan menyebabkan kegagalan dalam menjalankan aplikasi.

Gambar berikut ini menunjukkan seperangkat perintah khas yang dikeluarkan oleh pengguna Git dan arah aliran informasi yang sesuai :



Perintah heroku create membuat aplikasi baru di Heroku selain git remote yang digunakan untuk menerima file sumber aplikasi Anda. Perintah git remote hanyalah referensi ke repositori kode sumber jarak jauh. Secara default, nama remote yang dibuat adalah heroku. Dimungkinkan untuk mengubah nama default; namun, ini harus dilakukan menggunakan parameter baris perintah dengan cara berikut :

$ heroku create
Creating gentle-mesa-5445... done, stack is cedar
http://gentle-mesa-5445.herokuapp.com/ |
git@heroku.com:gentle-mesa-5445.
git
Git remote heroku added

Untuk memverifikasi jarak jauh di konfigurasi Git Anda, ketikkan perintah berikut :

$ git remote -v
heroku git@heroku.com:gentle-mesa-5445.git (fetch)
heroku git@heroku.com:gentle-mesa-5445.git (push)

Untuk mengaitkan repo Git dengan aplikasi yang ada, ketik perintah berikut :

$ heroku git:remote -a gentle-mesa-5445
Git remote heroku added.

Untuk memberi nama remote sebagai nama khusus alih-alih Heroku, seperti yang dilakukan sebelumnya, gunakan opsi berikut :

$ heroku git:remote -a gentle-mesa-5445 –r myremote
Git remote myremote added.

Mengganti nama aplikasi

Dimungkinkan untuk mengganti nama aplikasi dengan menggunakan perintah heroku: rename. Aplikasi menjadi dapat diakses dengan nama baru dengan segera, dan nama lama tidak boleh digunakan lagi untuk mengatasinya.

Misalnya, untuk mengganti nama aplikasi sebagai foozball.herokuapp.com, Anda dapat mengetik perintah berikut :

$ heroku apps:rename foozball
Renaming gentle-mesa-5445 to foozball... done
http://foozball.herokuapp.com/ |
git@heroku.com:foozball.git
Git remote heroku updated

Perintah ini berhasil karena kami mengetik di dalam folder aplikasi. Jika Anda ingin mengubah nama aplikasi dari luar checkout Git atau folder aplikasi, Anda dapat mengetik perintah berikut :

$ heroku apps:rename newname --app oldname
http://newname.heroku.com//git@heroku.com:newname.git

Remote Git apa pun yang menunjuk ke nama lama perlu diubah atau diperbarui secara manual untuk mencerminkan perubahan nama aplikasi sebagai berikut :

$ git remote rm heroku

$ heroku git:remote -a foozball

Mengirim kode ke Heroku

Setelah Anda membuat remote, remote Heroku dalam hal ini, Anda bisa mendorong kode Anda ke Heroku dengan mengetik perintah berikut :

$ git push heroku master

updating 'refs/heads/master'

Perintah push git secara efektif mendorong kode ke lingkungan remote Heroku. Dalam hal ini, kami mendorong kode ke master jarak jauh saat kami mendorong kode untuk pertama kalinya dari lingkungan lokal kami.

Remote push git secara efektif mendorong kode ke Lingkungan remote Heroku. Dalam hal ini, kami mendorong kode ke master jarak jauh saat kami mendorong kode untuk pertama kalinya dari lingkungan lokal kami.

$ git push heroku somebranch:master

Atau, Anda dapat menggabungkan kode dengan master dan mendorong kode Anda ke Heroku.

Jika Anda menggunakan asal nama jarak jauh khusus, maka perintah push Anda tidak harus secara eksplisit menentukan nama jarak jauh, dan Anda bisa mengetik berikut ini :

Jika Anda menggunakan asal nama jarak jauh khusus, maka perintah push Anda tidak harus secara eksplisit menentukan nama jarak jauh, dan Anda bisa mengetik berikut ini :

$ git push

Di sini, asal digunakan sebagai nama jarak jauh default.

Mengoptimalkan ukuran siput

Anda mungkin tidak ingin mendorong semuanya dalam aplikasi Anda ke Heroku karena berbagai alasan. Sebagai contoh, Anda tidak benar-benar perlu mengirim file log atau aset statis ke lingkungan Heroku jarak jauh Anda karena tidak berguna untuk menyimpannya di remote karena mereka akan menempati ruang disk tanpa alasan yang sah.

Untuk alasan ini, Git memberi Anda cara untuk melewati file atau direktori yang tidak Anda inginkan atau yang perlu Anda dorong ke lingkungan jauh. Kemampuan ini tersedia melalui file .gitignore, yang berisi daftar file yang tidak ingin Anda dorong. Daftar ini bisa berupa nama file literal atau nama file wildcard. Git mengabaikan file-file ini ketika mendorong kode ke lingkungan Heroku jarak jauh. Ini memiliki manfaat tambahan menjaga ukuran slug lebih kecil dan mempercepat waktu boot up untuk dino.

Saat mendorong kode Anda ke Heroku, pastikan file yang tidak perlu tidak didorong, misalnya, log atau direktori kerja sementara. Sebagai pengguna, Anda dapat mengkonfigurasi Git untuk mengabaikan file-file tertentu, dan jika Anda suka, hapus dari repositori Anda. Konfigurasi ini dapat dilakukan dalam file .gitignore. Beberapa kerangka kerja yang tersedia saat ini berisi file .gitignore secara default, yang mencantumkan pola file atau file literal yang ingin mereka abaikan saat mendorong kode ke Heroku. Anda juga dapat mengedit file .gitignore di folder aplikasi untuk membuat daftar pola file atau nama file tambahan untuk menghindari dorongan.

Misalnya, untuk mengabaikan direktori temp sepenuhnya, Anda dapat menambahkan berikut ini di file .gitignore :

```
$ echo temp >> .gitignore
$ git add .gitignore
$ git commit -m "ignored temp dir completely"
```

Untuk mengabaikan file dengan ekstensi log, yaitu file * .log, Anda dapat membuat file .gitignore dan menambahkan instruksi

untuk mengabaikannya sambil mendorong kode ke Heroku sebagai berikut :

$ mkdir log
$ echo '*.log' > log/.gitignore

File .gitignore akan berisi perincian berikut sekarang :

***.log**

Untuk menambahkan direktori log ke repositori, gunakan perintah berikut :

$ git add log

Untuk menghapus file dari lingkungan Anda (misalnya, untuk menghapus semuanya dari folder log), Anda bisa mengetik perintah berikut :

$ git rm -r -f log
rm 'log/dev.log'
rm 'log/prod.log'
rm 'log/svr.log'
rm 'log/test.log'

Mengkloning aplikasi Heroku yang ada

Sebagai seorang pengembang, Anda mungkin ingin membuat salinan aplikasi Heroku Anda dan mencoba sesuatu yang radikal dengannya sambil menjaga aplikasi yang ada tetap berfungsi seperti apa adanya. Ini dapat dicapai dengan mengkloning aplikasi Anda menggunakan fasilitas Git.

Gunakan heroku git: clone untuk membuat tiruan dari aplikasi Anda sebagai berikut :

$ heroku git:clone -a <NEW APPNAME>

Ini akan membuat direktori baru bernama setelah aplikasi Anda dan secara otomatis menambahkan remote Heroku Git untuk memungkinkan perubahan di masa depan.

Forking aplikasi

Ketika memelihara aplikasi yang paling kompleks, itu adalah praktik umum untuk menciptakan beberapa lingkungan untuk mendukung pengembangan berbagai fitur secara independen, sehingga memaralelkan seluruh proses dan menciptakan efisiensi. Sangat masuk akal untuk menggunakan lingkungan produksi yang stabil sebagai dasar untuk berbagai lingkungan ini, apa yang digunakan untuk tahap atau menguji aplikasi.

Menggunakan perintah garpu heroku memungkinkan Anda untuk melakukan hal berikut :

Menggunakan perintah garpu heroku memungkinkan Anda untuk melakukan hal berikut :

1. Salin aplikasi yang ada.
2. Reprovision menggunakan add-on (dengan paket harga yang sama).
3. Salin variabel konfigurasi.
4. Salin data apa pun dari database Heroku Postgres.

Berikut ini adalah hasil setelah berhasil menjalankan perintah :

- Pengguna Heroku yang menjalankan perintah adalah pemilik aplikasi baru.
- Jika aplikasi menggunakan komponen apa pun yang tidak gratis, pemilik aplikasi bertanggung jawab atas biaya yang dikeluarkan. Oleh karena itu, sangat masuk akal untuk memverifikasi aplikasi sumber untuk setiap komponen yang dapat diisi sebelum melakukan tugas.
- Data Heroku Postgres secara otomatis disalin ke aplikasi baru.
- Add-on hanyalah reprovisioned, dan semua data yang diperlukan ekspor atau impor untuk layanan ini perlu dilakukan secara manual. Jika add-on tidak dapat ditentukan karena paket asli tidak ada lagi, perbarui paket pada aplikasi sumber dan coba garpu lagi. Jika Anda sudah menjalankan garpu heroku, Anda perlu menghancurkan aplikasi target sebelum mencoba kembali dengan cara berikut :

  $ heroku destroy -a toapp

  Misalnya, buat garpu menggunakan perintah berikut :

$ heroku fork -a fromapp toapp
Creating fork toapp... done
Copying slug... done
Adding pgbackups:plus... done
Adding heroku-postgresql:dev... done
Creating database backup from sourcapp... .. done
Restoring database backup to toapp... .. done
Copying config vars... done
Fork complete, view it at http://toapp.herokuapp.com/

Seperti kebanyakan perintah, Anda memerlukan utilitas klien Heroku toolbelt yang diinstal untuk dapat menjalankan perintah garpu heroku.

Efek samping dari forking aplikasi

Keputusan untuk mem-fork aplikasi versus kloning itu sangat penting dalam menentukan kinerja aplikasi Anda selama masa pakainya. Oleh karena itu, penting untuk memahami efek samping dari forking aplikasi baru sehingga Anda menggunakan kapabilitas forking secara bijaksana ketika Anda harus. Berikut ini adalah efek samping dari forking aplikasi :

- Aplikasi bercabang sama seperti aplikasi baru karena mereka memiliki formasi dyno default yang terdiri dari satu dino web dan tidak ada pekerja atau dino lainnya. Anda dapat mengatur dinamika aplikasi bercabang Anda untuk memenuhi kebutuhan Anda.
- Proses forking mengecam titik akhir SSL pada aplikasi baru tetapi tidak menambahkan sertifikat apa pun secara otomatis. Jika aplikasi Anda menggunakan domain khusus dengan SSL, Anda perlu menambahkan sertifikat baru ke instance endpoint SSL pada aplikasi target sebagai berikut:

  $ heroku certs:add server.crt server.key -a toapp
  Resolving trust chain... done
  Adding SSL Endpoint to toapp... done
  example now served by hawaii-9876.herokussl.com

Add a new DNS CNAME record utilizing this new endpoint URL to
serve requests via HTTPS.

- Tidak ada domain khusus yang disalin sebagai bagian dari proses forking karena domain khusus hanya dapat dimiliki oleh satu aplikasi. Jika Anda ingin menggunakan domain khusus di lingkungan baru Anda, Anda harus menambahkannya secara manual dan membuat tambahan DNS yang diperlukan.
- Anda perlu mentransfer jadwal pekerjaan secara manual dari satu aplikasi ke aplikasi lainnya. Buka dasbor penjadwal untuk fromapp dan toapp, bandingkan, dan salin pekerjaan secara manual dengan cara berikut :

  $ heroku addons:open scheduler -a fromapp
  $ heroku addons:open scheduler -a toapp
- Ketika Anda bercabang aplikasi, itu tidak secara otomatis membuat remote Git baru di namespace proyek Anda saat ini. Pertama, buat remote secara manual dan kemudian gunakan aplikasi yang baru bercabang seperti yang ditunjukkan pada perintah berikut :

  $ heroku info -a toapp
  === toapp
  ...
  Git URL: git@heroku.com:toapp.git

  Tambahkan remote Git bernama forked, yang mewakili URL sebarkan untuk toapp : **$ git remote add forked git@heroku.com:toapp.git**

  Menyebarkan ke lingkungan baru dengan :

  $ git push forked master
- Tidak ada pengguna dari aplikasi sumber ditransfer ke aplikasi bercabang. Anda perlu menambahkan kolaborator secara manual sebagai berikut :

  $ heroku sharing:add someoneelse@johndoe.com -a toapp
- Proses forking menyalin semua database yang ada di sourceapp tetapi tidak mempertahankan hubungan garpu / ikuti di antara

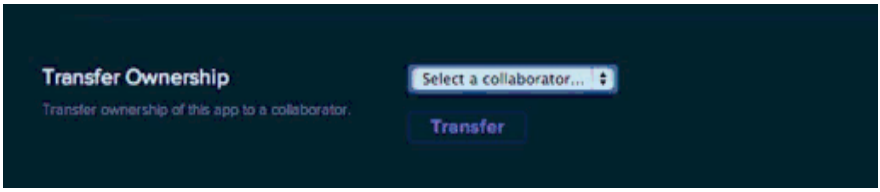
mereka. Hapus sendiri database asing dan buat kembali secara manual garpu atau pengikut.

- Setiap fitur Heroku lab yang diaktifkan pada aplikasi sumber tidak diaktifkan kembali pada aplikasi target.

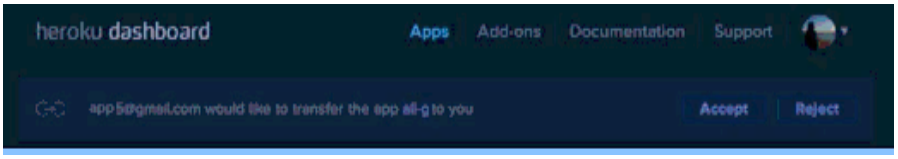
Mentransfer Aplikasi

Anda dapat mentransfer aplikasi antar akun Heroku kapan saja melalui dasbor Heroku.

Pemilik aplikasi saat ini harus memulai permintaan transfer. Untuk memulai transfer, pemilik dapat menavigasi ke halaman pengaturan aplikasi di dasbor. Sepertinya tangkapan layar berikut :



Sebagai pemilik saat ini, Anda dapat memulai permintaan transfer dengan memilih kolaborator untuk mentransfer aplikasi. Pemilik baru dapat menerima transfer dengan menerima permintaan transfer yang tertunda di bagian atas dasbor setelah dimulai :



Pemilik baru memiliki kemampuan untuk menerima atau menolak permintaan transfer apa pun. Selain itu, pemilik asli permintaan dapat membatalkan atau mengubah permintaan transfer sebelum diterima atau ditolak oleh pemilik baru.

Mengoptimalkan penyebaran

Aplikasi Heroku yang paling sederhana dijalankan di setidaknya dua lingkungan yang berbeda lingkungan pengembangan lokal Anda dan lingkungan terpencil Heroku.

Aplikasi produksi yang diberikan dapat membuat banyak pengembang mengerjakannya. Anda dapat meminta beberapa pengembang berinteraksi dengan aplikasi produksi yang sama ketika mereka ingin mendorong perubahan lokal mereka ke sana. Selain itu, pengembang mungkin memiliki cabang kode tambahan untuk menguji aplikasi mereka atau memverifikasi persyaratan nonfungsional seperti kinerja.

Semua hal bekerja dengan baik selama ada konkurensi antara berbagai lingkungan yang Anda gunakan, yaitu, konfigurasi yang sama dan kode sumber umum menghasilkan perilaku fungsional yang sama. Namun, di dunia nyata, segala sesuatunya menjadi rumit ketika ukuran proyek meningkat dan jumlah pengembang yang mengerjakannya tumbuh dengan cepat. Selain itu, berbagai hal dapat menjadi rumit jika aplikasi sedang dikembangkan secara lokal pada sistem operasi yang berbeda atau menggunakan pengaturan lingkungan khusus. Situasi seperti itu dapat menyebabkan hasil yang aneh ketika Anda mencoba menggunakan aplikasi di Heroku. Untuk mengatasi tantangan seperti itu, kita perlu cara yang efektif untuk mengelola penyebaran.

Lingkungan pengembangan, yang biasanya memiliki fitur-fitur baru di dalamnya, mungkin berisi kode yang rusak atau mungkin mengarah ke database yang berbeda dari produksi (misalnya, SQLite3 versus. Postgres). Selama periode waktu tertentu, berbagai lingkungan mulai memimpin atau menunda produksi karena aliran perubahan paralel yang terjadi dan tidak ada sistem untuk menjaga mereka tetap sinkron. Skenario seperti itu, jika tidak dikelola dengan baik, adalah resep untuk inkonsistensi dan kegagalan aplikasi Anda.

Jadi, bagaimana kita memastikan bahwa kita tidak mendorong kode buggy ke produksi atau tidak menyebarkan sesuatu yang merusak produksi dan menyebabkan banyak mulas pada pengguna?

Salah satu cara untuk menangani masalah ini adalah dengan menggunakan konsep pementasan aplikasi Anda sebelum memindahkannya ke produksi. Idenya adalah untuk membuat lingkungan pementasan meniru produksi dalam segala hal dan mengarahkan perubahan ke produksi melalui pementasan. Kecuali hal-hal berjalan seperti yang diharapkan dalam pementasan, kami

tidak memindahkan apa pun ke lingkungan produksi yang sudah berfungsi dengan baik.

Menciptakan lingkungan seperti itu cukup mudah di Heroku sebagai berikut :

```
$ heroku create --remote staging
Creating weak-mountain-783.... done
http://weak-mountain-783.heroku.com/ |
git@heroku.com:weak-mountain-783.
git
Git remote staging added
```

Sekarang, Anda dapat memunculkan lingkungan pementasan dengan menggunakan perintah berikut:

```
$ git push staging master
...
$ heroku run rake db:migrate --remote staging
...
$ heroku ps --remote staging
=== web: `bundle exec thin start -p $PORT -e production`
web.1: up for 21s
```

Untuk memberikan variabel konfigurasi khusus untuk pementasan saat menggunakan kerangka kerja **Ruby on Rails** (**RoR**), ketikkan perintah berikut:

```
$ heroku config:add RACK_ENV=staging
RAILS_ENV=staging --remote staging
```

Parameter konfigurasi ini biasanya digunakan oleh aplikasi untuk mencari tahu detail lingkungan. Untuk mengaktifkan aplikasi RoR untuk menggunakan lingkungan pementasan, Anda perlu membuat file config / environment / staging.rb sebelumnya, ditambah dengan pengaturan RAILS_ENV. Anda dapat mem-boot server Anda untuk menggunakan lingkungan pementasan untuk aplikasi Anda.

Karena aplikasi Anda berjalan dan berjalan dalam pementasan, Anda dapat memonitornya untuk setiap perbedaan, dan jika semuanya berjalan dengan baik, Anda sekarang dapat membuat versi produksi yang sama dengan melakukan langkah-langkah berikut. Anda juga perlu mengonfigurasikan add-on apa pun, mengatur vars lingkungan, atau menambahkan kontributor ke kedua aplikasi ini saat mereka berevolusi agar tetap sinkron :

$ heroku create --remote production
Creating fierce-ice-327.... done
http://fierce-ice-327.heroku.com/ | git@heroku.com:fierce-ice-327.git
Git remote production added
$ git push production master
...
$ heroku run rake db:migrate --remote production
...
$ heroku ps --remote production
=== web: `bundle exec thin start -p $PORT -e production`
web.1: up for 16s

Dengan demikian, kami memiliki kode sumber yang sama berjalan ketika dua aplikasi Heroku yang terpisah diatur secara identik.

Untuk mempermudah, Anda dapat menggunakan perintah git config untuk menentukan aplikasi default alih-alih harus mengetikkan nama aplikasi setiap kali Anda menjalankan perintah Git.

Misalnya, untuk mengonfigurasi pementasan sebagai remote default Anda, ketikkan perintah berikut :

$ git config heroku.remote staging

Ini akan mengonfigurasi pementasan sebagai lingkungan default Anda dan menambahkan bagian ke file .git / config proyek yang menunjukkan hal yang sama.

Untuk menjalankan perintah pada aplikasi yang berbeda, cukup gunakan opsi --remote <env name>.

Dalam banyak kasus, mungkin saja Anda mulai dengan aplikasi kecil yang tidak terlalu cerewet dan karenanya dapat digunakan secara langsung untuk produksi. Namun, seiring waktu, aplikasi mungkin mengalami peningkatan dalam kompleksitas dan skala dan membutuhkan pemantauan yang lebih dekat. Anda mungkin ingin lebih berhati-hati sebelum menggunakan perangkat tambahan baru untuk produksi sekarang. Anda bisa mengikuti pendekatan yang sama dengan pementasan lingkungan di sini juga.

Untuk membuat lingkungan pementasan, Anda akan memerlukan parameter konfigurasi dan informasi tambahan untuk lingkungan produksi Anda sehingga Anda dapat mengatur lingkungan pementasan dengan menggunakan detail yang sama.

Anda dapat mengetik perintah berikut di lingkungan produksi Anda untuk menangkap parameter konfigurasi :

$ heroku config
...list of configuration variables

Anda dapat mengetik perintah berikut di lingkungan produksi Anda untuk mengambil informasi tambahan :

$ heroku addons
...list of addons

Terakhir, buat lingkungan pementasan dengan mengetik perintah berikut :

$ heroku create --remote staging --addons <add-on 1>,<add-on 2>,...
$ heroku config:add CFGVAR1=abc CFGVAR2=xyz

Selain itu, Anda harus menyalin basis data produksi Anda ke lingkungan panggung menggunakan alat cadangan yang tersedia seperti add-on PGBackups.

Pilihan suatu daerah

Heroku saat ini sedang ditingkatkan untuk mendukung konsep wilayah penempatan, yaitu, Anda dapat memilih wilayah geografis tempat aplikasi Anda akan digunakan. Fitur ini pada dasarnya memungkinkan Anda untuk mengurangi latensi yang dialami oleh pengguna aplikasi Anda.

Saat ini, Heroku tersedia di dua wilayah geografis: AS dan UE. Jika sebagian besar pengguna Anda berada di Eropa, masuk akal untuk menggunakan aplikasi Anda di Eropa karena akses akan lebih cepat bagi mereka. Semua aplikasi dibuat di wilayah AS secara default.

Cara termudah untuk memeriksa wilayah aplikasi Anda adalah dengan menggunakan perintah info heroku berikut :

$ heroku info
=== warm-current-5432
Git URL: git@heroku.com:warm-current-5432.git
Owner Email: user@test.com
Region: eu
Repo Size: 164M
...

Untuk memverifikasi daftar semua wilayah yang tersedia, ketikkan perintah berikut :

$ heroku regions
=== regions
eu Europe
us United States
Specifying the region

Anda dapat menggunakan flag --region untuk menentukan wilayah saat membuat aplikasi :

$ heroku create --region eu
Creating warm-current-5432... done, region is eu
http://warm-current-5432.herokuapp.com/ |
git@heroku.com:warm-current-5432.git
Git remote heroku added

Ada beberapa aspek dari aplikasi Anda yang perlu dipertimbangkan saat menggunakan aplikasi di wilayah tertentu.

Add-on dengan dukungan wilayah akan disediakan di wilayah yang sama dengan aplikasi. Berikan mereka seperti yang biasa Anda lakukan dengan menggunakan perintah berikut :

$ heroku addons:add <addon_name>

Untuk memverifikasi add-on yang didukung untuk wilayah Anda, ketikkan perintah berikut :

$ heroku addons:list --region=<region_name>

Di sini, nama wilayah bisa kita atau eu.

Anda juga dapat masuk ke addons.heroku.com dan mencari add-on yang didukung untuk wilayah Anda.

Beberapa add-on mungkin tidak memerlukan koneksi latensi rendah ke aplikasi Anda. Oleh karena itu, dapat digunakan di wilayah default jika tidak tersedia di wilayah aplikasi Anda. Jika add-on sensitif terhadap latensi dan tidak tersedia di wilayah yang sama dengan aplikasi Anda, penyediaan tidak akan lengkap dan akan gagal dengan cara berikut :

$ heroku addons:add cloudcounter
Adding cloudcounter on warm-current-5432... failed
! This app is in region eu, cloudcounter:basic is only available in region us.

Aplikasi dikerahkan ke wilayah yang ditentukan pada penciptaan (yang secara default adalah kita). Mudah untuk menggunakan aplikasi Anda. Pekerjaan penjadwal apa pun, atau salah satu dari dino, akan dijalankan di wilayah yang sama di mana aplikasi dikerahkan.

Gunakan dorongan git dengan cara berikut untuk menyebarkan, seperti dalam kasus penyebaran biasa :

$ git push heroku master

Saat Anda mengembangkan aplikasi yang membutuhkan dukungan SSL, Anda harus menyediakan titik akhir SSL untuk aplikasi dan mengunggah sertifikat SSL sebagai berikut :

$ heroku addons:add ssl
Adding ssl on yeppy... done, v1 ($20/mo)
add your certificate with `heroku certs:add PEM KEY`.
Use `heroku addons:docs ssl` to view documentation.
$ heroku certs:add server.crt server.key
Resolving trust chain... done
Adding SSL Endpoint to yeppy... done
yeppy now served by yeppy.herokuapp.com

Jika domain khusus Anda dikonfigurasi dengan benar, tidak ada konfigurasi DNS tambahan yang diperlukan. Semua lalu lintas ke www.yeppy.com sekarang dapat dilayani melalui SSL.

Tidak ada perbedaan antara menambahkan domain khusus ke aplikasi Anda yang berjalan di luar wilayah AS atau ke aplikasi di wilayah AS. Menggunakan wilayah yang berbeda bersifat transparan untuk aplikasi dengan domain khusus dan setiap permintaan pengguna ditangani secara normal.

Pelacakan perubahan aplikasi

Terkadang, Anda ingin melacak perubahan yang dilakukan pada aplikasi Heroku Anda dan tertarik untuk mengetahui kapan versi baru aplikasi tersebut didorong ke Heroku. Biasanya, dalam aplikasi tingkat produksi, Anda mungkin ingin memberi tahu pengguna tertentu tentang rilis baru atau melacak penyebaran baru dengan mencatatnya. Untuk kasus seperti itu, Heroku menyediakan Deploy Hooks.

Deploy Hook memungkinkan Anda menerima pemberitahuan setiap kali versi baru aplikasi didorong ke Heroku. Menyiapkan kait ini cukup sederhana, dan pengembang mendapatkan beragam pilihan untuk jenis Penyebaran Kait yang dapat diatur. Heroku menyediakan Deploy Hooks dalam bentuk e-mail kepada pengguna dan pesan ke ruang obrolan api unggun atau akun basecamp untuk

memberi tahu Anda tentang penyebaran baru. Hook Pekerjaan secara inheren merupakan mekanisme pemberitahuan yang dapat dimanfaatkan untuk berintegrasi dengan aplikasi lain melalui fitur olahpesan bawaan di fitur **Deploy Hooks.**

Menyiapkan Penyebaran Kait

Setiap Deploy Hook adalah pengaya sendiri. Menyiapkan Deploy Hook baru sama seperti menambahkan add-on baru ke Heroku. Mari kita bahas berbagai Deploy Hooks yang didukung di Heroku.

Tempat penampungan

Tujuan dari Penyebaran Kait ini adalah untuk mengirim pesan ke akun Basecamp yang Anda tentukan dalam proyek dan kategori posting tertentu. Letakkan Kunci API Anda di bawah para nama pengguna.

Pengaturan sampel untuk Basecamp adalah sebagai berikut :

```
$ heroku addons:add deployhooks:basecamp \
 --url=http://testaccount.basecamphq.com \
 --username=00000000 \
 --project=weak-mountain-783 \
 --category=deploys \
 --title="{{user}} deployed weak-mountain-783" \
 --body="check it at {{url}}"
Adding deployhooks:basecamp to weak-mountain-783...Done.
```

Api unggun

Tujuan dari Penyebaran Kait ini adalah untuk mengirim pesan otomatis ke akun Api Unggun ketika aplikasi Anda didorong.

Setup sampel untuk Campfire adalah sebagai berikut :

```
$ heroku addons:add deployhooks:campfire \
 --url=testsubdomain \
 --ssl=1 \
 --api_key=0000000 \
 --room=devlounge \
```

--message="{{user}} deployed weak-mountain-783"
Adding deployhooks:campfire to weak-mountain-783...Done.
E-mail

Tujuan dari Penyebaran Kait ini adalah untuk mengirim satu atau lebih email dengan subjek dan badan. Lebih banyak penerima dapat ditentukan dengan menambahkan alamat email yang dipisahkan oleh spasi.
Contoh pengaturan untuk email adalah sebagai berikut :

**$ heroku addons:add deployhooks:email **
**--recipient=john.doe@whereami.com **
**--subject="Weak-mountain-783 Deployed" **
--body="{{user}} deployed app"
Adding deployhooks:email to weak-mountain-783...Done.

HTTP

Tujuan dari Penyebaran Kait ini adalah untuk melakukan posting HTTP ke URL yang disediakan dengan parameter yang dikirim dengan jenis aplikasi mime / x-www-form-urlencoded.
Pengaturan sampel untuk HTTP adalah sebagai berikut :

**$ heroku addons:add deployhooks:http **
--url=http://whereami.org
Adding deployhooks:http to weak-mountain-783...Done.

IRC

Tujuan dari Penyebaran Pengait ini adalah untuk terhubung ke server yang ditentukan dan mengirimkan pesan ke kamar; nick dan kata sandi adalah opsional. Atur port server dengan mengirim port = XXXX.
Pengaturan sampel untuk IRC adalah sebagai berikut :

**$ heroku addons:add deployhooks:irc **
**--server=irc.freenode.net **
**--room=qahouse **
**--nick=testcomm **
**--password=secret **

--message="{{user}} deployed app"
Adding deployhooks:irc to weak-mountain-783...Done.

Setelah kait diatur, git push akan menunjukkan bahwa mereka dijadwalkan untuk berjalan dengan cara berikut:

$ git push heroku master
...
-----> Heroku receiving push
-----> Rails app detected
Compiled slug size is 66K
-----> Launching..... done
-----> Deploy hooks scheduled, check output in your logs
http://weak-mountain-783.heroku.com deployed to Heroku

Keluaran dan kesalahan Hook muncul di logfile aplikasi sebagai berikut:

$ heroku logs
...
2013-03-15T15:04:23-05:00 heroku[deployhooks]: Sent email notification to
xyz@example.com

Anda bisa menggunakan variabel untuk menentukan bagian dari Penyebaran Pengait. Misalnya, variabel {{app}}, ketika digunakan dalam sebuah pesan, akan diganti dengan nama aplikasi ketika Deploy Hook dieksekusi. Banyak variabel lain juga dapat digunakan dalam definisi kait untuk menambahkan informasi spesifik tentang penyebaran ketika itu terjadi.

Beberapa variabel ini adalah pengguna (pengguna yang menggunakan aplikasi), URL (URL aplikasi), head_long atau kepala (komit pengidentifikasi), atau git_log (log komit antara penerapan terakhir dan yang sekarang).

Manajemen rilis

Ketika Anda mendorong kode baru, melakukan perubahan konfigurasi, atau memodifikasi sumber daya, Heroku membuat rilis baru dan memulai ulang aplikasi secara otomatis.

Melalui mekanisme manajemen rilis yang kuat, Heroku memberikan fleksibilitas untuk melakukan hal berikut :

- Daftar sejarah rilis.
- Gunakan rollback untuk kembali ke rilis sebelumnya jika penyebaran atau konfigurasi salah

Memeriksa rilis yang diinstal

Untuk memeriksa riwayat rilis aplikasi Anda, Anda dapat mengetik perintah berikut :

$ heroku releases
=== gentle-mesa-5445 Releases
v2 Enable Logplex dianra@gmail.com 2012/12/03 03:39:37
v1 Initial release dianra@gmail.com 2012/12/03 03:39:36

Output sebelumnya menunjukkan bahwa aplikasi telah memiliki dua rilis sejauh ini, yang terbaru ditampilkan di bagian atas.

Memverifikasi rilis baru

Rilis baru dapat dibuat oleh Heroku ketika Anda mendorong kode baru atau mengubah beberapa parameter konfigurasi. Misalnya, menambahkan parameter konfigurasi baru TEST_MAX_ARGS membuat rilis baru untuk aplikasi. Untuk memeriksa apakah rilis baru memang dibuat, ketikkan perintah berikut :

$ heroku releases
=== gentle-mesa-5445 Releases
v3 Add TEST_MAX_ARGS config dianra@gmail.com 2013/01/27 20:16:21 (~
17s ago)
v2 Enable Logplex dianra@gmail.com 2012/12/03 03:39:37
v1 Initial release dianra@gmail.com 2012/12/03 03:39:36

$ heroku releases:info v3
=== Release v3
By: dianra@gmail.com
Change: Add TEST_MAX_ARGS config
When: 2013/01/27 20:16:21 (~ 57s ago)
=== v3 Config Vars
TEST_MAX_ARGS: 45

Output sebelumnya jelas menunjukkan bahwa rilis baru v3 telah dibuat, dan itu juga menunjukkan log perubahan yang dilakukan. Parameter konfigurasi yang diubah dibatasi dengan jelas di bawah header v3 config vars.

Meluncurkan kembali rilis

Kadang-kadang, Anda perlu mengembalikan rilis tertentu setelah Anda menyadari bahwa itu dapat merusak aplikasi dalam beberapa kasus. Anda dapat mengeluarkan perintah berikut:

$ heroku rollback
Rolling back gentle-mesa-5445... done, v2
! Warning: rollback affects code and config vars; it doesn't add or
remove addons. To undo, run: heroku rollback v3

Atau, Anda bisa spesifik dan mengeluarkan perintah berikut:

$ heroku rollback <version name>

Namun, Anda harus mengambil tindakan pencegahan sebelum mengembalikan aplikasi, karena Anda mungkin harus mengembalikan status database Anda juga agar keduanya tetap sinkron. Balikkan migrasi database terlebih dahulu dan kemudian terapkan rollback ke aplikasi Anda setelah selesai. Untuk mengembalikan database pada aplikasi RoR, Anda dapat menggunakan perintah heroku rake db: rollback.

Ringkasan

Dalam bab ini, kami meninjau berbagai komponen penempatan di bawah Heroku. Kami juga meninjau sistem kontrol revisi Git, yang merupakan pusat pengembangan aplikasi di bawah Heroku. Selain itu, kami memahami cara mengoptimalkan ukuran aplikasi dengan mengabaikan file aplikasi yang tidak relevan sebelum mendorong kode, dan cara mengelola beberapa lingkungan untuk memastikan bahwa aplikasi bergerak ke produksi

hanya setelah disertifikasi bahwa ia berfungsi dengan benar di lingkungan yang serupa. Akhirnya, kami meninjau bagaimana kami dapat memonitor perubahan baru pada aplikasi melalui Deploy Hooks dan bagaimana kami dapat memeriksa informasi spesifik rilis melalui perintah manajemen rilis Heroku.

Pada bab selanjutnya, kita akan fokus pada cara menjalankan aplikasi Heroku.

BAB 5
MENYATUKAN SEMUANYA

Salah satu kekuatan inti dari platform Heroku adalah kemampuannya untuk berfungsi sebagai platform polyglot, tempat pengembang dapat memilih dan memilih yang didukung bahasa pemrograman dan kembangkan aplikasi web yang kaya fitur. The Heroku platform juga mendukung bahasa-bahasa ini dengan menyediakan add-on untuk memasukkan spesifik fitur (misalnya, halaman caching) di aplikasi web ini. Aplikasi web digunakan mulus, dipantau, dan dibagikan dengan pengembang lain dengan mengklik tombol.

Sebelumnya dalam buku ini, kami fokus pada aspek-aspek spesifik seperti membangun dan menggunakan aplikasi pada platform Heroku, dan belajar bagaimana berbagai teknik dapat digunakan untuk melakukan tugas yang berguna dengan platform.

Dalam bab ini, kami memberikan ringkasan dari semua yang telah kami pelajari sejauh ini. Kami akan pelajari yang berikut:

Bagaimana cara menambahkan dukungan Heroku dan menggunakan aplikasi Java di Heroku menggunakan Eclipse

- Cara mengelola aplikasi Heroku dan melakukan semua manajemen aplikasi operasi di dalam Eclipse

Kami bisa melakukan ini dengan bahasa lain dan IDE lainnya. Pilihan bahasa benar-benar tidak penting. Poin kuncinya adalah seberapa cepat aplikasi cloud pengembangan diaktifkan menggunakan Eclipse sebagai alat pengembangan dan penyebaran untuk Heroku.

Dukungan Heroku untuk Java

Heroku mulai fokus pada dukungan Ruby sebagai bahasa pemrograman pilihan. Sebagian besar fitur dan sampel yang digunakan di Heroku ditulis dalam Ruby. Bahkan sistem pembuatan Heroku menggunakan alat bantu pembuatan Ruby sebagai toolkit default untuk membangun dan menyebarkan aplikasi web di Heroku. Segera, Heroku memperluas dukungan bahasanya ke termasuk lebih banyak bahasa pemrograman, sehingga menjadikannya platform polyglot sejati—tempat Anda bisa menulis aplikasi dalam bahasa pemrograman yang didukung.

Selain Ruby, Heroku mulai mendukung Java, Python, PHP, Clojure, dan lainnya bahasa. Dukungan Heroku untuk bahasa-bahasa ini dikombinasikan dengan. Dan Bermain arsitektur telah membantu Heroku menjangkau audiens yang jauh lebih besar. Heroku sekarang berkembang menjadi salah satu pilihan terbaik untuk mengembangkan dan menggunakan aplikasi web dengan sangat dengan mudah dan cepat. Heroku telah menemukan pengikut pendekatan polyglot di seluruh Internet komunitas pengembang – mulai dari pekerja lepas hingga perusahaan perangkat lunak web yang lebih besar cepat membangun aplikasi web yang bermanfaat.

Di bagian ini, kami akan meninjau dukungan Heroku untuk salah satu yang paling populer bahasa pemrograman di dunia perangkat lunak – Java.

Dukungan umum untuk Java

Tumpukan Heroku Cedar dapat menjalankan berbagai jenis aplikasi web Java. Tumpukan Cedar menggunakan Maven sebagai sistem build untuk mengkompilasi kode sumber Java menjadi executable aplikasi. Dengan menetapkan keberadaan a file pom.xml, sistem Heroku mendeteksi bahwa dorongan kode tertentu adalah

aplikasi Java. Heroku menggunakan OpenJDK toolkit untuk dijalankan Aplikasi Java. Saat ini, tumpukan Heroku Cedar mendukung OpenJDK 6 dan 7.

Dukungan basis data untuk aplikasi Java

Ketika Anda membangun aplikasi web Java di Heroku, database Postgres secara otomatis disediakan untuk aplikasi. Jika paket Postgres default tidak sesuai dengan kebutuhan aplikasi Anda,

Anda dapat memutakhirkan paket tambahan basis data tergantung pada kebutuhan bisnis Anda.

Anda dapat menemukan rincian lebih lanjut dari paket yang tersedia di *https: //devcenter.heroku.com / artikel / heroku-postgres-plan.*

Konfigurasi lingkungan

Saat Anda menggunakan aplikasi web Java untuk pertama kalinya, Heroku menetapkan yang berikut ini variabel konfigurasi atau lingkungan untuk aplikasi Anda:

PATH Ini adalah sekumpulan direktori tempat executable akan dicari oleh proses pembuatan atau langkah lain dari penyebaran aplikasi dan eksekusi, misalnya, */ usr / bin: / bin* (*varian* UNIX) *dan C: \ <JAVA_HOME> \ bin; C: \ Windows.*

JAVA_OPTS Berikut ini adalah opsi baris perintah untuk Java penyusun:

- Xss512k: Opsi ini mengatur ukuran Java stack
- -Xmx384m: Opsi ini mengatur memori tumpukan maksimum dialokasikan
- -XX: + UseCompressedOops: Opsi ini memungkinkan referensi menjadi 32-bit dalam JVM 64-bit dan akses dekat ke 32 GB tumpukan Ini dapat disesuaikan secara manual agar sesuai dengan kebutuhan spesifik lingkungan aplikasi. Misalnya, di Heroku, ada batas atas ke memori dyno, sehingga ukuran tumpukan maksimum perlu disesuaikan agar berada di bawah

batas memori dyno di untuk memungkinkan eksekusi aplikasi Anda yang optimal.

MAVEN_OPTS Opsi baris perintah untuk alat Maven Build adalah sebagai berikut:

- -Xss512k: Ini mengatur ukuran Java stack
- -Xmx384m: Ini mengatur memori tumpukan maksimum dialokasikan
- -XX: + UseCompressedOops: Opsi ini memungkinkan referensi menjadi 32-bit dalam JVM 64-bit dan akses tutup hingga 32 GB tumpukan

Opsi ini dapat disesuaikan secara manual untuk memenuhi kebutuhan spesifik aplikasi build.

PORT Ini adalah port HTTP tempat proses web akan mengikat.
DATABASE_URL This is the database connection URL to be used by the application.

Jika Anda membutuhkan lebih banyak memori dyno untuk aplikasi Anda dan mulai menggunakan 2X dynos untuk memenuhi kebutuhan memori aplikasi Anda, perubahan ini tidak akan tercermin secara otomatis di aplikasi Anda konfigurasi. Anda perlu mengedit JAVA_OPTS dan MAVEN_OPTS untuk mencerminkan perubahan.

Mengintegrasikan Eclipse dengan Heroku

Pada langkah-langkah berikut, kami menunjukkan cara mengkonfigurasi pengembangan Heroku lingkungan di Eclipse. Eclipse adalah salah satu terintegrasi yang paling populer dan kuat platform pengembangan perangkat lunak di pasar saat ini. Jutaan pengembang menggunakan Eclipse untuk menulis dan

menggunakan enterprise dan aplikasi Java lainnya menggunakan Eclipse untuk Edisi JEE.

Pada bagian ini, pertama-tama kami membuat daftar prasyarat untuk memungkinkan pengembangan a Aplikasi cloud Heroku menggunakan Eclipse, dan kemudian menunjukkan pendekatan langkah demi langkah untuk mengkonfigurasi plugin Eclipse untuk Heroku. Akhirnya, kami membuat contoh aplikasi web di Java dan menyebarkan aplikasi. Kami juga menunjukkan kepada Anda bagaimana mengelola dan memantau Java aplikasi web menggunakan fitur plugin Eclipse untuk Heroku.

Prasyarat

Sebelum Anda memulai perjalanan Anda menyebarkan aplikasi Heroku menggunakan Eclipse, lakukan langkah-langkah berikut:

1. Dapatkan akun Heroku *di www.heroku.com.*
2. Unduh Eclipse. Pergi ke *http://www.eclipse.org/downloads/* dan unduh Eclipse IDE untuk pengembang Java EE. Buka zip ZIP yang diunduh dan mengembalikan konten paket ke *C: \ Eclipse atau direktori lainnya.*

Mengkonfigurasi Heroku di Eclips

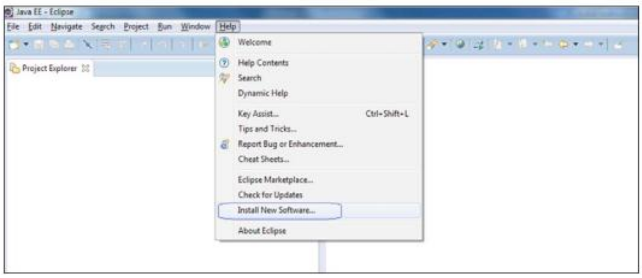
Cukup mudah untuk menambahkan dukungan Heroku untuk menulis, menggunakan, dan mengelola aplikasi web Heroku dari Eclipse. Jika Anda belum pernah menggunakan Heroku sebelumnya dan Anda adalah pengembang Java, menggunakan Eclipse adalah cara terbaik untuk memulai secara instan dengan Heroku. Ada instruksi yang tersedia untuk melakukan hal yang sama melalui Heroku CLI *di https://devcenter.heroku.com/articles/getting-started-with-java. Mari kita mulai.*

Menginstal plugin Eclipse untuk Heroku

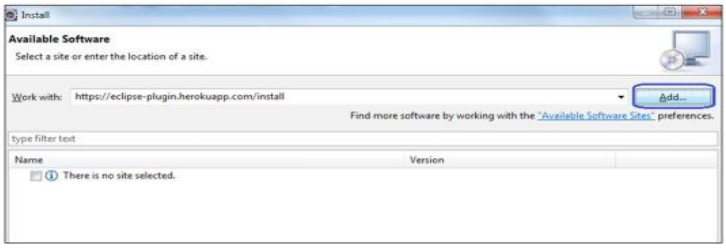
Langkah pertama untuk menulis dan menggunakan aplikasi web Java di Heroku adalah menginstal Plugin Eclipse untuk Heroku. Prosedur yang cukup mudah ini membantu Anda mengatur plugin Eclipse Heroku yang kemudian dapat digunakan untuk membantu

Anda berinteraksi dengan Lingkungan Heroku dari Eclipse. Jadi, siap? Lakukan langkah-langkah berikut untuk menginstal plugin Eclipse untuk Heroku:

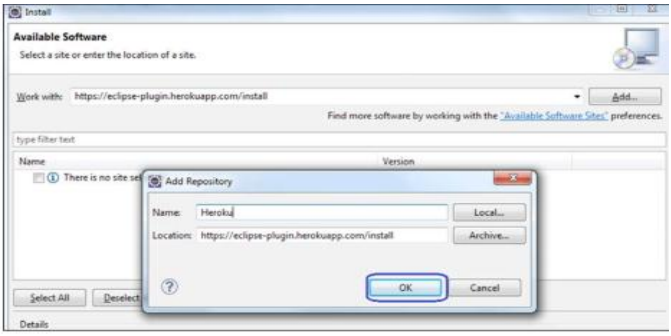
1. Buka Eclipse dan arahkan ke menu **Help,** dan di dalamnya, pilih **Install Opsi menu Perangkat Lunak baru** seperti yang ditunjukkan pada tangkapan layar berikut:



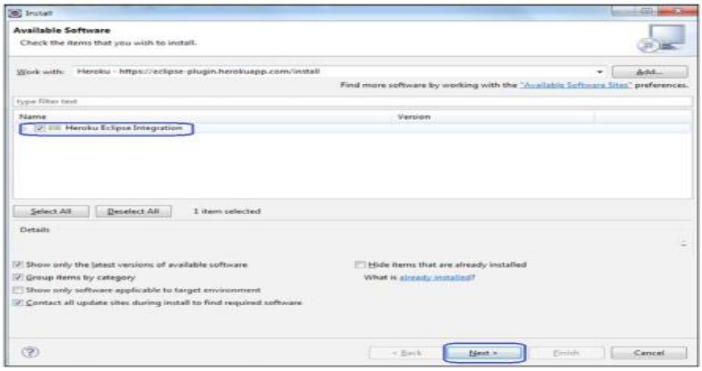
2. Masukkan URL plugin Eclipse Heroku, *https: // eclipse-plugin. herokuapp.com/install* , dan klik Tambah seperti yang ditunjukkan di bawah ini tangkapan layar:



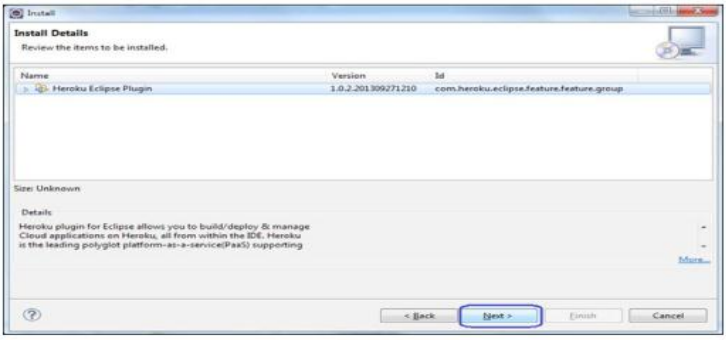
3. Masukkan nama repositori sebagai Heroku dan klik OK seperti yang ditunjukkan pada tangkapan layar berikut:



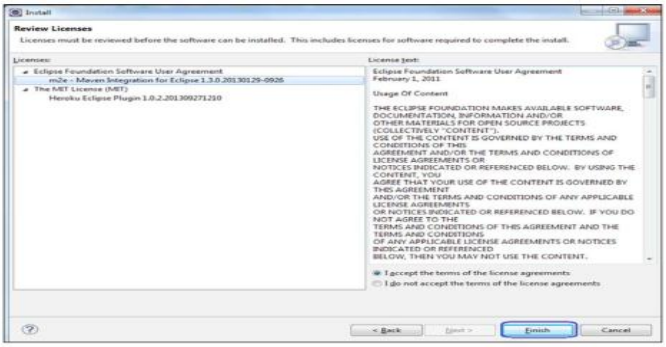
4. Periksa nama plugin Heroku Eclipse Integration dan klik Next as ditunjukkan pada tangkapan layar berikut:



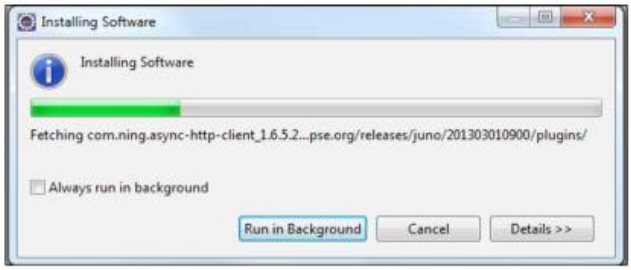
5. Tinjau detail plugin dan klik Berikutnya seperti yang ditunjukkan di tangkapan layar berikut:



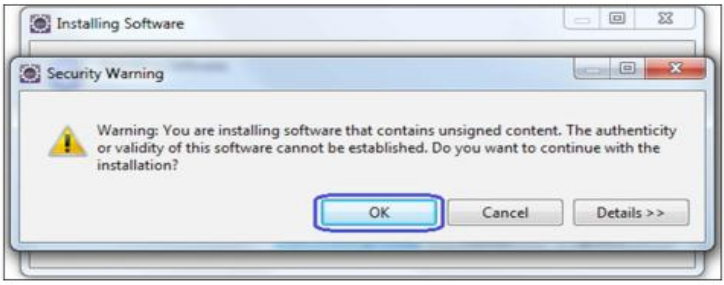
6. Terima perjanjian lisensi dan klik Finish seperti yang ditunjukkan pada tangkapan layar berikut:



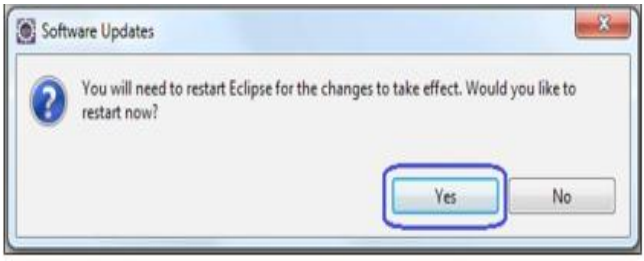
7. Perangkat lunak Heroku sekarang sedang diinstal.



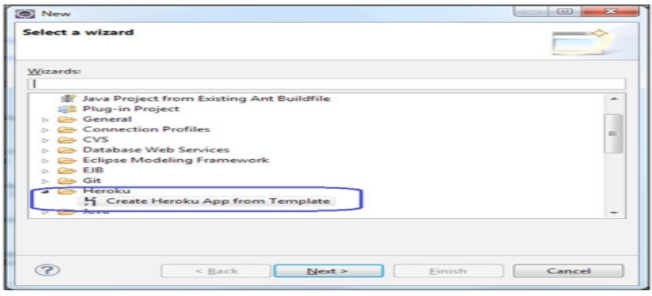
8. Klik OK untuk mengabaikan peringatan konten yang tidak ditandatangani.



9. Klik Ya untuk membuat perubahan efektif.



10. IDE Eclipse akan memulai kembali. Anda dapat pergi ke Baru | Lainnya untuk memverifikasi apakah Dukungan Heroku telah ditambahkan.



Menyiapkan Heroku untuk pengembangan

Mengembangkan aplikasi berbasis Java menggunakan Heroku semudah menulis aplikasi Java Anda di tempat. Yah, itu cukup transparan sejauh mengatur dukungan Heroku prihatin. Anda memerlukan plugin dan dibuat dan Anda menginstalnya. Itu dia. Kami memberi tahu Eclipse Akun Heroku untuk digunakan untuk aplikasi yang baru apa kredensial keamanannya diperlukan untuk mengakses layanan API dan kode push / pull dari platform Heroku.

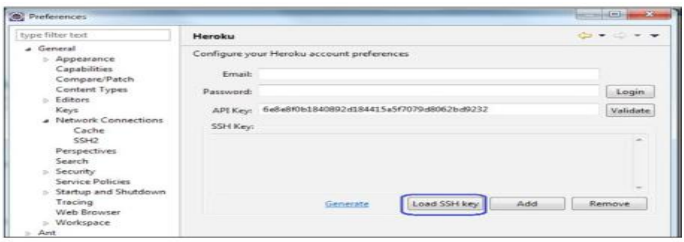
Menyiapkan dukungan SSH

Seperti dibahas dalam bab-bab sebelumnya, SSH digunakan sebagai mekanisme keamanan untuk transport perubahan Anda ke dan dari Heroku. Anda dapat mengkonfigurasi SSH untuk pengembangan aplikasi Heroku Anda dalam dua cara berikut:

- Membuat kunci baru dengan menavigasi ke Windows | Preferensi | Heroku
- Memuat kunci SSH yang ada yang sudah disiapkan untuk mesin Anda bekerja pada

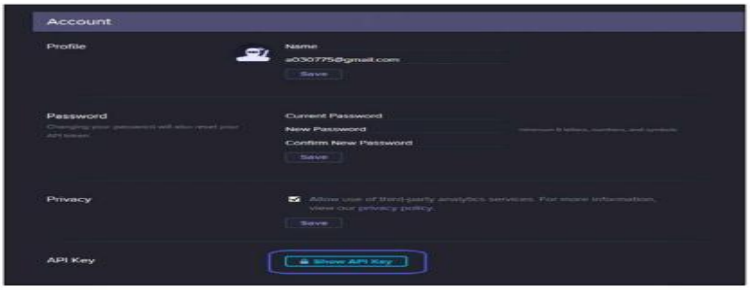
Karena saya sudah mengerjakan Heroku menggunakan antarmuka **baris perintah (CLI), saya sudah memiliki kunci SSH**. Menghasilkan yang baru sama mudahnya. Kami akan lihat cara memuat kunci SSH yang ada di langkah-langkah berikut:

1. Buka Eclipse.
2. Pergi ke Windows | Preferensi | Heroku dan klik pada kunci Load SSH.

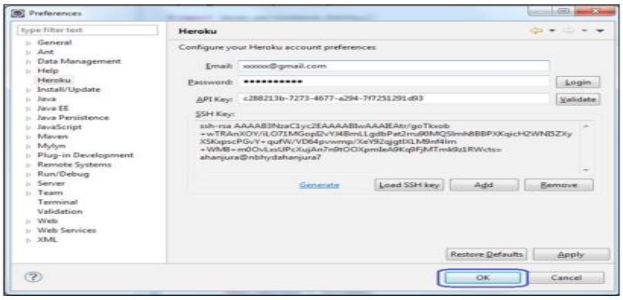


3. Pilih file dari folder .ssh dari direktori pengguna Anda.

Ini akan memuat kunci SSH ke dalam preferensi Eclipse Heroku. Seperti yang sudah saya lakukan memiliki kunci API, saya akan masuk ke akun Heroku saya untuk mengambil kunci dengan **menavigasi ke Akun | Tampilkan Kunci API.**

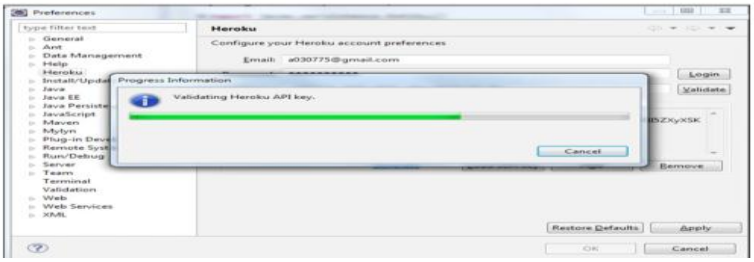


4. Masukkan kunci API yang diambil di bagian Kunci API dan klik OK.



Heroku akan memvalidasi informasi yang dimasukkan termasuk kunci API dan akan kembalikan kesalahan, jika ada. Kunci SSH dan API sekarang diunggah ke konfigurasi Eclipse. Ini melengkapi konfigurasi kunci SSH dan API untuk menggunakan

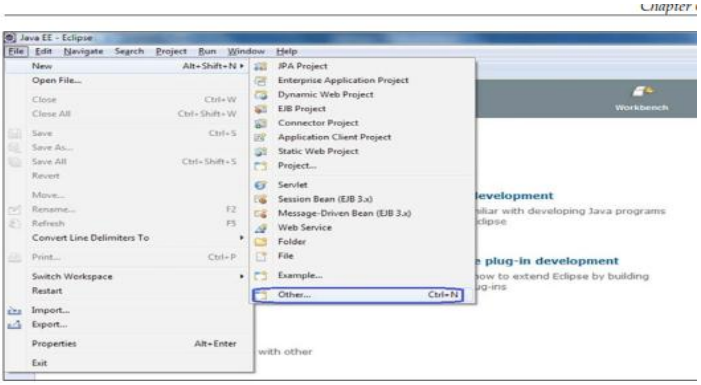
Aplikasi web Heroku melalui Eclipse. Sekarang, Anda siap untuk pergi dan menulis yang pertama Aplikasi web Heroku berbasis Eclipse.



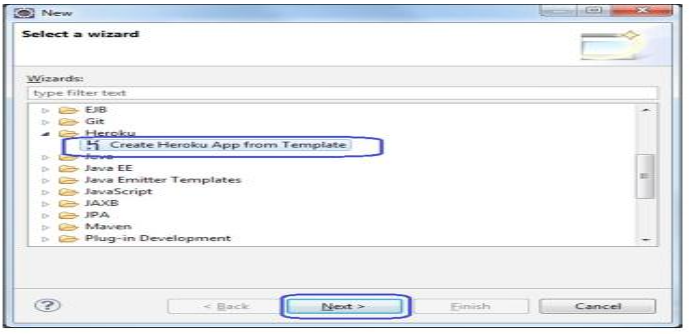
Membuat aplikasi Java Heroku baru di Eclipse

Setelah Anda menginstal plugin Eclipse untuk Heroku dan mengatur kunci SSH dan API di Eclipse, Anda siap membangun aplikasi web Java berbasis Heroku baru. Melakukan ini, lakukan langkah-langkah berikut:

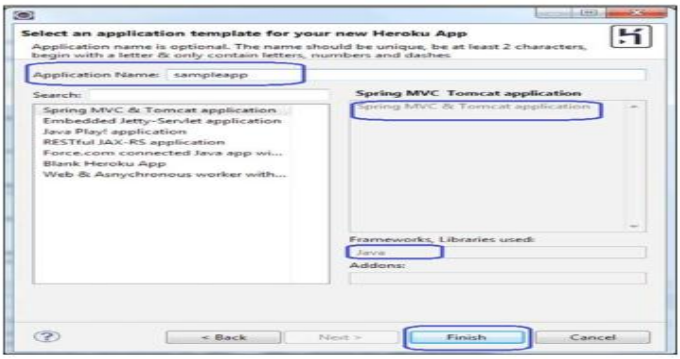
1. Untuk memulai, buka Eclipse dan buka **New | Lainnya** seperti yang ditunjukkan pada tangkapan layar berikut:



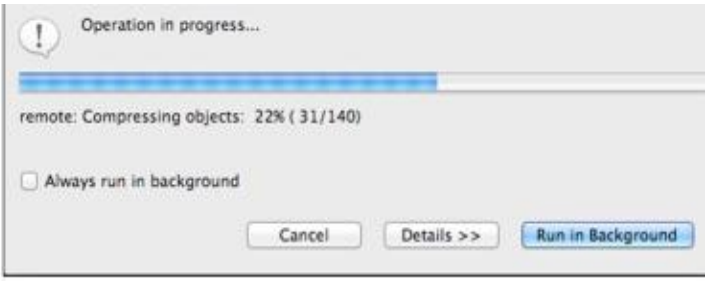
2. Klik pada opsi Buat Aplikasi Heroku dari Template.



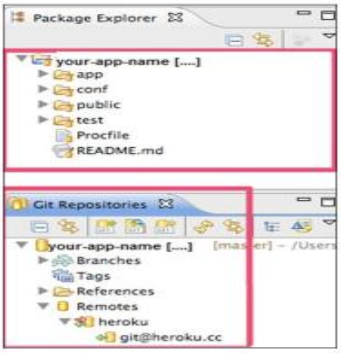
3. Anda dapat membuat aplikasi dari daftar aplikasi templat yang sudah ada tersedia di Eclipse. Masukkan nama aplikasi (atau biarkan kosong, di yang mana nama akan di-autogenerasikan untuk aplikasi) dan klik Finish.



Aplikasi baru, sampleapp, akan dibuat untuk Anda. Begitu aplikasi dibuat, plugin akan menarik repositori kode sumber sebagai lokal Git repositori.



4. Setelah aplikasi dibuat, Anda dapat membuka Paket **Explorer Eclipse** folder dan perhatikan struktur direktori yang baru dibuat untuk aplikasi.



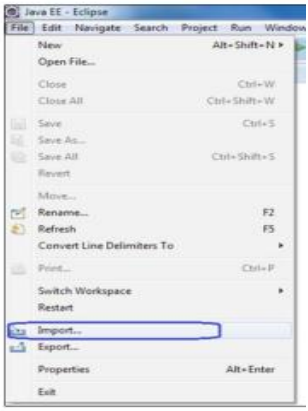
Sekarang Anda siap untuk melakukan perubahan pada aplikasi web Anda. Setelah selesai, Anda bisa mendorong pembaruan untuk Heroku menggunakan operasi yang disediakan oleh Eclipse. Menggunakan Eclipse, Anda bisa cukup banyak melakukan apa pun yang dapat Anda lakukan menggunakan antarmuka baris perintah Heroku. Anda dapat melihat informasi aplikasi, menyebarkan aplikasi yang dimodifikasi, memantau log, skala aplikasi, menyegarkan aplikasi, dan banyak lagi. Kami akan segera bereksperimen masing-masing aspek bermanfaat pengembangan aplikasi cloud berbasis Eclipse ini.

Menggunakan aplikasi Heroku yang ada

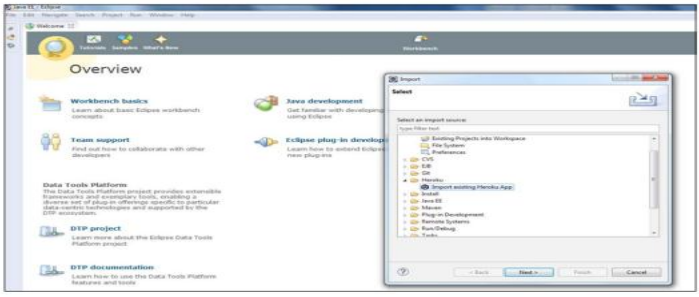
Heroku memberlakukan batasan pada berapa banyak aplikasi yang dapat Anda miliki di akun Heroku Anda untuk menghindari penyalahgunaan oleh pengguna yang dapat mencekik infrastruktur Heroku dengan menjalankan a sejumlah besar aplikasi, sehingga banyak menggunakan sumber daya sistem Heroku seperti itu seperti memori, CPU, dan bandwidth jaringan. Batasnya adalah 100 aplikasi untuk akun yang diverifikasi dan lima aplikasi untuk yang belum diverifikasi. Karena itu, bermanfaat untuk menggunakan kembali aplikasi yang ada yang mungkin tidak lagi digunakan secara aktif dan menghabiskan sumber daya. Kita bisa menggunakan aplikasi Heroku yang ada ini dan mengubahnya untuk membuat aplikasi web yang bermanfaat.

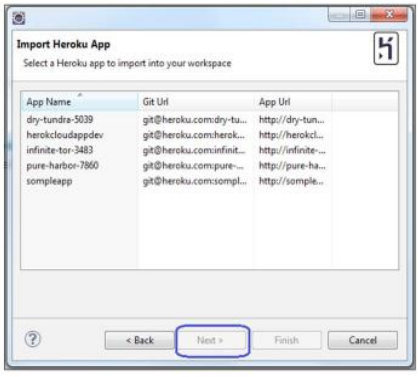
Untuk mengimpor aplikasi dari akun Heroku Anda, Anda dapat melakukan langkah-langkah berikut:

1. Navigasikan | ke **File Impor.**



1. Pilih opsi Impor Aplikasi Heroku yang ada di menu Impor.



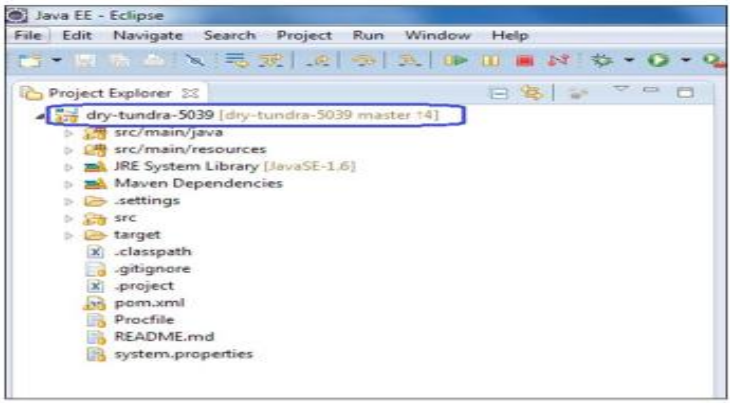


`Kami akan melihat daftar aplikasi Heroku yang dapat diimpor ke ruang kerja Anda. Pilih dry-tundra-5039 dan klik Next.

2. Selama impor, Eclipse dapat mendeteksi jenis proyek (opsional berdasarkan pilihan pengembang) dan sesuai dengan itu menerapkan karakteristik tipe itu ke proyek yang diimpor. Proyek kami dibangun menggunakan Maven. Saat mengimpor proyek, Eclipse akan menerapkan sifat - sifat proyek Maven ke kami proyek. Pilih **opsi proyek yang terdeteksi** Otomatis dan klik **Selesai.**

Eclipse akan mengimpor proyek yang dipilih dan membuat pengaturan proyek yang diperlukan

3. Pergi ke tampilan **Project Explorer** dengan menavigasi ke **Window | Tampilkan View** pada menu Eclipse dan kemudian mengklik **Project Explorer**. Verifikasi itu proyek yang diimpor tersedia untuk digunakan dan diedit, seperti yang ditunjukkan dalam tangkapan layar berikut:



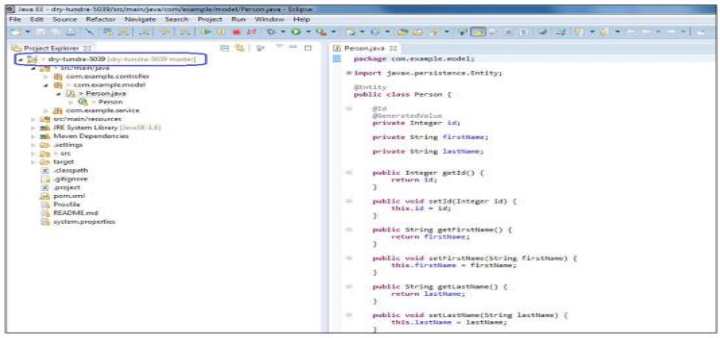
Kami telah berhasil mengimpor aplikasi Heroku ke Eclipse. Kita sekarang dapat mengedit kode sumber yang terletak di folder src

Pushing code to Heroku

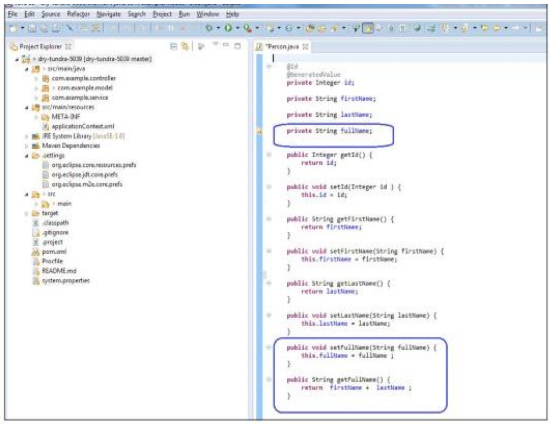
Sekarang kita memiliki proyek yang berfungsi di Eclipse, mari kita memodifikasi kode sumber lalu

dorong perubahan ke Heroku dengan melakukan langkah-langkah berikut:

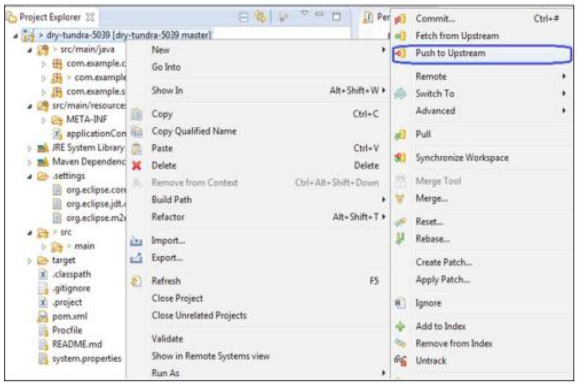
1. Buka Eclipse dan buka tampilan **Project Explorer** dengan menavigasi ke **Window** | Perlihatkan **tampilan pada menu** Eclipse dan kemudian klik pada **Project Explorer**



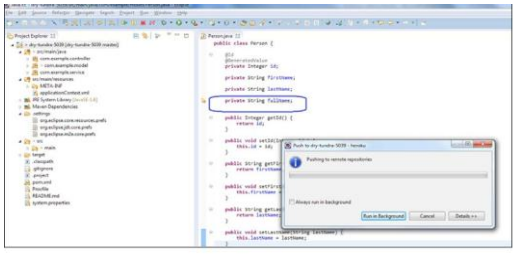
2. Ubah file sumber Person.java dengan menambahkan metode baru di akhir kelas Person seperti yang ditunjukkan pada tangkapan layar berikut:



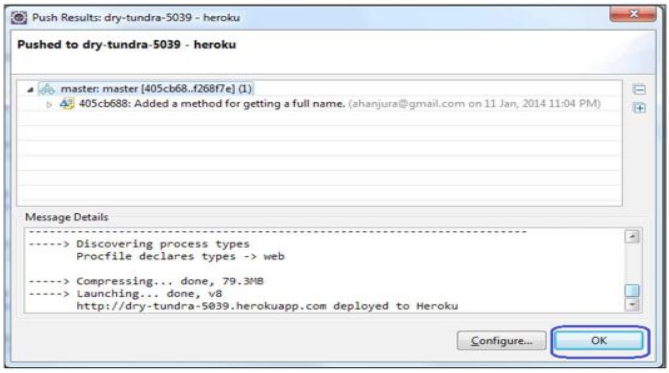
3. Setelah melakukan perubahan yang diperlukan dan menyimpannya, pilih proyek di tampilan **Project Explorer** dan kemudian arahkan ke **Tim | Dorong ke Hulu**, seperti yang ditunjukkan pada tangkapan layar berikut:



Dengan mengklik opsi menu Push to Upstream, perubahan kode Anda akan didorong ke repositori jarak jauh aplikasi Anda.



Akhirnya, Eclipse menampilkan pesan push yang berhasil. Klik OK untuk menyelesaikan proses mendorong kode Anda ke Heroku.

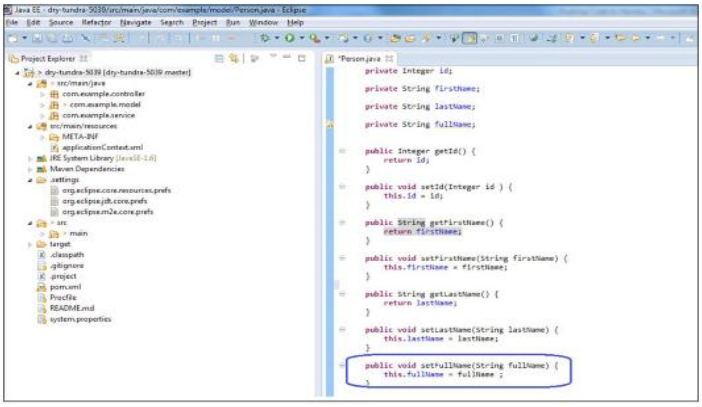


Pushing code to the Git repository

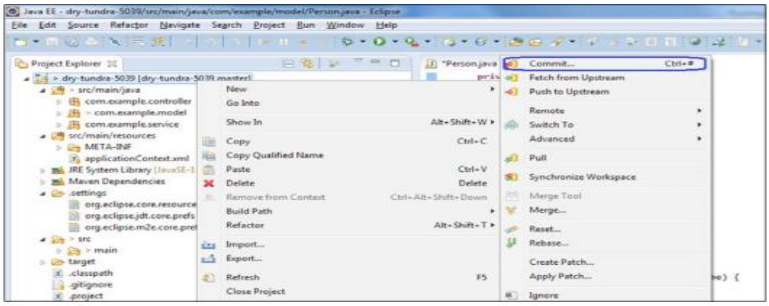
Eclipse memungkinkan Anda untuk mendorong kode Anda ke sistem kontrol versi terdistribusi Git sama seperti klien Git lainnya.

Anda dapat memilih kode untuk check-in, masukkan kredensial repositori Git Anda, dan kemudian masukkan pesan komit Anda (opsional). Klik **OK**, dan voila, Anda sudah selesai. Ini adalah diilustrasikan sebagai berikut:

1. Buka Eclipse dan arahkan **ke Window | Tampilkan** Tampilan pada menu Eclipse dan klik pada **Project Explorer**.

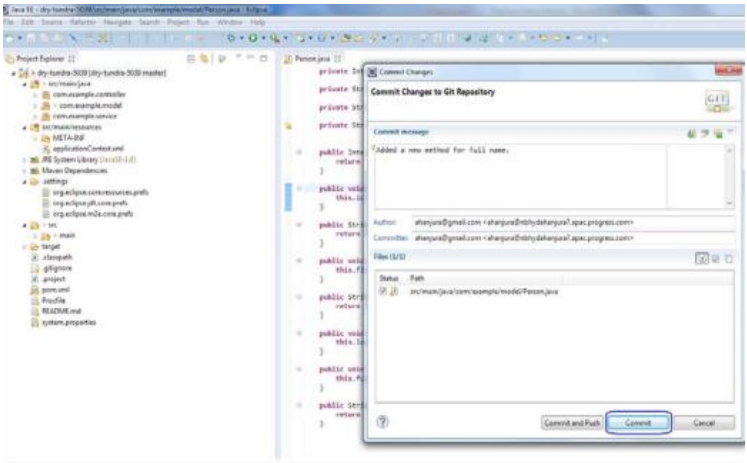


2. Buat perubahan pada kode dan simpan di Eclipse. Klik kanan pada proyek dan arahkan ke **Tim | Melakukan**.



3. Masukkan kredensial Git Anda untuk mengakses repositori Git Anda.

4. Masukkan pesan **komit** untuk perubahan yang telah Anda buat dan klik Melakukan. Perhatikan bahwa ada opsi **Komit dan Dorong juga**. Pilihan ini akan mengkomit perubahan Anda ke Git dan mendorongnya ke Heroku dalam sekali jalan



Sekarang, perubahan kode Anda akan disinkronkan dengan repositori Git dan siap digunakan oleh pengembang lain.

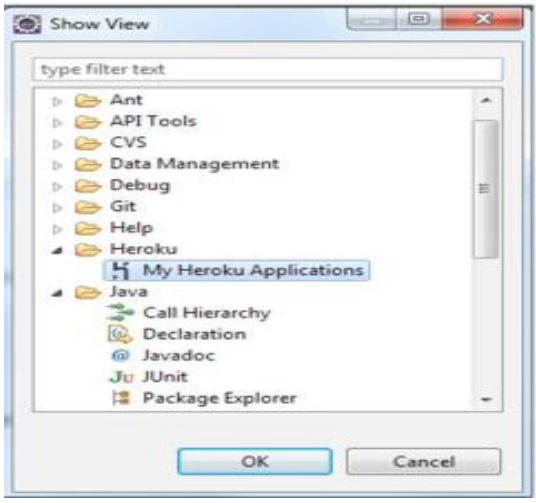
Mengelola aplikasi Heroku di Eclipse

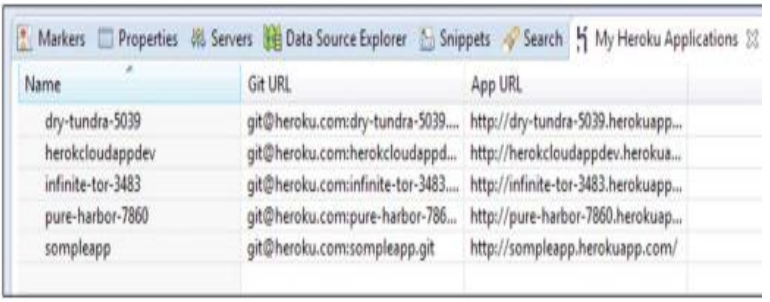
Anda dapat memiliki banyak aplikasi web Java yang digunakan di Heroku pada satu titik waktu. Anda dapat masuk ke akun Heroku Anda dan mengelola aplikasi web ini, atau Anda bisa gunakan fasilitas yang disediakan dalam Eclipse untuk melakukan semua yang Anda butuhkan untuk mengelola aplikasi web Anda. Di bagian saat ini, kita akan melihat cara menggunakan Eclipse untuk melakukan yang terbaik tugas umum untuk mengelola aplikasi web Anda.

Melihat aplikasi Heroku Anda

Eclipse membuat pemantauan dan pengelolaan aplikasi web Heroku Anda sangat sederhana dengan memberikan tampilan Heroku di opsi menu **Window**. Dengan mengklik a beberapa tombol, Anda dapat dengan mudah meninjau daftar aplikasi Heroku yang Anda miliki. Ini bisa dilakukan dengan melakukan langkah-langkah berikut:

1. Buka Eclipse, buka **Window | Tampilkan Tampilan** | Lainnya, lalu pilih **Saya Aplikasi Heroku.**





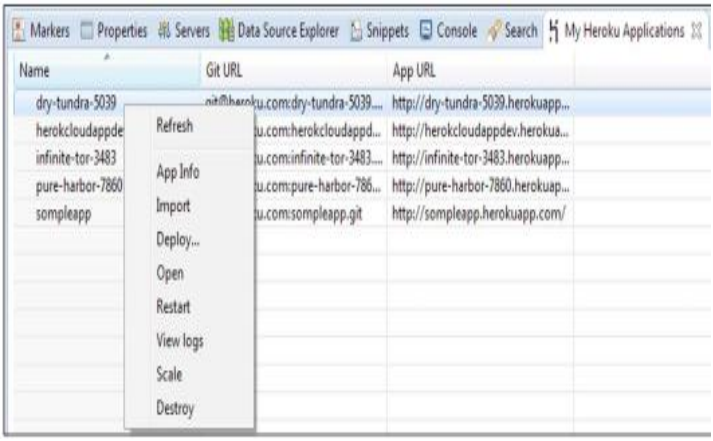
Eclipse akan menampilkan daftar aplikasi Heroku milik akun Anda, atau Anda adalah kolaborator di, bersama dengan informasi relevan lainnya seperti nama aplikasi, URL Git, dan URL aplikasi untuk aplikasi itu, seperti yang ditunjukkan di tangkapan layar berikut:

Mendapatkan detail aplikasi

Di Eclipse, cukup mudah untuk meninjau detail berbagai aplikasi Heroku dan melakukan berbagai operasi pada mereka, seperti kita menggunakan command-line Heroku antarmuka. Eclipse adalah konsol satu atap dan platform pengembangan untuk bangunan dan mengelola aplikasi Heroku berbasis Java. Anda dapat mengelola semua sumber daya aplikasi termasuk add-on, koneksi basis data, dan perpustakaan tambahan dari Eclipse.

Meninjau detail aplikasi

Untuk meninjau daftar operasi yang tersedia untuk aplikasi tertentu, klik kanan pada aplikasi. Daftar operasi yang tersedia untuk setiap aplikasi sangat mirip dengan daftar yang tersedia dari baris perintah Heroku, tetapi lebih mudah ditemukan dalam Eclipse. Tangkapan layar berikutnya menunjukkan daftar operasi yang tersedia untuk aplikasi **dry-tundra-5039:**



Set operasi yang tersedia untuk setiap aplikasi meliputi:

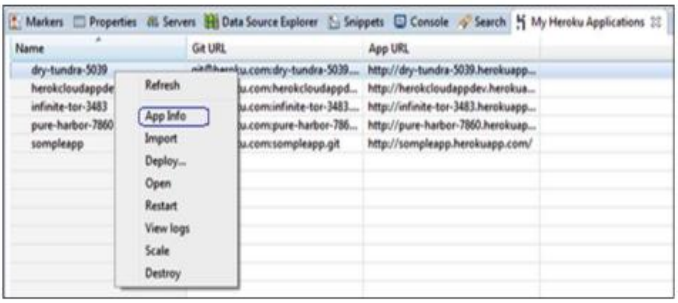
- **Refresh**: Ini menyegarkan daftar aplikasi Heroku dengan pembaruan terbaru.
- **Info Aplikasi:** Ini menyediakan daftar opsi tingkat aplikasi berikut untuk pengguna:
    - **Info Aplikasi** : Ini mencantumkan nama, URL, URL repositori Git, dan nama domain aplikasi. Anda dapat mengganti nama aplikasi demikian juga.
    - **Kolaborator** : Ini mengulas dan mengelola daftar pengembang mengerjakan aplikasi ini.
    - **Variabel Lingkungan** : Ini mengulas dan mengelola daftar variabel lingkungan / variabel konfigurasi tersedia di aplikasi.
    - **Proses** : Ini memberikan detail dino yang terkait dengan ini aplikasi berdasarkan jenis proses.
- **Impor:** Ini mengimpor aplikasi web ke Eclipse.
- **Menyebarkan** : Ini menyebarkan aplikasi ke Heroku.
- **Buka:** Ini membuka aplikasi di browser.
- **Restart:** Ini akan me-reboot dino aplikasi.
- dinamika web. **Lihat log:** Ini melihat log aplikasi untuk aplikasi yang diberikan.
- **Skala:** Ini menaikkan atau menurunkan aplikasi web dengan menambah / mengurangi jumlah

- **Hancurkan:** Ini menghancurkan / membersihkan aplikasi web dan melepaskan semua sumber daya terkait.

Masuk lebih dalam ke informasi aplikasi

Untuk mendapatkan informasi lebih rinci tentang aplikasi tertentu, buka **My Tab Aplikasi Heroku** dan kemudian ikuti langkah-langkah ini:

1. Klik kanan pada aplikasi tertentu yang Anda minati dan klik di **Info App.**

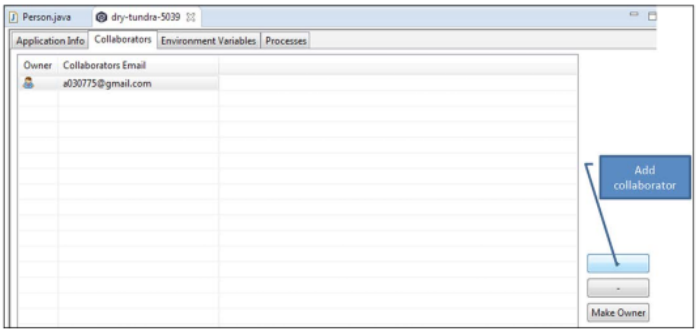


Tab **Info Aplikasi** memberikan detail tentang nama, URL situs web, Git URL repositori, dan nama domain aplikasi. Anda juga dapat mengganti nama aplikasi menggunakan tombol Ubah nama.

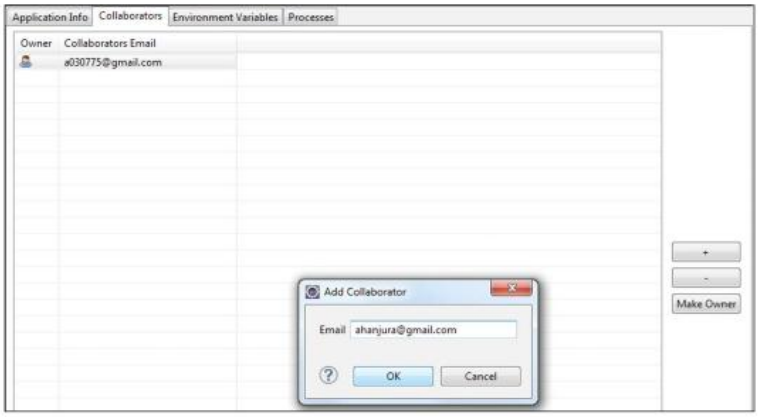
Menambahkan kolaborator ke aplikasi

Menambahkan kolaborator membantu Anda membagikan proyek Anda dengan pengembang lain yang mungkin tertarik untuk berkontribusi dalam pengembangan dan pengelolaan aplikasi Anda. Menambahkan atau mengelola kolaborator cukup mudah menggunakan Eclipse. Untuk melakukan ini, lakukan langkah-langkah berikut:

1. Buka tab **Kolaborator** dan klik tombol + untuk menambahkan kolaborator proyek saat ini.



2. Tambahkan alamat email kolaborator baru dan klik OK.



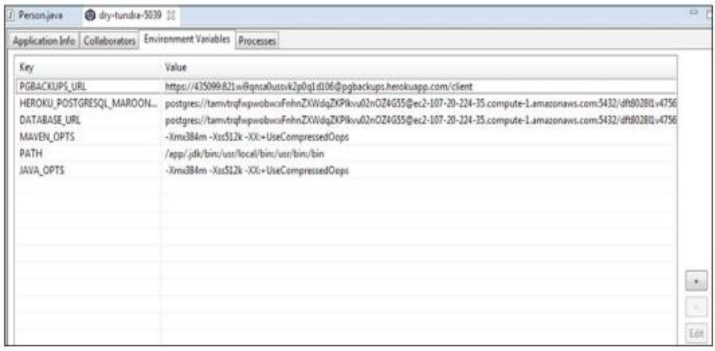
3. Kolaborator baru ditambahkan ke daftar kolaborator untuk proyek yang diberikan. Pemilik aplikasi ditunjukkan oleh ikon di bidang pemilik seperti yang ditunjukkan pada tangkapan layar berikut:

Anda juga dapat menghapus kolaborator dengan memilih nama dan mengklik tombol -, yang terletak tepat di bawah tombol

add. Untuk menetapkan kembali pemilik aplikasi, pilih kolaborator dan kemudian klik tombol Buat pemilik.

Mengubah variabel lingkungan

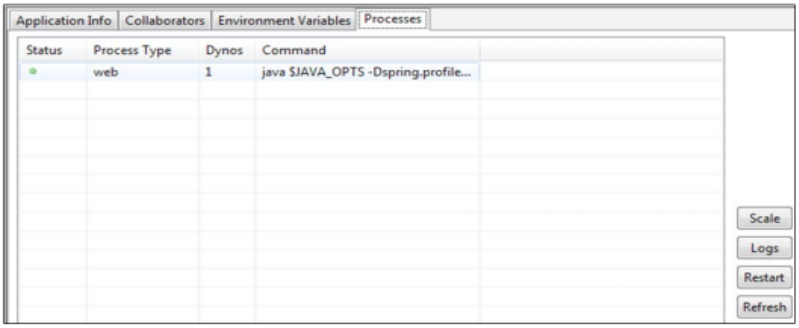
Mengklik pada tab **Variabel Lingkungan** akan menampilkan daftar lingkungan yang ada variabel atau variabel konfigurasi yang tersedia untuk aplikasi. Memilih variabel lingkungan tertentu dan klik tombol **Edit** untuk memodifikasi variabel lingkungan. Anda dapat mengklik tombol + untuk menambahkan lingkungan baru variabel. Untuk menghapus variabel lingkungan tertentu, pilih lingkungan variabel dan kemudian klik pada tombol -. Konfirmasikan pilihan Anda untuk menghapus variabel lingkungan secara permanen.



Manajemen proses Heroku di Eclipse

Satu keuntungan utama yang membedakan platform Heroku adalah kemudahan menggunakannya. Itu Eclipse IDE juga menyediakan antarmuka yang sangat intuitif untuk mengelola **proses Heroku** dari tab **Processes** dari aplikasi web.

Klik pada tab Proses. Anda akan melihat informasi terkait proses yang sangat mirip dengan format Procfile. Palet kanan juga menyediakan operasi tambahan yang dapat dieksekusi untuk aplikasi yang dipilih.



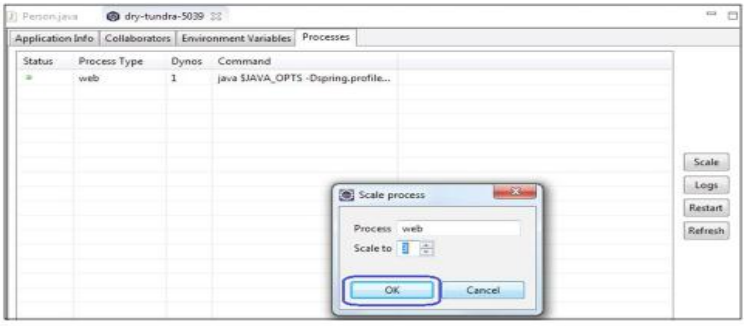
Antarmuka Eclipse menyediakan operasi terkait **proses** berikut di Halaman proses:

- **Skala:** Ini memungkinkan Anda untuk menaikkan / menurunkan jumlah dino tertentu jenis proses untuk aplikasi saat ini
- **Log:** Ini menunjukkan log aplikasi untuk aplikasi web tertentu
- **Restart:** Ini akan me-reboot dino yang terkait dengan aplikasi web
- **Refresh:** Ini memperbarui atau menyegarkan aplikasi Heroku saat ini dengan informasi konfigurasi terbaru

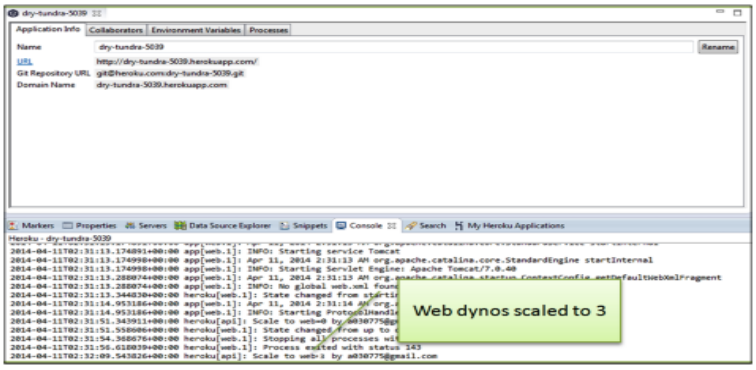
Mari kita selidiki masing-masing operasi terkait proses ini satu demi satu.

Menskalakan dinamika aplikasi Anda

Untuk meningkatkan / menurunkan dino untuk jenis proses tertentu dari aplikasi web Anda, pilih jenis proses dan masukkan nilai skala, yaitu, jumlah dino yang Anda butuhkan untuk kinerja aplikasi optimal. Klik **OK** untuk mengkonfirmasi perubahan.

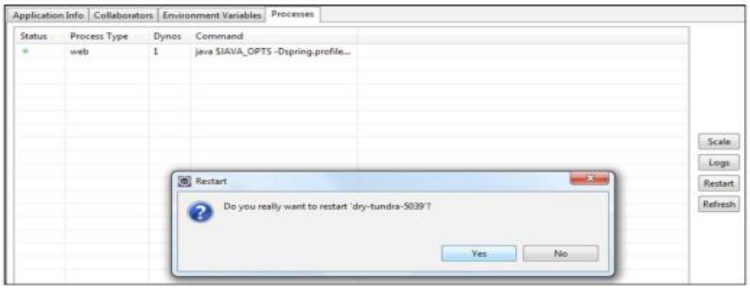


Beralih ke tampilan **Log** untuk melihat pembaruan yang dibuat oleh Heroku. Heroku naik jumlah dyno untuk jenis proses web dan membuat dua lagi dinamika web untuk aplikasi web Anda.

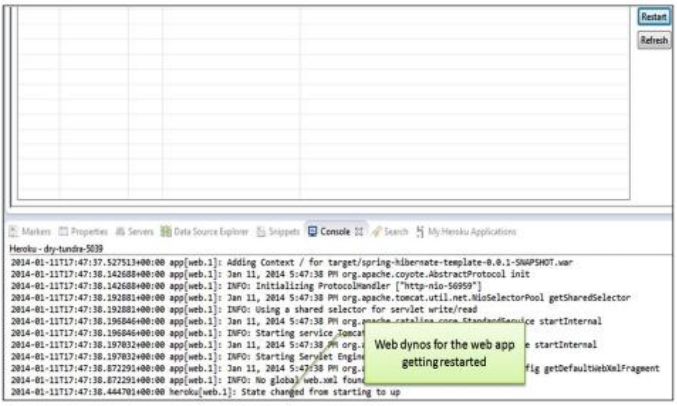


Mulai ulang aplikasi web

Anda Anda juga dapat memulai kembali aplikasi web menggunakan tombol **Restart** pada tab **Processes** aplikasi web Anda. Pada mengklik Restart, Heroku langsung me-restart dino terkait aplikasi web Anda. Tangkapan layar berikut menunjukkan dialog untuk mengonfirmasi pilihan kami restart aplikasi web **dry-tundra-5039**:



Untuk memverifikasi apa yang terjadi di balik tirai platform Heroku, beralihlah ke Tampilan **log.**



Di bagian ini, kami belajar cara menggunakan berbagai opsi yang tersedia di Eclipse untuk mengelola aplikasi Heroku. Beberapa klik dapat membantu Anda melihat, mengubah, dan perluas aplikasi web Anda. Eclipse memberikan abstraksi yang sangat kuat atas Antarmuka baris perintah Heroku, yang membantu pengembang melakukan hampir semua hal operasi menggunakan klik mouse sederhana. Inilah yang benar-benar dibutuhkan pengembang.

Ringkasan

Bab ini membantu kami bereksperimen dengan apa yang kami pelajari di bagian awal ini Book. Kami dapat membangun aplikasi web menggunakan Java toolset dan didemonstrasikan betapa mudahnya untuk bangun dan berjalan dengan pengembangan cloud Heroku aplikasi dalam waktu singkat. Bukankah itu mudah?

Pilih bahasa pemrograman apa pun yang didukung Heroku dan kemudahan membangun aplikasi tidak akan berbeda. Dalam hal itu, Heroku tidak menemukan kembali roda untuk pengembang, tetapi itu menciptakan platform yang tepat, membuat add-on yang tepat tersedia, dan membantu Anda menggunakan aplikasi web dengan mudah. Semua yang perlu dikhawatirkan oleh pengembang tentang adalah bagaimana menulis logika bisnis aplikasi; tidak ada keanehan platform, tidak instalasi perangkat lunak pihak ketiga yang tidak

praktis, dan tanpa pengelolaan malam tanpa tidur rilis / penyebaran baru.

Pada bab selanjutnya, kita akan melihat secara rinci ekosistem perangkat lunak di sekitarnya Heroku. Kami akan belajar cara menulis aplikasi web dalam cloud IDE, memberi Anda pengalaman pengembangan berbasis cloud end-to-end. Kami akan memperkenalkan Anda kepada Dukungan Heroku untuk database PostgreSQL dan kemudian memberikan tips tentang cara melakukannya buat aplikasi Anda berfungsi dengan DB PostgreSQL. Kami akan meninjau langkah-langkahnya secara rinci untuk mengelola DNS untuk aplikasi Anda. Saya akan membahas yang baru diperkenalkan fitur untuk dino di Heroku yang membantu menjalankan aplikasi web lebih cepat dan lebih banyak efisien.

BAB 6
PRAKTEK HEROKU

Nah, ini dia. Sejauh ini, dalam buku ini, kami telah memahami dasar-dasar platform, dan kami telah mempelajari cara kerja Heroku di belakang layar. Kami meninjau infrastruktur dasar yang memungkinkan Heroku, apakah itu adalah mekanisme proses dyno atau lingkungan logplex logging. Kami juga melihat daftar alat yang didukung pada platform Heroku yang membuat pengembangan aplikasi pada platform menjadi sangat mudah. Infrastruktur pengaya Heroku semakin membantu aplikasi Anda untuk menggunakan layanan terbaik untuk memenuhi kebutuhan aplikasi Anda, apa pun itu. Dalam bab ini, kita akan mengeksplorasi beberapa praktik terbaik saat mengembangkan aplikasi pada platform Heroku.
Kami akan membahas topik-topik berikut dalam bab ini :

- **Cara mengembangkan aplikasi web di cloud**: Kami akan mengeksplorasi cara menarik untuk melakukan pengembangan web pada IDE berbasis cloud yang popular.
- **Dukungan database Postgres Heroku**: Kita akan melihat cara mengatur penyimpanan data Heroku, Postgres, dan mengkonfigurasinya untuk digunakan di aplikasi yang paling populer.
- **Mengelola DNS dalam Heroku**: Kami akan memperkenalkan DNS dan kemudian menunjukkan cara menggunakan DNS dengan aplikasi Heroku.

- **Fitur-fitur canggih dari platform Heroku**: Kami akan memperkenalkan 2X dyno yang membantu Anda menambah memori dyno Anda dan mengeksplorasi bagaimana Anda dapat menjaga aplikasi Heroku kecil Anda tetap hidup.

Platform pengembangan One Cloud

Dalam model pengembangan yang dibahas sejauh ini, kami masih perlu menginstal klien Heroku atau perangkat lunak klien lainnya (Ruby atau interpreter Php) untuk membangun aplikasi web kami sebelum mendorongnya ke platform Heroku untuk dieksekusi. Sementara model ini bekerja dengan baik dan lazim, model pengembangan lain yang memegang janji yang sama adalah kembangkan dan gunakan di cloud. Dalam model ini, pengembang menulis kode dalam **integrated development environment** (**IDE**) berbasis browser dan menggunakan alat yang didukung dalam IDE untuk membangun, menyebarkan, dan memecahkan masalah aplikasi web. Di bagian ini, kami memperkenalkan IDE berbasis cloud yang sangat kuat yang disebut Cloud 9 yang terintegrasi secara mulus dengan platform Heroku dan memungkinkan Anda melakukan jenis operasi terkait aplikasi yang sama seperti yang akan Anda lakukan menggunakan Heroku CLI.

Memperkenalkan IDE Cloud 9

IDE Cloud 9 (https://c9.io) adalah salah satu IDE pengembangan berbasis cloud yang menyediakan integrasi dengan Heroku **Platform as a Service (PaaS**). Platform Cloud 9 IDE menyediakan fitur kolaborasi internal yang memungkinkan pengembang menulis, menjalankan, dan men-debug kode aplikasi web di dalam browser web dari mana saja. Seperti platform Heroku, Cloud 9 IDE juga mendukung berbagai bahasa pemrograman di lingkungan IDE: Java, Ruby, JavaScript, dan Node. js untuk beberapa nama.

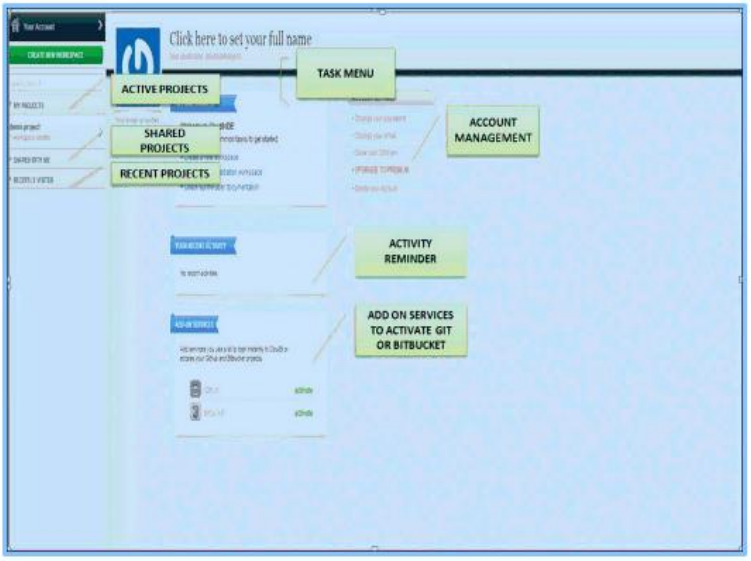
Cloud 9 IDE memberikan akses yang mudah bagi pengembang ke kode menggunakan protokol aman untuk transfer data antara IDE dan platform penyebaran. Dan masih ada lagi; tim dapat berkolaborasi saat menulis aplikasi, mengedit aplikasi, dan mengkonsolidasikan perubahan secara efisien saat diperlukan.

Cloud 9 IDE membuat penulisan kode menjadi transparan bagi pengembang. Pengembang menulis kode dalam lingkungan pengembangan seperti Eclipse, dan mendapatkan fitur yang sama seperti penyelesaian kode, pencocokan Regex, pencarian file, atau tema adaptif di editor online. Semakin membaik. Cloud 9 IDE menawarkan fitur-fitur utama tambahan seperti kode melipat, banyak kursor, fokus bagian kode, dan tarik-dan-lepas, yang sangat meningkatkan produktivitas pengembang. Anda dapat menggunakan baris perintah IDE Cloud 9 bawaan untuk menjalankan berbagai perintah selain menginstal paket yang diperlukan sesuai permintaan. Cloud 9 IDE juga berintegrasi dengan Git – kontrol versi terdistribusi yang paling umum dan platform kolaborasi sosial – di samping mendukung sistem kontrol versi Mercurial.

Pada bagian ini, kita akan secara singkat belajar tentang penggunaan IDE Cloud 9 untuk menyebarkan aplikasi sampel pada platform Heroku. Dengan melakukan itu, kami juga akan meninjau beberapa fitur umum dari Cloud 9 IDE. Untuk membuat akun Anda, kunjungi https://c9.io.

Antarmuka pengguna C9

Dasbor C9 menyediakan fitur lingkungan pengembangan yang kaya. Tangkapan layar berikut mengidentifikasi berbagai bagian dasbor C9 yang ditampilkan saat pengguna masuk ke lingkungan C9:



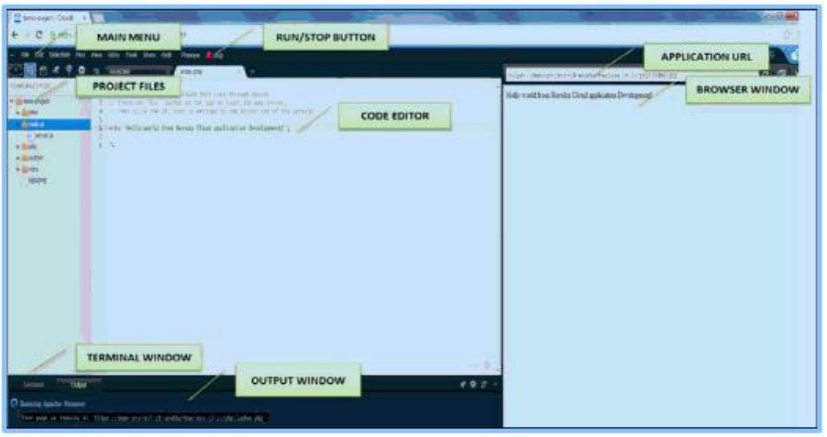
Dasbor C9 terdiri dari komponen-komponen berikut :

- **Task Menu**: Menu ini mencantumkan tindakan atau tugas yang tersedia. Ini berisi tindakan yang paling umum digunakan yang dapat dilakukan pengembang.
- **Account Manajemen**: Menu ini memungkinkan Anda untuk mengelola akun Anda, memperbarui rincian email, atau menghapus akun Anda.
- **Active Projects**: Bagian ini mencantumkan proyek Anda yang sedang aktif. Anda dapat mengklik proyek tertentu untuk mulai mengeditnya.
- **Shared Project**: Bagian ini mencantumkan proyek-proyek yang dikembangkan oleh pengembang bekerja bersama dengan pengguna lain secara kolaboratif.
- **Recent Projects**: Bagian ini berisi daftar beberapa proyek terakhir yang dikerjakan oleh pengembang. Mengklik salah satu proyek ini memungkinkan pengembang mengedit proyek itu.
- **Add on services**: Opsi ini memungkinkan Anda untuk mengkonfigurasi kontrol versi untuk aplikasi cloud Anda. Saat ini, ia mendukung GitHub dan Bitbucket sebagai repositori kode sumber untuk kode aplikasi Anda.
- **Activity reminder**: Opsi ini mencatat aktivitas pengguna dalam urutan kronologis untuk penggunaan pengembang.

Tampilan proyek C9

Tampilan proyek C9 menangkap tampilan dan nuansa sebenarnya dari editor kode dan komponen lain dari lingkungan pengembangan C9. Tampilan proyek C9 terdiri dari menu utama termasuk menjalankan aplikasi dan tombol debug. Selain itu, lingkungan pengembangan C9 sangat fleksibel dan menyediakan jendela terminal untuk mengetik perintah dari CLI, membuatnya sangat mudah untuk beralih dari eksekusi aplikasi berbasis menu ke lingkungan eksekusi berbasis baris perintah. Anda dapat melihat hasil eksekusi Anda di jendela output juga. Untuk aplikasi web, jendela browser di sebelah kanan menampilkan output dari eksekusi skrip atau program Anda. Ini melayani pengembang dengan sangat baik dalam menyoroti masalah yang ada, kemudian membuat perubahan pada aplikasi mereka, dan menjalankannya secara instan.

Tangkapan layar berikut menyoroti bagian-bagian penting dari tampilan C9:



Menyiapkan preferensi di lingkungan IDE C9

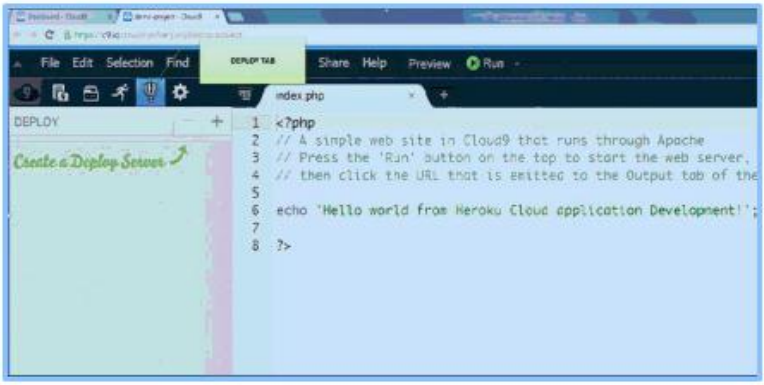
Pengembang dapat mengatur preferensi editor tertentu menggunakan tab Preferensi di menu. Lingkungan pengembangan C9 menyediakan serangkaian kaya preferensi pengguna yang paling umum. Preferensi pengguna yang ditetapkan menjadi aktif dengan efek langsung, sehingga meningkatkan pengalaman pengembang dan produktivitas keseluruhan.

Menyebarkan pada Heroku

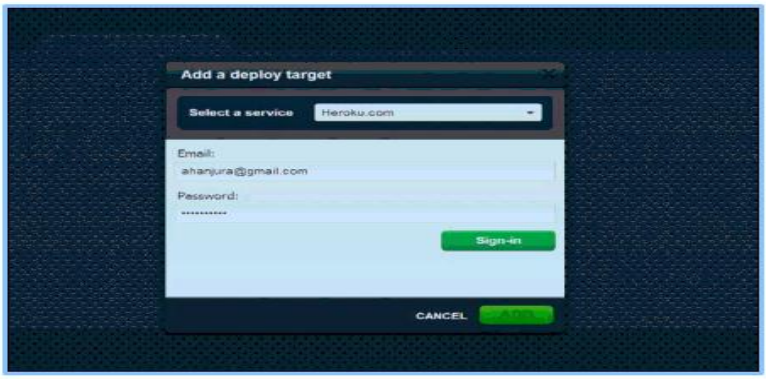
Sekarang kita sudah akrab dengan lingkungan IDE C9 dasar, mari kita menulis aplikasi sederhana dan menyebarkannya ke platform Heroku. Fitur "kode di mana saja, sebarkan di mana saja" ini adalah paradigma kuat yang diaktifkan oleh PaaS seperti Heroku.

Lakukan langkah-langkah berikut untuk menyebarkan aplikasi berbasis Cloud 9 di Heroku :

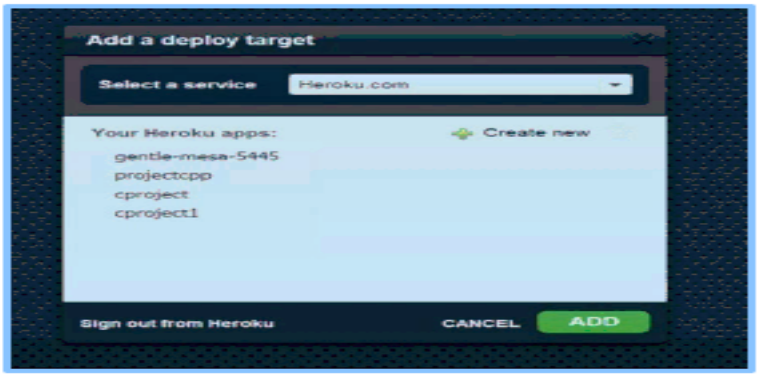
1. Tulis aplikasi web sederhana.
2. Simpan aplikasi web dan klik pada tab Deploy. Ini akan menunjukkan tanda + di bagian sebelah kiri seperti yang ditunjukkan pada tangkapan layar berikut:



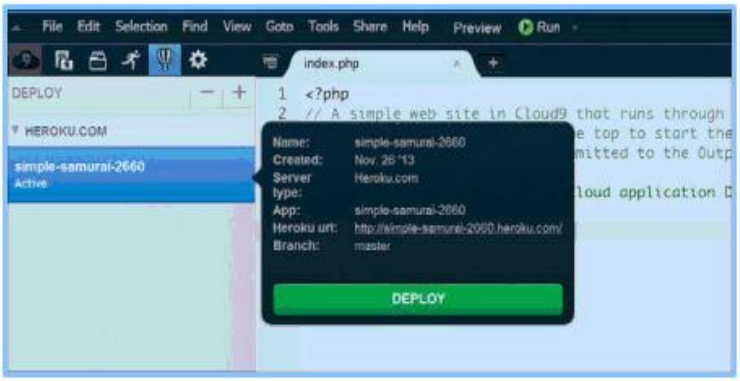
3. Klik pada tanda +. Pilih lingkungan penyebaran target di Tambahkan dialog target penyebaran seperti yang ditunjukkan pada tangkapan layar berikut:



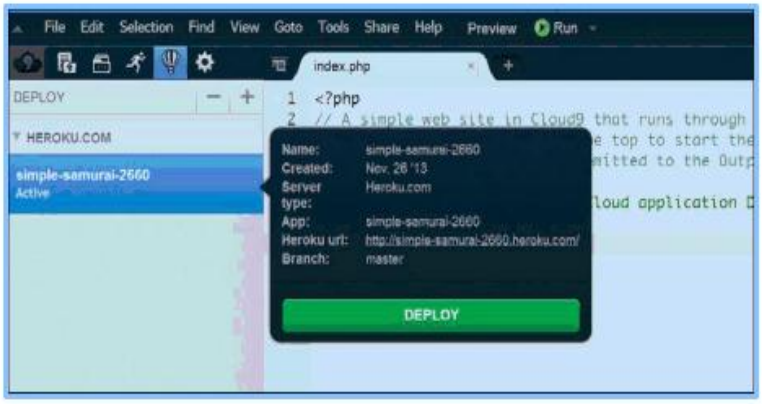
4. Setelah masuk, Anda akan melihat daftar aplikasi yang digunakan dan opsi untuk membuat aplikasi baru. Pilih tombol ADD, seperti yang ditunjukkan pada tangkapan layar berikut:



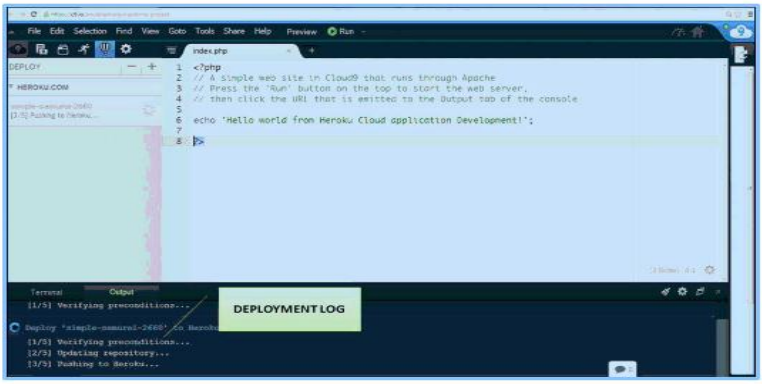
5. Pilih nama aplikasi baru, dalam hal ini, simple-samurai-2660, dan klik ADD. Aplikasi baru akan dibuat untuk Anda seperti yang ditunjukkan pada tangkapan layar berikut:



6. Klik pada nama aplikasi yang baru dibuat. Anda akan melihat pop-up dengan opsi untuk menyebarkan aplikasi. Klik DEPLOY, seperti yang ditunjukkan pada tangkapan layar berikut:



7. Tinjau log penyebaran di jendela Output seperti yang ditunjukkan pada tangkapan layar berikut untuk memeriksa kemajuan penyebaran. Jika tidak ada kesalahan dalam aplikasi Anda, itu akan berhasil digunakan pada platform Heroku:

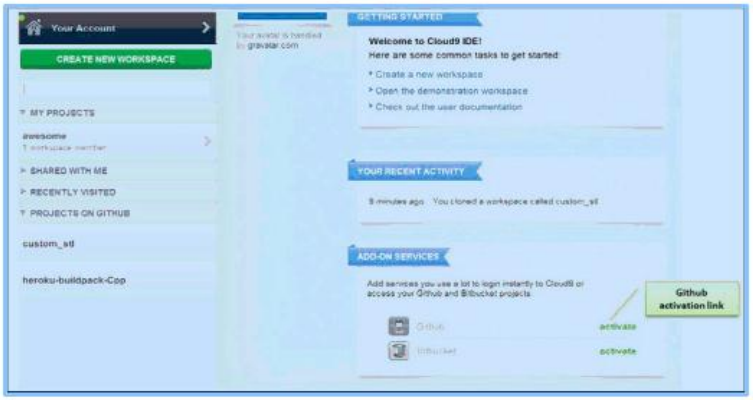


Melakukan operasi Git menggunakan C9 IDE

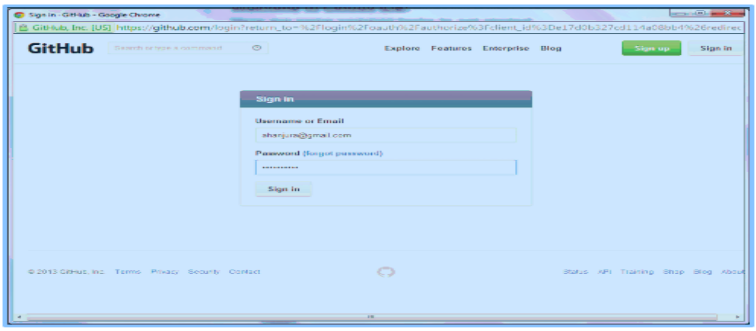
Ruang kerja C9 memungkinkan pengembang untuk berintegrasi mulus dengan repositori GitHub dan Bitbucket. Memeriksa kode dari mereka dan memeriksanya kembali sangatlah mudah.

Sebagai contoh, untuk mengkloning proyek GitHub, ikuti langkah-langkah ini:

1. Pergi ke dasbor C9.



2. Klik pada hyperlink Github di bagian ADD-ON SERVICES as digambarkan dalam tangkapan layar sebelumnya.
3. Masukkan kredensial pengguna dan masuk seperti yang ditunjukkan pada tangkapan layar berikut:



4. Tautan PROYEK PADA GITHUB di bilah navigasi kiri mencantumkan semua proyek GitHub seperti yang ditunjukkan pada tangkapan layar berikut:

5. Untuk mengkloning proyek GitHub, klik salah satu tautan proyek di bilah navigasi kiri seperti yang ditunjukkan pada tangkapan layar berikut:

6. Klik tombol CLONE TO EDIT yang ditunjukkan pada tangkapan layar sebelumnya. Pilihan yang ditunjukkan pada tangkapan layar berikut ini ditampilkan:

7. Klik CREATE dan C9 akan mengkloning proyek GitHub ke ruang kerja aktif seperti yang ditunjukkan pada tangkapan layar berikut:

Lingkungan pengembangan cloud C9 memberikan tingkat fleksibilitas yang tinggi dengan menyediakan integrasi out-of-the-box dengan GitHub dan Bitbucket. Menggunakan proyek Git Anda di C9 sangat intuitif dan semudah mengelola kode menggunakan lingkungan pengembangan lokal. Lingkungan pengembangan C9 menjaga runtime bahasa transparan bagi penggunanya. Kode aman dan dicadangkan secara teratur.

Heroku dan penyimpanan data

Sebagian besar aplikasi perlu menyimpan data – sementara atau permanen. Penyimpanan data ini bisa berbentuk database relasional penuh sesak nafas atau hanya cache yang jauh lebih kecil untuk menyimpan data aplikasi yang paling sering digunakan. Aplikasi Anda mungkin perlu menyimpan struktur data yang kompleks atau pasangan nama / nilai yang lebih sederhana. Setiap

PaaS akan membutuhkan dukungan yang masuk akal untuk kebutuhan penyimpanan data tersebut sehingga aplikasi yang ditulis pada platform dapat dengan mudah berbicara dengan tingkat data dan melakukan operasi CRUD pada data aplikasi yang dihasilkan selama umur aplikasi.

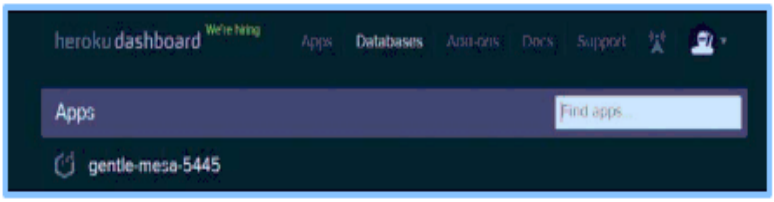
Heroku menawarkan database PostgreSQL sebagai layanan basis data relasional untuk memenuhi kebutuhan akan tier basis data yang kuat, berkinerja tinggi, dapat diskalakan, dan sangat tersedia. Layanan Postgres ditawarkan dan dikelola sebagai tambahan Heroku di platform Heroku. Menggunakan kekuatan penuh Postgres adalah masalah memilih paket yang tepat berdasarkan kompleksitas penempatan Anda. Jalankan beberapa perintah baik di CLI atau melalui dasbor dan voila, Anda bangun dan berjalan dengan layanan Heroku PostgreSQL.

Di bagian ini, kita akan fokus pada bagaimana memulai menggunakan layanan Heroku PostgreSQL. Kami akan memahami cara membuat, mengubah, dan menghapus basis data dan cara membuat replika basis data dan memantau basis data Anda. Layanan Heroku Postgres memberi Anda karakteristik failover dan ketersediaan yang memenuhi kebutuhan aplikasi web Anda yang terus berkembang.

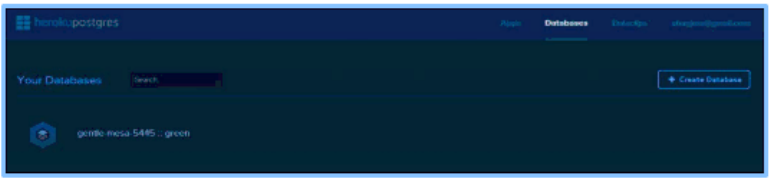
Membuat database Heroku Postgres

Jika Anda menjelajahi opsi menggunakan layanan database di Heroku, lakukan langkah-langkah berikut:

1. Langkah pertama adalah pergi ke http://postgres.heroku.com dan masuk ke situs menggunakan kredensial Heroku Anda:



2. Setelah masuk, klik tautan Databases di menu atas. Anda akan melihat opsi, Buat Database dan daftar database yang saat ini tersedia:



3. Klik Buat Basis Data seperti yang ditunjukkan pada tangkapan layar berikut:

Halaman ini memberi Anda perbandingan konfigurasi database yang tersedia dan harga relatifnya. Ini adalah pilihan paling penting yang akan Anda buat sebelum menggunakan layanan database PostgreSQL Heroku. Anda harus memperhatikan dengan baik opsi yang tersedia dan memilih salah satu yang sesuai dengan kebutuhan aplikasi Anda serta anggaran Anda. Pilihan Anda akan tergantung pada penggunaan memori yang diharapkan dari basis data Anda, berapa banyak penyimpanan fisik yang Anda butuhkan untuk basis data, jumlah koneksi basis data yang dibutuhkan aplikasi Anda untuk perilaku optimal, dan tentu saja berapa banyak Anda bersedia membayar untuk layanan tersebut. Bagi penggemar hobi, menggunakan rencana pengembangan gratis sudah cukup dalam banyak kasus.

Setelah Anda memilih paket yang tepat, membuat database itu mudah. Misalnya, jika Anda memilih opsi **Tengu** untuk aplikasi sampel-db-aplikasi Anda, gunakan perintah berikut untuk

menambahkan konfigurasi **Tengu** Heroku PostgreSQL ke aplikasi Anda :

$ heroku addons:add heroku-postgresql:tengu --app sample-db-app
Adding heroku-postgresql:tengu to sample-db-app... done, v2 ($200/mo)
Attached as HEROKU_POSTGRESQL_XXXXXX
The database should be available in 3-5 minutes
Use 'heroku pg:wait' to track status
heroku-postgresql:tengu documentation available at:
https://devcenter.heroku.com/articles/heroku-postgresql

Anda dapat menjalankan perintah pg: wait setelah itu untuk menunggu pembuatan basis data Anda selesai :

$ heroku pg:wait
Waiting for database ... available

Setelah database dibuat dan tersedia, gunakan baris perintah Heroku untuk memverifikasi detail database sebagai berikut:

$ heroku pg --app sample-db-app
=== HEROKU_POSTGRESQL_XXXXXX
Conn Info:
[Deprecated] Please use 'heroku pg:credentials HEROKU_POSTGRESQL_XXXXXX
to
view connection info
Created: 2013-11-02 01:29 UTC
Data Size: 6.9 MB
Fork/Follow: Temporarily Unavailable
Maintenance: not required
PG Version: 9.x.x
Plan: Tengu
Status: available
Tables: 0

Masuk ke database

Untuk memverifikasi kredensial pengguna basis data, jalankan perintah pg: kredensial sebagai berikut :

```
$ heroku pg:credentials HEROKU_POSTGRESQL_XXXXXX
Connection info string:
"dbname=database_name host=ec2-12-34-567-890.compute-
1.amazonaws.com
port=3421
user=user_name password=database_password
sslmode=require"
```

Untuk mengatur basis data yang diberikan menjadi basis data utama untuk aplikasi Anda, sample-db-app, promosikan basis data yang diberikan menjadi basis data utama menggunakan heroku pg: promosikan perintah sebagai berikut:

```
$ heroku pg:promote
HEROKU_POSTGRESQL_XXXXXX_URL --app sample-db-
app
Promoting HEROKU_POSTGRESQL_XXXXXX_URL to
DATABASE_URL... done
```

Pada promosi database yang sukses, Heroku akan mengatur variabel konfigurasi terkait-database, termasuk DATABASE_URL, untuk menunjuk ke database yang baru dipromosikan.

Untuk meninjau informasi konfigurasi ini, gunakan perintah heroku config CLI dan berikan nama aplikasi sebagai berikut:

```
$ heroku config --app sample-db-app
=== sample-db-app Config Vars
DATABASE_URL:
postgres://user_name :database_password@ec2-12-34-
567-890.
compute-1.amazonaws.com:3421/database_name
HEROKU_POSTGRESQL_XXXXXX_URL:
postgres://user_name :database_password@ec2-12-34-
567-890.
compute-1.amazonaws.com:3421/database_name
```

Itu dia. Anda telah berhasil membuat, memverifikasi, dan meninjau layanan Heroku Postgres untuk aplikasi Anda. Anda juga dapat melihat database yang dibuat di konsol web Heroku Postgres.

Heroku menyediakan fasilitas untuk mengakses terminal interaktif Postgres ke database melalui SSL melalui CLI.

Untuk terhubung ke layanan database Postgres, gunakan perintah berikut:

```
$ heroku pg:psql HEROKU_POSTGRESQL_XXXXXX
psql (9.x.x)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
database_name=>
NOTE Using the psql terminal requires a local installation of
the
PostgreSQL
(libpq) client. For platform-specific instructions for
installing
Postgres, see
https://devcenter.heroku.com/articles/local-postgresql
```

Heroku menyediakan serangkaian operasi basis data yang menarik yang dapat digunakan untuk memenuhi kebutuhan penyimpanan data yang khas. Berapa kali Anda ingin memiliki database pengujian yang dibuat untuk Anda mirip dengan database produksi pelanggan atau melakukan pengujian beban pada berbagai konfigurasi database atau bereksperimen dengan perubahan skema sebelum membukanya ke set pengguna yang lebih luas? Heroku PostgreSQL punya jawaban untuk ini juga. Jawabannya adalah garpu basis data. Heroku memberi Anda kemampuan untuk melakukan fork database pilihan Anda dan membuat replika yang tepat. Namun, Heroku membatasi kemampuan untuk melakukan fork basis data hanya untuk paket tingkat produksi berbayar dan tidak untuk paket pengembang gratis.

Membuat lebih banyak basis data – garpu

Kadang-kadang, Anda mungkin memerlukan snapshot dari database yang ada, yaitu, Anda akan membutuhkan potongan basis data yang berisi semua data dari database yang diberikan pada titik waktu tertentu. Contoh umum adalah ketika Anda ingin menguji aplikasi Anda terhadap basis data pelanggan untuk mengisolasi dan memecahkan masalah. Anda mungkin ingin pelanggan memberikan snapshot data pada titik waktu tertentu. Di sinilah garpu database masuk dan memberi Anda kemampuan untuk membuat database dari snapshot dari database lain. Ada juga kasus penggunaan lainnya; misalnya, tim operasi mungkin ingin menguji migrasi basis data pada salinan produksi sebelum melakukan migrasi aktual, atau personel keamanan data mungkin ingin menyimpan salinan data produksi Anda pada suatu titik waktu untuk mengamankan data pelanggan yang penting kalau-kalau ada yang salah dengan database produksi. Semua contoh ini memberikan kasus yang menarik untuk sebuah utilitas yang dapat membuat snapshot dari database. Opsi garpu basis data tidak hanya itu.

Anda dapat membuat garpu dari database aplikasi Anda menggunakan parameter --fork command-line seperti yang ditunjukkan pada set perintah berikut :

```
$ heroku addons:add heroku-postgresql:tengu --fork
HEROKU_POSTGRESQL_
XXXXXX
Adding heroku-postgresql:tengu to test-db-app... done, v...
($200/mo)
Attached as HEROKU_POSTGRESQL_YYYYYY
Database will become available after it completes forking
Use 'heroku pg:wait' to track status
heroku-postgresql:tengu documentation available at:
https://devcenter.heroku.com/articles/heroku-postgresql
```

Menyinkronkan basis data melalui basis data pengikut

Saat ini, organisasi memiliki sejumlah besar data yang dihasilkan oleh berbagai aplikasi perangkat lunak. Ukuran dan kerumitan basis data produksi tampaknya telah berlipat ganda secara eksponensial. Penempatan basis data menjadi lebih rumit.

Dalam penyebaran yang lebih besar, basis data yang berbeda digunakan untuk menangani permintaan dari aplikasi yang hanya membaca, dibandingkan aplikasi yang terus memperbarui data juga. Organisasi tidak lagi mampu membiarkan semua orang meminta basis data produksi bahkan jika itu hanya untuk membaca beberapa catatan dari tabel besar. Untuk memenuhi kebutuhan spesifik dari permintaan kueri terdistribusi pada basis data yang cukup besar, Heroku mendukung konsep pengikut basis data. Jika bukan karena pengikut basis data ini, keseluruhan respons sistem basis data Anda mungkin melambat secara signifikan dan menyebabkan keterlambatan yang tidak perlu dalam melayani permintaan.

Pengikut basis data adalah salinan hanya baca dari sumber basis data yang diperbarui dengan perubahan terbaru ke sumber dalam waktu dekat. Pengikut basis data memenuhi kebutuhan sebagian besar aplikasi yang perlu menjalankan kueri baca-saja yang berjalan lama atau membuat laporan berdasarkan data yang disimpan di berbagai tabel.

Untuk membuat pengikut basis data, kita dapat menggunakan opsi --follow perintah-baris sebagai berikut:

```
$ heroku addons:add heroku-postgresql:tengu --follow
HEROKU_POSTGRESQL_
XXXXXX
Adding heroku-postgresql:tengu to test-db-app... done, v...
($200/mo)
Attached as HEROKU_POSTGRESQL_YYYYYY
Follower will become available for read-only queries when up-
to-date
Use 'heroku pg:wait' to track status
heroku-postgresql:tengu documentation available at:
https://devcenter.heroku.com/articles/heroku-postgresql
```

Jika kita ingin menggunakan kembali basis data pengikut, kita dapat menjalankan perintah unfollow yang akan mengonversi basis data pengikut hanya baca menjadi salinan baca / tulis. Ini, dengan cara, setara dengan forking database asli pada saat itu.

Untuk berhenti mengikuti basis data aplikasi yang diberikan, jalankan perintah berikut:

$ heroku pg:unfollow HEROKU_POSTGRESQL_YYYYYY
! HEROKU_POSTGRESQL_YYYYYY will become writable and no longer
! follow HEROKU_POSTGRESQL_XXXXXX. This cannot be undone.
! WARNING: Destructive Action
! This command will affect the app: test-db-app
! To proceed, type "test-db-app" or re-run this command with --confirm
test-db-app
> test-db-app
Unfollowing heroku_postgresql_YYYYYY... done

Heroku baru-baru ini mulai memberi pengguna kemampuan untuk membatalkan permintaan yang tampaknya berjalan untuk waktu yang tidak wajar dan menghabiskan banyak CPU, memori, atau sumber daya lainnya. Sebelumnya, membatalkan permintaan seperti itu akan membutuhkan pengguna untuk menghubungi dukungan Heroku yang kemudian akan membatalkan permintaan.

Untuk mencoba membatalkan permintaan yang sudah berjalan lama, mulailah dua sesi terminal. Di terminal pertama, masuk ke layanan Postgres dan jalankan kueri yang berpotensi lambat, misalnya, pilih * dari proses; (di mana proses adalah tabel yang sudah ada yang berisi jutaan catatan):

$ heroku pg:psql HEROKU_POSTGRESQL_XXXXXX
psql (9.x.x)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
database_name=> select * from processes;

Di terminal kedua, gunakan langkah-langkah berikut untuk membatalkan permintaan yang berpotensi lambat:

$ heroku pg:psql HEROKU_POSTGRESQL_XXXXXX
psql (9.x.x)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

```
database_name=> SELECT pg_cancel_backend(procpid)
FROM pg_stat_activity
WHERE
current_query LIKE '%processes%';
ERROR: canceling statement due to user request
```

Terminal pertama juga akan menampilkan pesan yang sama, menunjukkan bahwa permintaan yang lambat dibatalkan.

Salah satu kebutuhan paling penting bagi setiap pengguna basis data, terutama yang mengelola basis data, adalah kemampuan untuk memonitor basis data dan operasi yang berjalan pada basis data. Salah satu cara yang jelas untuk mengetahui apa yang dilakukan database adalah dengan melihat log basis data. Di Heroku, log Heroku Postgres berada di lokasi yang sama dengan log aplikasi.

Memeriksa log basis data

Untuk melihat pesan log khusus database, gunakan perintah heroku log dan berikan filter postgres ke perintah sebagai berikut:

```
$ heroku logs --ps postgres
2013-11-05T03:45:12+00:00 app[postgres]: [17-1] [XXXXXX]
LOG:
checkpoint starting: time
2013-11-05T03:45:12+00:00 app[postgres]: [18-1] [XXXXXX]
LOG:
checkpoint complete: wrote 0 buffers (0.0%); 0 transaction log
file(s)
added,
0 removed, 0 recycled; write=0.000 s, sync=0.000 s, total=0.231
s; sync
files=0,
longest=0.000 s, average=0.000 s
[...]
```

Kinerja dan Postoku Heroku basis data

Untuk mengukur rasio hit-cache database Anda untuk tabel, masuk ke psql dan jalankan kueri sebagai berikut:

```
$ heroku pg:psql HEROKU_POSTGRESQL_XXXXXX
psql (9.x.x)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
database_name=> SELECT
database_name-> sum(heap_blks_read) as heap_read,
database_name-> sum(heap_blks_hit) as heap_hit,
database_name->  (sum(heap_blks_hit) -
sum(heap_blks_read)) /
sum(heap_blks_hit) as ratio
database_name--> FROM pg_statio_user_tables;
heap_read | heap_hit | ratio
----------+----------+----------------------62351
 | 81208 | 0.232206186582602
(1 row)
```

Hasilnya menunjukkan bahwa rasio hit-cache hanya 23 persen, yang berarti ukuran cache yang lebih besar dapat meningkatkan kinerja kueri basis data.

Permintaan serupa untuk mengukur rasio hit-cache untuk indeks adalah sebagai berikut:

```
database-name=> SELECT
database-name-> sum(idx_blks_read) as idx_read,
database-name-> sum(idx_blks_hit) as idx_hit,
database-name-> (sum(idx_blks_hit) - sum(idx_blks_read)) /
sum(idx_blks_hit) as ratio
database-name-> FROM
database-name-> pg_statio_user_indexes;
idx_read | idx_hit | ratio
---------+---------+----------------------38
 | 78 | 0.512820512820
(1 row)
```

Di sini, rasio hit-cache hanya 51 persen untuk indeks Anda juga menunjukkan bahwa Anda bisa mendapatkan rasio hit yang lebih baik jika Anda menggunakan cache database yang lebih besar.

Pemulihan bencana di Heroku PostgreSQL

Cara termudah untuk memulihkan data Anda adalah dengan sering membuat cadangan. Heroku menyediakan mekanisme yang sangat mudah untuk mencadangkan data Anda melalui add-on pgbackups gratis. Gunakan add-on ini untuk membuat cadangan data sesuai permintaan.

Untuk membuat cadangan, Anda harus terlebih dahulu menambahkan add-on pgbackups ke akun Anda. Untuk melakukannya, gunakan perintah berikut:

```
$ heroku addons:add pgbackups
Adding pgbackups to test-db-app... done, v... (free)
You can now use "pgbackups" to backup your databases or import an
external backup.
pgbackups documentation available at:
https://devcenter.heroku.com/articles/pgbackups
```

Ada beberapa opsi yang tersedia dalam bentuk paket cadangan yang dapat Anda gunakan berdasarkan kebutuhan bisnis Anda. Keputusan Anda untuk menggunakan rencana tertentu mungkin tergantung pada tingkat otomatisasi yang dibutuhkan, jumlah ekspor yang diperlukan, dan periode penyimpanan cadangan. Secara default, cadangan otomatis dilakukan pada basis data yang ditunjukkan oleh parameter konfigurasi DATABASE_URL.

Untuk membuat cadangan manual, gunakan perintah berikut:

```
$ heroku pgbackups:capture HEROKU_POSTGRESQL_XXXXXX
HEROKU_POSTGRESQL_XXXXXX ----backup---> b001
Capturing... done
Storing... done
```

Gunakan perintah --expirecommand untuk memutar cadangan Anda dengan secara otomatis menghapus cadangan yang lama sebelum membuat yang baru.
Untuk melihat daftar cadangan, gunakan perintah berikut:

```
$ heroku pgbackups
ID | Backup Time | Size | Database
-----+------------------+-------+----------------------
b001| 2013/11/01 16:45.23 | 1.6KB |
HEROKU_POSTGRESQL_XXXXXX
b001| 2013/11/01 16:48.05 | 1.6KB |
HEROKU_POSTGRESQL_XXXXXX
```

Huruf b dalam ID menunjukkan bahwa itu adalah cadangan manual. Jika itu yang otomatis, yang akan diawali dengan ID sebagai gantinya.

Jika Anda perlu mengakses salah satu dari database ini, Anda dapat mengekspos file cadangan Anda melalui URL yang tersedia untuk umum sebagai berikut:

```
$ heroku pgbackups:url b001
https://s3.amazonaws.com/hkpgbackups/smp405893@heroku.com/
b001.dump?AWSAccessKeyId=<...>&Expires=1234567890&Signature=tMuN1n65T
1gaNuEdR3
```

Mengakses URL ini (URL hanya aktif selama 10 menit) memungkinkan Anda untuk mengunduh .dumpfile yang berisi dump Postgres standar dari database Anda, cocok untuk mengimpor ke database Postoku Heroku atau non-Heroku Postgres.

Untuk menghancurkan cadangan lama, gunakan perintah berikut:

```
$ heroku pgbackups:destroy b001
Destroying b001... done
$ heroku pgbackups
ID | Backup Time | Size | Database
-----+------------------+-------+----------------------
```

b001| 2013/11/01 16:48.05 | 1.6KB | HEROKU_POSTGRESQL_XXXXXX

Untuk mengembalikan cadangan dari versi sebelumnya yang disimpan oleh pgbackups, gunakan perintah restore sebagai berikut:

$ heroku pgbackups:restore HEROKU_POSTGRESQL_XXXXXX b001
HEROKU_POSTGRESQL_XXXXXX <---restore--- b001
HEROKU_POSTGRESQL_XXXXXX
2013/11/01 16:48.05
1.6KB
! WARNING: Destructive Action
! This command will affect the app: test-db-app
! To proceed, type "test-db-app" or re-run this command with --confirm test-db-app
> test-db-app
Retrieving... done
Restoring... done

Beberapa pengamatan tentang cadangan adalah sebagai berikut:

- Cadangan yang sangat besar dibagi menjadi beberapa URL. Gunakan utilitas penggabungan file seperti perintah UNIX cat untuk menggabungkannya.
- Ketika Anda mengembalikan cadangan database, itu menimpa database saat ini dengan struktur dan konten cadangan sebelumnya. Oleh karena itu, merupakan ide bagus untuk membuat cadangan dari basis data saat ini sebelum menimpanya melalui pemulihan.
- Jumlah cadangan yang tersedia ditentukan oleh add-on pgbackup yang Anda pilih. Anda mungkin perlu menghapus cadangan lama Anda untuk membuat yang baru.

Mengimpor data ke Postgres

Seringkali, Anda mungkin perlu mengimpor data dari lingkungan pelanggan untuk mereproduksi masalah atau memigrasikan data dari satu database ke yang lain, atau pindah ke Heroku Postgres dari beberapa database lain. Kasing semacam itu membutuhkan mekanisme untuk mengimpor data ke Heroku Postgres. Anda dapat mengimpor format dump yang kompatibel dengan Heroku Postgres ke Postgres, atau menggunakan alat yang tersedia untuk secara langsung memuat database Postgres Anda dari sumbernya. Dalam banyak kasus, Anda perlu mengekspor data dari basis data Anda saat ini sebelum dapat diimpor ke tempat lain. Praktik yang baik saat mengekspor adalah untuk memastikan bahwa file dump dikompresi dan bahwa tidak ada hak akses dan informasi ruang lingkup kepemilikan disimpan di dalam data yang diekspor.

Ada dua kemungkinan untuk mengimpor data ke dalam basis data Heroku Postgres:

- Database sumber adalah database Postgres: Dalam hal ini, Anda harus terlebih dahulu membuat dump Postgres dari database sumber sebelum dapat diimpor ke database Postgres tujuan. Untuk membuat dump dari database Postgres sumber Anda, gunakan perintah berikut:

  $ pg_dump -Fc --no-acl --no-owner old_pg_db_name > old_pg_db.dump
- Basis data sumber adalah basis data non-Postgres: Ketika basis data sumber adalah non-Postgres, Anda perlu mengonversi basis data ke format dump Postgres untuk memungkinkannya diimpor ke basis data Postgres.

Jika Anda bermigrasi jauh dari MySQL, Anda dapat menggunakan alat mysql2psql. Ini adalah alat open source untuk mengonversi database dan dapat memuat data langsung dari MySQL ke database Heroku Postgres.

Ada beberapa cara mengimpor data ke Heroku Postgres:

- Melalui fungsi mengembalikan pgbackups
- Menggunakan alat baris perintah Postgres, pg_restore

Mengimpor data dengan file pgbackups mengharuskan file dump database tersedia melalui URL publik.

Untuk mengimpor dump basis data dari URL yang tersedia untuk umum, gunakan pgbackups: restore perintah sebagai berikut:

```
$ heroku pgbackups:restore HEROKU_POSTGRESQL_
XXXXXX
'https://myherokudump.s3.amazonaws.com/processes.dum
p'
HEROKU_POSTGRESQL_XXXXXX <---restore---
processes.dump
! WARNING: Destructive Action
! This command will affect the app: test-db-app
! To proceed, type "test-db-app" or re-run this command
with --confirm
test-db-app
> test-db-app
Retrieving... done
Restoring... done
ahan-ltr2:test-db-app ahan
```

Setelah dipulihkan, periksa apakah data yang diimpor sudah benar dengan masuk ke Heroku Postgres dan verifikasi beberapa tabel dan datanya:

```
$ heroku pg:psql HEROKU_POSTGRESQL_XXXXXX
psql (9.X.X)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
database_name=> select * from processes limit 2;
id | start_date | proc_name | proc_desc | proc_type |
end_date
--------+------------+---------------+-----------------+---
-----+----
10001| 2013-09-02 | Hiring | HRMSW | A | 2013-09-23
10002| 2013-10-09 | LoanMgmt | BOALM | S | 2013-10-21
```

Untuk mengimpor langsung dari dump database di sistem file lokal Anda, gunakan perintah pg_restore sebagai berikut:

```
$ pg_restore --verbose --clean --no-acl --no-owner
-h ec2-12-34-567-890.compute-1.amazonaws.com -U
user_name -d database_
name
-p 3421 ~/Downloads/processes.dump
```

Anda dapat menemukan tentang opsi baris perintah lainnya dari http://www.postgresql.org/docs/9.2/static/app-pgrestore.html.

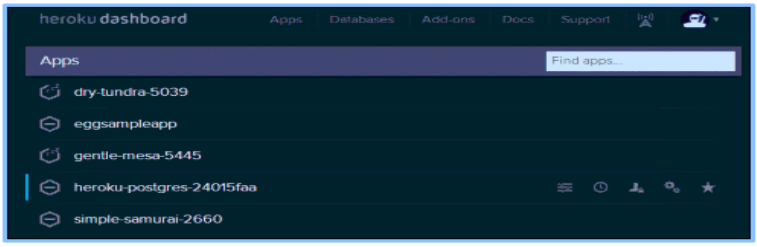
Menghapus basis data Heroku Postgres

Meskipun jarang, kadang-kadang Anda mungkin ingin menghapus database itu sendiri. Ini dapat dicapai dengan dua cara berikut:

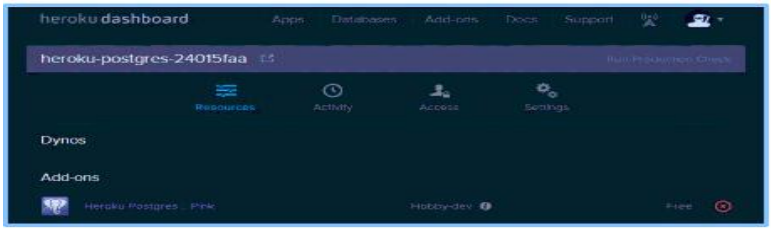
- Dengan menggunakan addons: hapus perintah sebagai berikut:
  - **$ heroku addons:remove HEROKU_POSTGRESQL_ YYYYYY**
    ! WARNING: Destructive Action
    ! This command will affect the app: test-db-app
    ! To proceed, type "test-db-app" or re-run this command with
    --confirm test-db-app
    > test-db-app
    Removing HEROKU_POSTGRESQL_YYYYYY from test-db-app... done, v...
    ($700/mo)
- Untuk cukup mereset database ke bentuk awal kosong tanpa menghancurkannya sepenuhnya, gunakan perintah berikut:
  $ heroku pg:reset HEROKU_POSTGRESQL_XXXXXX
  ! WARNING: Destructive Action
  ! This command will affect the app: test-db-app
  ! To proceed, type "test-db-app" or re-run this command with
  --confirm test-db-app
  > test-db-app

Resetting HEROKU_POSTGRESQL_XXXXXX... done
Ini akan menghapus semua data Anda dan menghapus struktur apa pun yang telah ditetapkan sebelumnya.

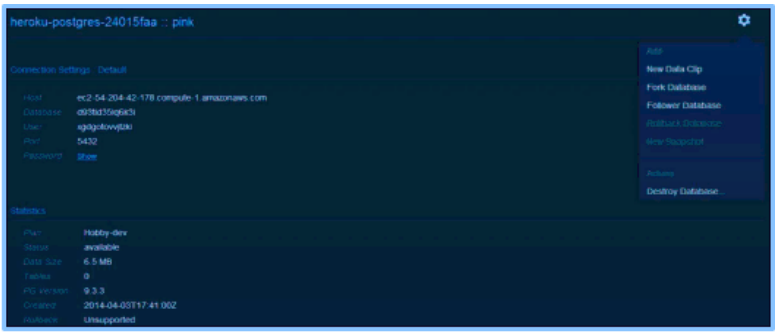
- Dengan menggunakan dasbor Heroku seperti yang ditunjukkan pada tangkapan layar berikut. Lakukan tugas-tugas berikut:
- Masuk ke dasbor Heroku dan pilih aplikasi yang databasenya perlu dihapus. Kami akan menghapus database aplikasi **heroku-postgre-24015faa**. Klik pada aplikasi **herokupostgre-24015faa** yang diperlihatkan dalam tangkapan layar berikut:



- Ini akan mengarahkan pengguna ke halaman detail aplikasi. Perhatikan bahwa ada paket database Heroku **Postgres :: Pink** yang terkait dengan aplikasi seperti yang ditunjukkan pada tangkapan layar berikut:

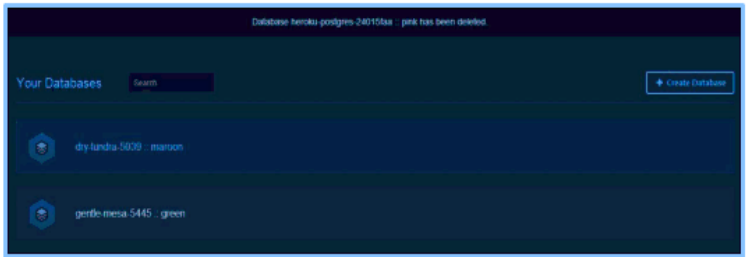


- Klik pada paket basis data (Heroku **Postgres :: Pink**). Ini akan mengarahkan pengguna ke halaman detail basis data seperti yang ditunjukkan pada tangkapan layar berikut. Perhatikan bahwa daftar menu di kanan atas memiliki opsi **Hancurkan Database**. Klik pada opsi menu **Hancurkan Database**:



- Konfirmasikan operasi penghancuran basis data dengan memasukkan **DESTROY** di area teks dan mengklik tombol **DESTROY**, seperti yang ditunjukkan pada tangkapan layar berikut:

- Database Heroku **Postgres :: Pink** aplikasi **heroku-postgre-24015faa** berhasil dihapus seperti yang ditunjukkan pada tangkapan layar berikut:



Jadi, kami mempelajari beberapa cara untuk menghapus database dari aplikasi dan menghapus database sekaligus dari akun Heroku. Sekarang, mari kita lihat bagaimana cara mengakses layanan database Heroku PostgreSQL secara eksternal dari lingkungan non-Heroku.

Mengakses Heroku Postgres secara eksternal

Database Heroku Postgres dikaitkan dengan aplikasi Heroku. Biasanya Anda mengkonfigurasi database Postgres untuk aplikasi Heroku. Heroku secara internal mengelola URL basis data melalui variabel konfigurasi yang disebut DATABASE_URL yang berisi rincian basis data termasuk lokasi dan kredensial.

Karena kredensial database dapat berubah secara dinamis (karena Heroku mengelola parameter konfigurasi basis data secara internal), disarankan untuk tidak menyalin kredensial sebagaimana lingkungan lain atau di dalam kode aplikasi Anda.

Variabel DATABASE_URL, yang dikelola oleh Heroku, dapat berubah dalam kondisi berikut:

- **Kegagalan basis data yang tidak dapat dipulihkan**: Jika perangkat keras yang menjalankan basis data Anda gagal, maka basis data Anda mungkin dipulihkan ke instance basis data yang berbeda dengan kredensial basis data yang sama.
- **Pertimbangan keamanan**: Pengguna dapat meminta perubahan kredensial menggunakan perintah pg: credentials – reset.
- **Persyaratan ketersediaan**: Kegagalan basis data dapat terjadi secara otomatis untuk paket Postgres yang diaktifkan HA setiap kali Heroku menentukan situasi kegagalan dan beralih ke siaga.

Mengakses kredensial basis data

Praktik terbaik adalah mengambil variabel konfigurasi ini dari aplikasi Heroku terkait ketika aplikasi dimulai. Anda harus menjalankan proses Anda sebagai berikut.

DATABASE_URL=$(heroku config:get DATABASE_URL -a app_name) process_name

Menghubungkan dari luar Heroku

Jika Anda ingin mengakses database Heroku Postgres Anda dari luar lingkungan Heroku, klien atau aplikasi Anda harus mendukung SSL untuk terhubung ke database Heroku PostgreSQL.

Untuk memastikan bahwa Anda selalu secara paksa memaksa koneksi SSL dari aplikasi Anda alih-alih mengandalkan perilaku default dari klien atau aplikasi, Anda harus menggunakan sslmode = require parameter pada string koneksi Heroku PostgreSQL.

Contoh untuk menghubungkan ke Heroku PostgreSQL menggunakan parameter koneksi sslmode ditunjukkan pada perintah berikut:

$ psql "host=<hostname or IP> port=<port no> user=<database user> password=<database password> sslmode=require"

Postgres ketersediaan tinggi

Heroku Postgres menawarkan rencana tingkat yang berbeda untuk pelanggan berdasarkan tingkat fungsionalitas dan ketahanan basis data yang diinginkan, yaitu, failover dan ketersediaan tinggi. Jika kebutuhan bisnis Anda tipikal dari aplikasi perusahaan atau sistem terdistribusi besar di mana kegagalan dan downtime diharapkan tetapi dikelola, Anda harus mempertimbangkan rencana premium atau tingkat perusahaan.

Memilih rencana yang tepat

Rencana perusahaan dan premium-tier dibundel dengan fitur-fitur canggih ketersediaan tinggi untuk database PostgreSQL. Artinya adalah jika terjadi kegagalan basis data yang tidak terduga, perangkat keras atau lainnya, basis data aplikasi Anda akan beralih ke basis data siaga yang berisi data yang hampir sama dengan basis data utama Anda. Peralihan mengurangi kemungkinan waktu henti yang lebih lama dan pada gilirannya menghindari respons aplikasi yang buruk atau tidak ada.

Fitur ketersediaan tinggi (**HA**) melibatkan pembuatan database siaga untuk database aplikasi utama Anda yang dapat digunakan jika terjadi kegagalan database yang tidak terduga. Siaga ini dikelola oleh staf Heroku PostgreSQL, dan biasanya

dikonfigurasikan sedekat mungkin ke basis data primer melalui msyncs data terjadwal dengan basis data primer. Ketika kegagalan basis data terjadi, adalah mungkin untuk jumlah kecil, tetapi dalam batas, jumlah data yang paling baru berkomitmen untuk hilang. Ada ambang yang dipilih sebagai delta perbedaan data yang dapat diterima antara database primer dan sekunder untuk digunakan sebagai siaga jika terjadi kegagalan.

Idealnya, database siaga harus merupakan replika tepat dari database primer. Hanya dengan demikian keadaan aplikasi dapat dipulihkan seperti sebelum kegagalan. Namun, dalam situasi praktis, ada kemungkinan kehilangan data kecil namun dapat diabaikan selama failover. Untuk meminimalkan kehilangan ini, Heroku Postgres mengambil beberapa langkah untuk menentukan apa yang dapat diterima. Misalnya, tidak mencoba failover ke siaga jika lebih dari 10 segmen basis data di belakang. Heroku juga mencoba menerapkan segmen yang diarsipkan tetapi belum diterapkan sebelum mengeluarkan modus siaga dari siap-saja.

Dalam kasus rencana basis data HA, database siaga dikonfigurasikan dalam zona ketersediaan yang berbeda (**AZ**) dari basis data primer untuk mencegah kehilangan data jika terjadi kegagalan seluruh AZ. Contoh database yang gagal dihancurkan dan siaga dibangun kembali. Karena database siaga tidak dapat dilihat langsung oleh aplikasi, Anda perlu membuat database pengikut untuk database baru jika Anda membutuhkan pengikut agar database utama Anda berfungsi setelah peralihan.

Efek samping lain dari kegagalan basis data adalah bahwa nilai variabel konfigurasi URL DATABASE_URL dan HEROKU POSTGRES diubah secara dinamis. Oleh karena itu, disarankan untuk tidak menggunakan nilai-nilai hardcod dari parameter konfigurasi ini di dalam kode aplikasi atau menggunakan nilai-nilai dalam aplikasi luar. Cara yang tepat untuk menggunakan variabel konfigurasi ini dibahas di bagian *Mengakses kredensial database.*

Kapan Heroku Postgres gagal?

Kegagalan basis data atau respons yang lambat tidak segera memicu kegagalan basis data di layanan basis data Heroku PostgreSQL. Dukungan Heroku PostgreSQL memastikan bahwa

database aplikasi benar-benar memerlukan failover (karena melibatkan biaya untuk beralih ke instance database baru dan mematikan yang sebelumnya), selain mengatur konfigurasi baru dan membuat aplikasi hidup kembali pada database baru. Dukungan Heroku menjalankan serangkaian tes untuk memastikan bahwa failover adalah satu-satunya pilihan. Tes dijalankan untuk periode waktu tertentu, biasanya dua menit, di berbagai aplikasi yang berbeda untuk melihat konsistensi dalam perilaku respons basis data. Hanya ketika terdeteksi bahwa database tidak responsif untuk semua permintaan untuk periode ini, maka failover dapat dimulai.

Efek dari failover

Ketika terjadi kegagalan basis data, aplikasi Heroku dipengaruhi dalam beberapa cara: URL DATABASE dan variabel URL HEROKU POSTGRES berubah dalam nilainya dan upaya dilakukan untuk me-restart aplikasi dengan kredensial basis data baru. Kinerja aplikasi mungkin akan terpengaruh untuk sementara waktu karena cache database yang dingin yang di-refresh dan diisi ulang selama periode waktu tertentu. Database siaga baru secara otomatis dibuat untuk database baru dan tersedia sedemikian rupa sehingga siaga memenuhi persyaratan yang diperlukan agar kegagalan dapat berhasil.

Memeriksa status ketersediaan setelah kegagalan

Setelah failover selesai, Anda dapat memeriksa status ketersediaan database yang baru dibuat dengan mengeluarkan heroku pg: info perintah sebagai berikut:

$ heroku pg:info

...

HA status: Available

...

Nilai bidang status HA yang Tersedia menunjukkan bahwa basis data siap digunakan. Ketika database sedang dalam persiapan atau belum disinkronkan untuk failover, bidang status HA akan menunjukkan status Sementara tidak tersedia.

Mengkonfigurasi domain dengan cara yang benar

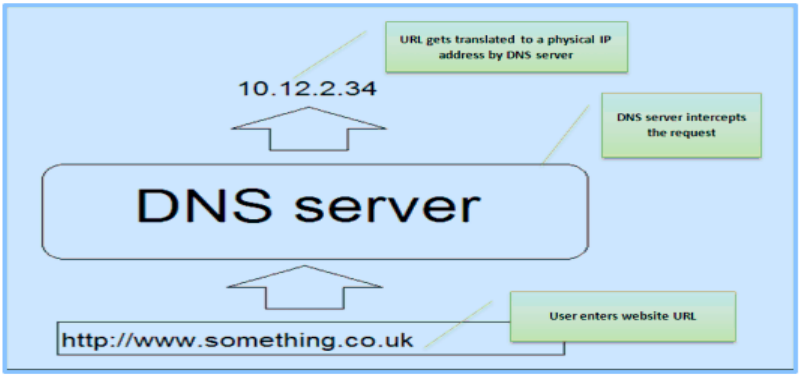
Ketika kami menyebarkan aplikasi web ke Heroku, aplikasi dapat diakses melalui subdomain aplikasi herokuapp.com (di cedar stack). Misalnya, jika aplikasi Anda bernama myshoppingcart, URL Heroku yang sepenuhnya memenuhi syarat untuk aplikasi tersebut adalah http://myshoppingcart.herokuapp.com. Jika Anda adalah situs e-commerce yang ingin menarik pengguna baru, Anda mungkin ingin menggunakan URL web yang lebih khusus untuk situs tersebut misalnya, www.buyshoes.com atau www.someecomsite.com. Kita dapat mencapai ini di Heroku dengan mengkonfigurasi aplikasi untuk menggunakan domain khusus. Heroku menyediakan kemampuan untuk memetakan satu atau beberapa domain khusus ke aplikasi apa pun secara transparan. Kami mulai dengan mendapatkan pengantar kecil ke sistem nama domain atau DNS yang merupakan teknologi dasar yang digunakan untuk mengkonfigurasi domain.

Ikhtisar DNS

Internet adalah jaringan besar komputer, masing-masing diidentifikasi oleh alamat IP seperti 1.2.3.4. Ketika pengguna meminta sumber daya (halaman, gambar), mereka memasukkan URL yang mudah diingat yang diterjemahkan ke alamat IP fisik oleh layanan Internet. Layanan Internet yang membantu menyelesaikan URL yang disediakan pengguna dan mudah diingat ke IP host yang benar disebut **Domain Name Sistem** atau **DNS**.

Ketika pengguna meminta sumber daya, sumber daya diidentifikasi oleh URI tertentu (biasanya disebut URL). Bagian dari URL adalah nama domain, yang dapat disederhanakan sebagai alternatif tekstual yang mudah diingat ke alamat IP mesin di mana sumber daya di-host.

Saat Anda meminta URI, sistem DNS memungkinkan, melalui resolver dan server DNS, untuk menyelesaikan permintaan Anda ke dalam IP yang sesuai dengan bagian nama host dari URI.



Dalam URL web seperti www.google.com, bagian com adalah apa yang kami wakili sebagai **top-level domain** (**TLD**). Domain tingkat atas mengidentifikasi domain milik negara atau kategori tertentu. Misalnya, .gov milik pemerintah AS dan .edu mewakili sebagian besar lembaga pendidikan Amerika.

Setiap kombinasi kata dan titik dalam nama domain sebelum domain tingkat atas menunjukkan tingkat dalam struktur domain. Setiap level mengacu pada server atau sekelompok server yang mengelola level domain itu. Misalnya, heroku di www.heroku.com adalah domain tingkat kedua dari domain tingkat atas com. Organisasi selanjutnya dapat menggunakan konsep **subdomain** untuk mengatur keberadaan web mereka, seperti Citibank.co.in, yang merupakan domain Citibank di bawah CO, tingkat tambahan yang dibuat oleh otoritas nama domain yang bertanggung jawab atas kode negara **IN** (**India**). Domain dapat berisi sejumlah besar mesin, dan karena semua nama dalam domain yang diberikan harus unik, alamat IP mesin harus dikelola untuk menjamin keunikan.

Pendaftar domain adalah otoritas yang melakukan hal itu. Ini menetapkan nama domain secara langsung di bawah satu atau lebih domain tingkat atas dan mendaftarkannya ke InterNIC, layanan dari Internet Corporation untuk Nama dan Nomor yang Ditugaskan (ICANN), yang memberlakukan keunikan nama domain di seluruh perangkat internet termasuk desktop, perangkat seluler, dan lainnya. Setiap pendaftaran domain adalah bagian dari basis data domain terpusat yang disebut basis data whois. BigRock dan

GoDaddy.com adalah beberapa perusahaan yang menawarkan layanan pendaftaran domain.

Server DNS yang mengelola domain tertentu disebut **permulaan otoritas** (**SOA**) untuk domain itu. Pada waktunya, hasil dari mencari host di SOA untuk domain menyebar ke server DNS lain di Internet. **Root name server** adalah jenis server DNS yang mulai di bagian atas hierarki domain untuk domain tingkat atas yang diberikan untuk mencari SOA untuk domain.

Sekarang kami telah memahami istilah umum yang digunakan di dunia DNS, mari kita lihat alat yang digunakan oleh penyedia layanan untuk mengelola DNS.

Mungkin ada beberapa konfigurasi domain yang tersedia untuk aplikasi web yaitu subdomain (satu atau banyak) seperti www.sampleapp.com, domain root seperti sampleapp.com atau domain wildcard seperti * .sampleapp.com. Ada banyak alat yang tersedia untuk mengelola pengaturan DNS untuk domain Anda untuk melakukan tugas-tugas berikut:

- Tambahkan subdomain
- Arahkan ulang ke URL yang sepenuhnya memenuhi syarat
- Konfigurasikan email
- Layanan manajemen domain lainnya

Ketika informasi DNS untuk domain dikonfigurasi, data disimpan dalam file zona di server DNS. Beberapa penyedia yang menawarkan layanan manajemen DNS juga menyediakan antarmuka web yang ramah pengguna untuk mengedit konfigurasi ini. Setiap konfigurasi DNS baru yang kami tambahkan disebut catatan. Jenis rekaman yang paling umum dikonfigurasi untuk server DNS ditunjukkan pada tabel berikut:

Ketika informasi DNS untuk domain dikonfigurasi, data disimpan dalam file zona di server DNS. Beberapa penyedia yang menawarkan layanan manajemen DNS juga menyediakan antarmuka web yang ramah pengguna untuk mengedit konfigurasi ini.

Setiap konfigurasi DNS baru yang kami tambahkan disebut catatan. Jenis rekaman yang paling umum dikonfigurasi untuk server DNS ditunjukkan pada tabel berikut:

Catat Nama	Notasi	Deskripsi
Tuan rumah	A	Ini adalah pemetaan dasar dari alamt IP ke nama Host
Nama Kanonikal	CNAME	Ini alias untuk domain. pengguna yang mengakses alias ini diarahkan ke server yang ditunjukkan dalam tipe catatan A

Bekerja dengan DNS di Heroku

Dalam konteks Heroku, untuk menggunakan domain khusus, sangat penting untuk secara akurat mengkonfigurasi informasi DNS aplikasi Anda (seperangkat aturan) ke nama host yang tepat. Setelah dikonfigurasi, informasi DNS akan disebarkan ke bagian lain Internet sebagai DNS. server berjalan di beberapa lokasi di Internet untuk melayani permintaan pengguna dari berbagai wilayah. Setiap permintaan lebih lanjut ke URL yang diperbarui pada akhirnya akan mengarahkan pengguna ke server yang benar yang menampung aplikasi Anda.

Konfigurasikan domain Anda

Satu aplikasi dapat memiliki sejumlah domain yang ditugaskan untuk itu dengan setiap domain milik satu atau lebih dari jenis yang dijelaskan sebelumnya.

Untuk menyiapkan domain, ikuti langkah-langkah ini:

1. Beri tahu Heroku domain khusus mana yang khusus untuk aplikasi Anda.
2. Konfigurasikan DNS aplikasi Anda untuk menunjuk ke Heroku.
3. Ikuti instruksi spesifik untuk setiap konfigurasi sebagaimana dirinci di bagian yang akan datang.

Aturan penambahan domain

Heroku memastikan bahwa domain yang diklaim oleh satu pengguna tidak digunakan oleh pengguna lain pada aplikasi yang berbeda.

Ada beberapa aturan yang terkait dengan menambahkan domain khusus ke aplikasi Heroku Anda:

- Anda hanya dapat menambahkan satu domain ke satu aplikasi. Misalnya, jika www.sampleapp.com ditambahkan ke aplikasi,

sampleapp, Anda tidak diizinkan menambahkannya ke aplikasi lain, seperti sampleapp2. Namun, satu aplikasi dapat memiliki beberapa domain atau subdomain yang ditugaskan.

- Anda dapat menambahkan domain wildcard jika Anda memiliki semua aplikasi yang sudah ada menggunakan subdomain yang sesuai. Misalnya, jika aplikasi sudah menggunakan www.sampleapp.com, Anda harus memilikinya untuk menambahkan * .sampleapp.com.
- Anda dapat menambahkan subdomain atau domain puncak jika Anda memiliki aplikasi yang ditugaskan ke domain wildcard yang sesuai. Misalnya, untuk menambahkan www.sampleapp.com atau sampleapp.com, Anda harus memiliki aplikasi dengan * .sampleapp. com, jika ada domain khusus seperti itu.

Menambahkan domain khusus ke Heroku

Untuk menambahkan subdomain khusus ke aplikasi Heroku Anda, gunakan domain heroku: tambahkan perintah CLI dari prompt perintah:

```
$ heroku domains:add www.sampleapp.com
Adding www.sampleapp.com to sampleapp... done
$ heroku domains:add example.com
Adding example.com to sampleapp... done
```

Di sini, kami menambahkan dua nama host yang berbeda ke domain.

Mengkonfigurasi DNS domain

Untuk setiap subdomain, konfigurasikan DNS Anda dengan catatan CNAME yang mengarahkan subdomain ke nama host Heroku herokuapp.com aplikasi Anda. Trailing. pada domain target mungkin atau mungkin tidak diperlukan, tergantung pada penyedia DNS Anda.

Catatan CNAME yang ditampilkan di sini menyelesaikan www.sampleapp.com ke aplikasi sampleapp:

Merekam	Nama	Target
CNAME	www	sampleapp.herokuapp.com

Memeriksa konfigurasi DNS

Anda dapat mengonfirmasi bahwa DNS Anda dikonfigurasi dengan benar dengan perintah host sebagai berikut:

$ host www.sampleapp.com
www.sampleapp.com is an alias for sampleapp. herokuapp.com.

Hasil dari perintah host menunjukkan bahwa nama host adalah alias atau CNAME untuk sampleapp.herokuapp.com.

Jika Anda bermaksud menggunakan domain root, misalnya, sampleapp.com atau sampleapp. co.in, Anda harus menambahkannya ke subdomain khusus apa pun sebagai berikut:

$ heroku domains:add sampleapp.com
Adding sampleapp.com to sampleapp... done

Zone apex domain (juga disebut domain telanjang atau root), misalnya, sampleapp. com, menggunakan catatan DNS A tidak didukung di Heroku, meskipun ada konfigurasi alternatif yang memungkinkan untuk domain root sementara masih memungkinkan sistem untuk menjadi tangguh dalam lingkungan runtime yang dinamis.

Ada beberapa layanan hosting DNS yang menyediakan cara untuk mendapatkan fungsionalitas seperti CNAME di puncak zona menggunakan tipe catatan khusus.

Dua tipe catatan tersebut sebagaimana tercantum dalam tabel berikut:

Jenis rekaman	Pemberi
ANAME	DNS made easy
ALIAS	DNS simple

To work around issues with the zone apex domains, you can set up the domain information as per the following steps (for most providers, the setup is very similar):

1. Arahkan entri ALIAS atau ANAME untuk domain puncak Anda ke sampleapp. herokuapp.com, sama seperti yang Anda lakukan dengan catatan CNAME.
2. Bergantung pada penyedia DNS, nilai kosong atau @ nama mengidentifikasi puncak zona:

Jika penyedia DNS tidak mendukung tipe catatan seperti itu, dan Anda tidak bisa beralih ke yang merekam, gunakan pengalihan subdomain untuk mengirim permintaan domain root ke aplikasi Anda di Heroku.

Sebagian besar penyedia layanan DNS menawarkan pengalihan subdomain yang membantu mengarahkan semua permintaan domain root ke subdomain yang ditentukan menggunakan pengalihan permanen HTTP 301. GoDaddy.com adalah salah satu penyedia layanan yang menawarkan layanan penerusan domain yang sangat penting. Namun, menggunakan penerusan domain dapat menyebabkan kesalahan atau peringatan muncul jika permintaan aman ke domain root (https: //sappapp.com dibuat. Jika tidak, tidak akan ada efek pada aplikasi Anda jika diakses dengan aman dalam bentuk subdomain SSL, misalnya, https://www.sampleapp.com, atau jika SSL tidak digunakan sama sekali.

Konfigurasi DNS dengan penerusan domain akan terlihat mirip dengan data yang diberikan dalam tabel berikut:

Gunakan domain wildcard untuk memetakan semua subdomain ke aplikasi Anda dengan satu catatan. Kasus penggunaan yang jelas untuk domain wildcard adalah dengan aplikasi yang menggunakan subdomain yang dipersonalisasi untuk setiap pengguna atau akun.

Kami dapat menambahkan domain wildcard hanya jika kami memiliki semua aplikasi yang sudah ada menggunakan domain tingkat atas yang sama. Misalnya, jika aplikasi sudah menggunakan www.sampleapp.com, Anda harus memilikinya untuk menambahkan * .sampleapp.com.

Untuk menambahkan domain wildcard ke aplikasi Anda, gunakan notasi * subdomain wildcard sebagai berikut:

$ heroku domains:add *.sampleapp.com
Adding *.sampleapp.com to sampleapp... done

Gunakan notasi * subdomain wildcard untuk menambahkan catatan CNAME ke sampleapp. herokuapp.com dengan penyedia DNS Anda seperti yang ditunjukkan pada tabel berikut:

Jika salah satu aplikasi memiliki domain wildcard, kami masih dapat menambahkan subdomain spesifik dari domain tingkat atas yang sama ke salah satu aplikasi kami yang lain. Subdomain spesifik dievaluasi sebelum domain wildcard ketika merutekan permintaan.

Menghapus subdomain khusus Heroku

Untuk menghapus subdomain khusus yang dikaitkan dengan aplikasi Heroku Anda, gunakan domain heroku: hapus perintah CLI dari prompt:

$ heroku domains:remove www.sampleapp.com
Removing www.sampleapp.com from example... done

Saat Anda memusnahkan suatu aplikasi, setiap domain khusus yang dikaitkan dengannya dilepaskan secara otomatis. Anda selanjutnya dapat menetapkan domain ini untuk aplikasi lain.

Pertimbangan terkait domain lainnya

Bahkan jika Anda mengatur domain khusus, domain Heroku sampleapp.herokuapp.com akan selalu tetap aktif.

1. Untuk mengarahkan pengguna ke domain khusus yang dikonfigurasikan secara eksklusif, aplikasi Anda harus mengirim status HTTP 301 (dipindahkan secara permanen) ke memerintahkan browser web untuk menggunakan domain khusus.
2. Bidang tajuk permintaan HTTP Host akan menunjukkan domain mana yang diminta pengguna dan mengarahkan pengguna jika nilai bidang itu adalah contoh aplikasi. sampleapp herokuapp.com.

3. Nama domain dapat berisi karakter non-ASCII atau aksen. Nama-nama domain ini harus ditambahkan hanya setelah mengkonversi karakter khusus menggunakan punycode. Misalnya, nama domain éso.com akan dikonversi ke xn--so-9la.com sebelum beralih ke domain heroku: tambahkan:
   $ heroku domains:add xn--so-9la.com

Mengoptimalkan aplikasi

Ada beberapa cara yang dapat dimanfaatkan oleh pengembang Heroku untuk mendapatkan lebih banyak dari aplikasi cloud mereka di platform Heroku. Di bagian ini, kami membahas beberapa fitur ini yang dapat digunakan untuk mengoptimalkan eksekusi aplikasi Heroku Anda saat dibutuhkan.

Efek 2X dyno

Heroku baru-baru ini memperkenalkan 2X dyno. 2X dyno memiliki kapasitas memori dua kali lipat jika dibandingkan dengan 1X dyno yang ada yang berjalan pada platform Heroku. 2X dyno hadir dengan 1 GB RAM dan dua kali lipat pembagian CPU agar proses Anda berjalan lebih cepat dan lebih efisien. 2X dyno juga lebih mahal. Ini menarik harga $ 0,10 / jam-persis dua kali lipat harga satu dino 1X.

Sekarang, pertanyaan yang dapat muncul adalah, "Mengapa saya membutuhkan 2X dyno di tempat pertama? Saya sudah menjalankan aplikasi produksi saya di instalasi Heroku yang ada. Mengapa saya harus beralih ke penyebaran Heroku berbasis-2X-dyno jika sama sekali?" Jawabannya tergantung pada kasus penggunaan atau skenario khusus untuk kebutuhan aplikasi Anda. Berikut ini adalah daftar berbagai skenario:

- **Skenario 1:** Perusahaan A memiliki sistem pemrosesan data besar yang memproses GB data melalui aplikasi pemrosesan data terdistribusi. Salah satu node yang dijalankan menggunakan 1X dyno dan terus-menerus menemukan R14 dari kesalahan memori. Selain itu, Anda terkadang menemukan masalah ini di seluruh node. Nah, 2X dyno mungkin bisa menjadi solusi dalam hal ini. Ini mungkin memberi Anda cukup memori

untuk menjalankan aplikasi pemrosesan terdistribusi tanpa membanjiri log dengan kesalahan yang sulit ditemukan.

- **Skenario 2**: Perusahaan B menggunakan runtime bahasa berbasis JVM dan aplikasi kritis bersifat multi-threaded. Meskipun JVM dirancang dengan tujuan untuk menyediakan konkurensi multithreaded, aplikasi Anda mungkin terbatas pada jumlah utas yang dapat didukung atau ukuran memori yang dapat digunakan oleh masing-masing utas untuk memproses data. 2X dyno juga bisa menyelamatkan dalam kasus ini.
- **Skenario 3**: Perusahaan C memiliki terlalu banyak 1X dino dalam sistem produksi Heroku mereka dan sistem ini tampaknya mengalami banyak masalah memori. Ini selanjutnya menghasilkan respons yang tertunda dan kinerja keseluruhan yang buruk untuk pengguna akhir. 2X dyno juga bisa melakukan trik di sini. Mengganti sebagian besar 1X dyno dengan jumlah yang lebih kecil dari 2X dyno dengan lebih banyak memori per dyno dan pembagian CPU akan meningkatkan kinerja masing-masing dyno. Ini hasil dari fakta bahwa aplikasi cenderung memanfaatkan antrian dalam-dino yang lebih baik dalam jumlah yang lebih kecil dari dino dan memberikan peningkatan kinerja secara keseluruhan.

Kapan saya membutuhkan 2X dyno?

2X dyno memberikan peningkatan kinerja luar biasa untuk aplikasi Anda jika aplikasi Anda menggunakan komputasi atau memori yang intensif. Ini dapat memberikan peningkatan kinerja yang diperlukan yang dibutuhkan oleh aplikasi produksi Anda karena dapat mendukung peningkatan basis pengguna. Namun, 2X dyno mungkin tidak seefektif jika aplikasi Anda lebih terikat I / O karena aplikasi tidak akan dapat memanfaatkan manfaat dua kali memori atau CPU.

Jadi, kapan Anda memutuskan untuk menggunakan 2X dyno? Nah, jika Anda mendapatkan banyak masalah memori R14, itu bisa menjadi salah satu petunjuk, atau jika Anda ingin memanfaatkan kemampuan yang lebih baik dalam mengelola antrian pajak dalam dyno 2X dibandingkan dengan tugas-tugas yang membelah antara beberapa dinamika 1X. Jika Anda ingin mengoptimalkan biaya

aplikasi Anda, itu juga merupakan alasan kuat untuk menggunakan 2X dynos. Menggunakan 2X dinos versus 1X dino sering dapat menyebabkan berkurangnya jumlah dino yang diperlukan oleh lebih dari satu faktor dua.

Memeriksa apakah Anda memerlukan 2X dyno

Ada beberapa cara untuk menentukan apakah aplikasi Heroku Anda mencapai batas tertinggi dalam hal penggunaan memori dan menjamin peningkatan memori untuk kinerja yang lebih baik. Heroku menyediakan alat yang berguna berikut untuk menentukan kebutuhan akan 2X dyno:

- **Log2viz**: Alat ini membantu kami memahami berapa banyak memori dan CPU yang digunakan oleh konfigurasi dyno aplikasi saat ini. Jika penggunaan memori mencapai batas atas 512 MB secara teratur, mungkin sudah saatnya untuk meningkatkan konfigurasi dyno Anda dengan 2X dyno. Pembaruan tersedia di https://github.com/heroku/log2viz.
- **NewRelic**: Alat pemantauan aplikasi NewRelic adalah salah satu alat terbaik yang tersedia untuk melihat perilaku web dyno secara berkelanjutan. Kumpulan grafik yang kaya, dan data yang ditampilkan oleh NewRelic dapat membantu memvisualisasikan dampak penggunaan memori yang tinggi pada konfigurasi aplikasi yang kompleks. Menggunakan 2X web dynos dapat membantu menyediakan waktu antrian yang konsisten untuk tugas dan meningkatkan waktu pemrosesan. Untuk mengetahui lebih lanjut tentang NewRelic, silakan kunjungi http://www.newrelic.com.

Bagaimana jika saya menggunakan 2X dynos?

2X dyno berharga 0,10 dolar per jam dan menyediakan dua kali memori per dyno (1 GB) dan CPU bila dibandingkan dengan 1X dyno. 2X dynos menyediakan aplikasi dengan kemampuan untuk skala secara vertikal. Menyebarkan 2X dino menggantikan 1X dino dapat mengurangi jumlah dino yang diperlukan untuk aplikasi Anda dengan faktor dua atau lebih. Menggunakan2X dynos juga mengurangi biaya pengoperasian aplikasi selain meningkatkan kinerja aplikasi dari tugas-tugas intensif memori secara signifikan.

Bahkan pengguna dengan langganan gratis (gratis 750 dino jam per bulan), dapat menggunakan 2X dino dan meningkatkan kinerja aplikasi mereka. Satu-satunya hal yang perlu diingat adalah bahwa pada migrasi semua aplikasi web ke 2X, jumlah jam gratis juga akan dikurangi menjadi 375 dyno jam per bulan. Penjadwal Heroku juga mendukung menjalankan dino 2X satu kali.

Tabel berikut mengilustrasikan efek menggunakan 2X dino di aplikasi web Anda:

Sekarang beberapa contoh ...

Setelah kami menentukan bahwa aplikasi membutuhkan 2X dyno, mengubah ukuran jumlah dyno aplikasi dapat dilakukan hanya dengan menggunakan perintah ps: resize:

- Untuk mengubah ukuran penghitungan dyno, ketik:
  $ heroku ps:resize web=2X worker=1X
  Resizing dynos and restarting specified processes... done
  web dynos now 2X ($0.10/dyno-hour)
  worker dynos now 1X ($0.05/dyno-hour)

- Untuk melihat ukuran dyno dari jenis proses, gunakan perintah ps:
  $ heroku ps
  === web (2X): `bundle exec unicorn -p $PORT -c ./config/unicorn.
  rb`
  web.1: up 2013/04/15 16:25:15 (~ 3h ago)
  web.2: up 2013/04/15 16:46:23 (~ 3h ago)
  web.3: up 2013/04/15 17:08:34 (~ 2h ago)
  === worker (1X): `bundle exec rake worker:job`
  worker.1: up 2013/04/15 16:39:04 (~ 3h ago)
  worker.2: up 2013/04/15 17:08:24 (~ 2h ago)
  worker.3: up 2013/04/15 16:30:55 (~ 3h ago)
- Untuk menerapkan ukuran dyno pada basis per jenis proses, atur tipe dyno untuk setiap jenis proses seperti yang ditunjukkan di bawah ini:
  $ heroku ps:resize web=2X worker=1X worker2=2X

- Untuk mengkonfigurasi satu-off 2X dino (misalnya, pekerjaan intensif memori), masukkan perintah berikut:
  $ heroku run --size=2X rake heavy:job

Catatan tentang 2X dynos

Sangat mungkin menjalankan aplikasi yang haus memori dengan 1X dyno dapat menyebabkan program Anda bertukar ke disk secara konstan karena log menunjukkan banyak kesalahan R14, dan kemudian Heroku harus secara paksa membunuh dyno jika kebutuhan memori melebihi tiga kali. batas memori dyno. Ini pada akhirnya menyebabkan banyak penurunan kinerja. Mungkin saja terjadi bahwa tidak peduli berapa banyak lagi 1X dinamo yang Anda mulai, aplikasi Anda tidak dapat mengikuti lonjakan kebutuhan memori dan sering berpindah ke disk. Solusi dalam kasus ini adalah memanfaatkan 2X dyno. Anda dapat menulis skrip untuk mendeteksi kebutuhan memori aplikasi dan segera setelah memori melewati ambang yang telah ditentukan (95 persen atau lebih), skrip dapat membunuh dyno dan memulai versi 2X dari ukuran dyno untuk aplikasi yang diubah ukurannya. Dengan cara ini, aplikasi bekerja dengan mulus tanpa menyebabkan hambatan kinerja bagi konsumen aplikasi.

Mengelola dinamika aplikasi Anda

Heroku adalah platform yang digunakan oleh pengembang produk komersial dan juga penggemar. Ini adalah platform yang ideal untuk membuktikan ide konsep sebelum membangun aplikasi web yang canggih di atasnya. Ada banyak pengguna yang menggunakan Heroku untuk tujuan itu. Biasanya, para pengguna ini memiliki konfigurasi Heroku minimum. Secara khusus, mereka hanya memiliki satu dino web yang menjalankan aplikasi mereka di platform Heroku. Bayangkan 10.000 dari pengguna ini mencoba menggunakan platform Heroku untuk meng-host aplikasi yang tidak berjalan secara teratur atau berjalan untuk waktu singkat. Yah, mengingat bahwa Heroku adalah PaaS yang sangat populer yang menjadi tuan rumah sejumlah besar aplikasi web dunia nyata dan sejumlah aplikasi hobi yang sama banyaknya, Heroku memiliki cara

untuk mengoptimalkan penggunaan sumber daya dyno. Heroku mematikan setiap aplikasi dyno web setelah satu jam tidak aktif.

Jika Anda memeriksa log aplikasi saat aplikasi tidak aktif, mereka mungkin terlihat seperti ini:

2013-11-30T08:23:09+00:00 heroku[web.1]: Idling

2013-11-30T08:23:17+00:00 heroku[web.1]: Stopping process with

SIGTERM

Karena Heroku mengelola aplikasi, pengguna akhir tidak mengetahui hal ini akan dimatikan kecuali jika pengguna memonitor aplikasi dengan cermat. Dalam kasus seperti itu, setiap kali permintaan baru datang untuk aplikasi web Anda, router yang memproses permintaan tersebut akan memberi tahu manajer dyno dan memintanya untuk mengaktifkan web dyno. Ini akan tercermin dalam log aplikasi sebagai berikut:

2013-11-30T10:12:34+00:00 heroku[web.1]: Unidling

2013-11-30T10:12:34+00:00 heroku[web.1]: State changed from created to starting

Sejak aplikasi dimulai dari awal, beberapa permintaan pertama mengalami beberapa detik penundaan. Poin yang perlu diperhatikan adalah bahwa jika Anda menjalankan lebih dari satu dyno web, dyno Anda tidak akan di-idle. Juga, dinamika pekerja juga tidak menganggur.

Cara umum untuk mengatasi kebijakan idle Heroku adalah dengan membuat skrip untuk mengirim ping sekali dalam satu jam agar web dyno tetap hidup.

Ada beberapa cara untuk menjaga agar web dyno tetap hidup:

- Menggunakan penjadwal Heroku
- Menggunakan fitur tambahan NewRelic

Menggunakan penjadwal Heroku

Sangat mudah untuk membuat dino ping tetap-hidup menggunakan penjadwal Heroku. Jika Anda seorang pengembang Ruby, Anda dapat membuat tugas menyapu sebagai berikut, untuk ping otomatis ke dyno web Anda:

1. Tulis skrip berikut:

```
desc "Pings YOUR_HEROKU_APP_URL to keep a dyno alive"
 task :wakeup_dyno do
 require "net/http"
 if ENV['YOUR_HEROKU_APP_URL']
 uri = URI(ENV['YOUR_HEROKU_APP_URL'])
 Net::HTTP.get_response(uri)
 end
end
```

2. Tambahkan YOUR_HEROKU_APP_URL ke lingkungan Heroku Anda:
   $ heroku config:add YOUR_HEROKU_APP_URL=https://yourherokuapp.herokuapp.com
3. Sekarang, siapkan penjadwal Heroku:
   $ heroku addons:add scheduler:standard
   $ heroku addons:open scheduler
   Perintah terakhir itu harus membuka antarmuka scheduler di browser Anda.
4. Siapkan tugas wakeup_dyno Anda untuk berjalan sekali dalam satu jam dan jalankan:
   $ rake wakeup_dyno

Menggunakan NewRelic untuk menjaga dino tetap hidup

NewRelic (www.newrelic.com) adalah layanan pemantauan aplikasi web dan seluler yang sangat populer. Ini memungkinkan Anda memantau aplikasi naik, turun, menyimpan data, perilaku pengguna, kinerja aplikasi, dan banyak lagi. Ini juga sangat mudah digunakan pada platform Hiroku. Ini mendukung hampir semua aplikasi web atau seluler yang ditulis dalam berbagai bahasa pemrograman seperti Ruby dan Java.

Untuk mengatur ping otomatis untuk aplikasi web Anda menggunakan NewRelic, ikuti langkah-langkah selanjutnya:

1. Tambahkan NewRelic ke akun Heroku Anda. Paket standar NewRelic dapat ditambahkan ke akun Anda sebagai berikut:
   $ heroku addons:add newrelic:standard
2. Buka antarmuka NewRelic sebagai berikut:
   $ heroku addons:open newrelic

3. Buka tab Laporan dan temukan opsi menu Ketersediaan.
4. Tambahkan URL ke monitor dan sesuaikan seberapa sering pemeriksaan dilakukan. Tetapkan waktu kurang dari satu jam untuk cukup sering melakukan ping terhadap dyno web Anda agar tidak ter-idle.

Ringkasan

Dalam bab ini, kami meninjau alat dan fitur yang dapat menambah secara signifikan cara kami mengembangkan aplikasi web Heroku. Kami menjelajahi IDE berbasis web Cloud 9 yang mendukung sebagian besar bahasa pemrograman populer dan menyediakan lingkungan pengembangan yang kuat untuk menulis aplikasi cloud. Kami juga menyelidiki bagaimana layanan data Heroku Postgres bekerja dan bagaimana Anda dapat menggunakan operasi basis data yang paling umum untuk mendukung aplikasi web Anda. Kami juga memperkenalkan resolusi nama domain dan cara kerjanya untuk Heroku ketika Anda mengonfigurasi domain khusus. Akhirnya, dalam fitur-fitur canggih, kami mengeksplorasi kekuatan 2X dyno yang dapat meningkatkan dino web Anda untuk meningkatkan skala dan memberikan kinerja yang unggul. Pada akhirnya, kami melihat beberapa teknik untuk menjaga dinamika web aplikasi kecil Anda saat tidak digunakan. Semua tips ini dapat membantu Anda sebagai referensi, karena Anda memelihara dan meningkatkan aplikasi Anda di platform Heroku.

Pada bab selanjutnya, kita akan mengeksplorasi aspek keamanan platform Heroku dan meninjau alat untuk mengukur kebutuhan keamanan aplikasi Anda dan lebih jauh mengamankan aplikasi cloud Heroku Anda.

BAB 7
KEAMANAN HEROKU

Kami hampir mendekati akhir perjalanan Heroku kami. Pada bab sebelumnya, kami melihat pada penggunaan lanjutan platform Heroku untuk mengembangkan aplikasi cloud. Kami menjelajahi lingkungan pengembangan terintegrasi (IDE) berbasis cloud yang populer dan kaya fitur disebut C9. Kami mengintip ke dalam cara bekerja dengan relasional Heroku PostgreSQL database dan gunakan database untuk melayani dan menyimpan data aplikasi. Melalui primer layanan nama domain (DNS), kami memahami cara yang tepat untuk mengkonfigurasi DNS untuk aplikasi Heroku Anda. Last but not least, kami diperkenalkan ke dunia 2X dynos- mitra yang lebih cepat dari proses dyno tradisional di Heroku. Ada aspek kuncinya pengembangan aplikasi cloud Heroku yang akan kita eksplorasi sekarang: keamanan. Keamanan telah menjadi salah satu perhatian utama bagi perusahaan dan UKM dalam menerima cloud sebagai lingkungan komputasi yang layak. Pikiran menjalankan aplikasi dan menyimpan data di lingkungan yang dihosting telah membuat pengguna berpikir dua kali sebelum memindahkannya aplikasi roti dan mentega ke dunia yang dihosting. Bagaimana jika data rusak? Bagaimana jika data diintai saat bepergian di jaringan? Bagaimana jika aplikasi tersebut diretas melalui lubang keamanan? Bagaimana jika aplikasi mati saat melayani ribuan permintaan klien? Ini dan banyak lagi pertanyaan telah diajukan penyedia layanan

cloud, dan secara tidak langsung platform yang mendasari, dan infrastruktur penyedia untuk memfokuskan banyak upaya mereka untuk mengamankan lingkungan komputasi awan. Tidak terkecuali Heroku

Keamanan pada Heroku perlu dilihat dalam dua dimensi:

- Ketika seorang pengembang berkomunikasi dari mesin lokal ke Heroku platform, penting bahwa komunikasi itu aman dan tidak ada perangkat lunak Program ini dapat mengintip isi komunikasi. Ini adalah Dimensi keamanan pertama sehubungan dengan bekerja dengan Heroku.
- Saat bekerja dengan Heroku, dimensi kedua keamanan adalah keamanan dari Heroku dan server pihak ketiga untuk melindungi aplikasi pengguna dan data terkait dari akses tidak sah. Kedua dimensi ini sama penting bagi pengembang untuk membangun aplikasi yang sangat aman di platform Heroku.
- Dalam bab ini, kami fokus pada dua dimensi keamanan berikut pada platform Heroku: Komunikasi antara mesin pengembang dan platform Heroku
- Keamanan terus menerus dari aplikasi dan data yang ada di Heroku dan server pihak ketiga

Kami juga meninjau beberapa alat keamanan aplikasi yang berguna seperti wwwhisper dan kertas timah, yang tersedia untuk digunakan dari perpustakaan add-on Heroku.

Ikhtisar

Upaya konstan Heroku adalah membiarkan pengembang fokus membuat aplikasi, sedangkan Heroku menyediakan semuanya, mulai dari layanan build, deployment, dan skalabilitas hingga layanan manajemen infrastruktur. Ketika datang ke keamanan, Heroku berlaku terbaik dan mengelola keamanan platform sehingga pelanggan dapat mengabdikan mereka energi sepenuhnya untuk membangun aplikasi web yang kuat, sangat tersedia, dan berkinerja aktif platform Heroku.Platform Heroku melindungi pelanggannya dari ancaman keamanan – dari fisik untuk aplikasi, mengisolasi

aplikasi pengguna dan data terkait – dengan memberlakukan lebih ketat kontrol di setiap lapisan platform. Kemampuan Heroku untuk menyebarkan keamanan dengan cepat pembaruan tanpa intervensi pelanggan atau gangguan layanan membuatnya menarik Lingkungan PaaS digunakan untuk menyebarkan aplikasi skala apa pun.

Jika kami memecah proses pengembangan aplikasi Heroku menjadi beberapa bagian, kami akan melakukannya lihat bahwa siklus hidup aplikasi Heroku terdiri dari yang berikut:

- Pembuatan aplikasi di lingkungan lokal (mesin PC / Linux)
- Mengakses penyimpanan data relasional atau nilai kunci dengan kredensial yang tersedia
- Mendorong kode aplikasi ke platform Heroku
- Menyebarkan aplikasi pada platform Heroku
- Menjalankan aplikasi dalam produksi dengan potensi menyimpan banyak data, cache, antrian, dan perangkat lunak pihak ketiga [218]

Ketika kita melihat Heroku dari aspek keamanan, semuanya bisa salah ini tahapan siklus hidup aplikasi. Mengingat bahwa semua aplikasi Heroku berjalan pada platform EC2 Amazon di lapisan infrastruktur berarti platform Heroku keamanan bergantung pada keamanan layanan infrastruktur perangkat keras yang mendasarinya juga. Oleh karena itu, mengamankan layanan platform Heroku memerlukan pengamanan masing-masing fase ini dari siklus hidup aplikasi Heroku. Bab ini menjelaskan bagaimana Heroku menyediakan lingkungan aman, hampir antipeluru, aman untuk aplikasi Anda berjalan dengan aman dan mulus.

Komunikasi antara pengembang mesin dan platform Heroku

Pada bagian ini, kami akan meninjau konsep keamanan dasar seputar komunikasi mesin-ke-mesin. Kami juga akan meninjau bagaimana pengembang berkomunikasi dengan server Heroku secara aman menggunakan yang terkenal protokol komunikasi.

Konsep umum keamanan

Pengembang Heroku secara teratur berinteraksi dengan server / platform Heroku untuk membuatnya perubahan pada aplikasi, konfigurasi aplikasi, atau data aplikasi. Dengan demikian, pengembang berulang kali harus terhubung ke sistem jarak jauh dan mengautentikasi kredensial pengguna sebelum diterima sebagai pengguna asli. Di bagian ini, kita melihat apa yang membuat komunikasi dengan sistem jarak jauh memungkinkan untuk pengembang Heroku dan juga diskusikan secara singkat algoritma yang mendasari yang digunakan untuk mengamankan akses ke interaksi antara pengembang dan sistem jarak jauh (dalam kasus kami server Heroku)

What the developer needs to communicate with the remote system (Heroku server) is as follows:

- Keamanan komunikasi; ini berarti bahwa tidak seorang pun dapat melakukannya membaca pesan yang dikirim, dan karenanya pesan tersebut harus dienkripsi selama transfer
- Integritas komunikasi itu penting, yaitu, kedua pihak – klien dan server – harus dapat memeriksa apakah pesan telah diubah dalam perjalanan
- Server dan klien harus dapat mengidentifikasi satu sama lain untuk membangun saluran otentikasi aman.

Untuk mengaktifkan kebutuhan pengembang ini, sebagian besar aplikasi menggunakan beberapa bentuk kriptografi algoritma untuk mengamankan komunikasi antara berbagai pihak. Yang paling jenis umum dari algoritma kriptografi untuk enkripsi dan keamanan data adalah sebagai berikut:

- **Algoritma kriptografi kunci publik**: Ini juga dikenal sebagai asimetris kriptografi. Dalam algoritma ini, sepasang kunci digunakan. Salah satu kuncinya disebut kunci pribadi dirahasiakan dan tidak dibagikan dengan pihak mana pun. Itu kunci lain yang dikenal sebagai kunci publik bukan rahasia dan dapat dibagikan pihak lain. Data dienkripsi oleh salah satu kunci hanya dapat didekripsi dan pulih menggunakan kunci lainnya.

Hampir tidak mungkin untuk mendapatkan privat kunci dari kunci publik meskipun kedua kunci tersebut secara matematis terkait. Contoh umum dari algoritma kunci publik adalah RSA algoritma Pengaturan Heroku SSH adalah salah satu skenario di mana Anda dapat menggunakan Algoritma RSA (dinamai berdasarkan pendirinya) untuk memungkinkan komunikasi yang aman antara mesin pengembang dan server Heroku.

- **Algoritma kriptografi kunci rahasia**: Ini juga dikenal sebagai simetris kriptografi. Dalam mekanisme ini, kuncinya adalah rahasia bersama antara dua pihak yang berkomunikasi, dan enkripsi dan dekripsi keduanya menggunakan kunci yang sama. Contoh umum algoritma kunci simetris adalah **Data Standar Enkripsi (DES**) dan **Standar Enkripsi Lanjutan (AES).**

Algoritma kriptografi asimetris lebih lambat daripada algoritma kunci simetris. Oleh karena itu, untuk sebagian besar kasus, aplikasi menggunakan algoritma asimetris untuk mengenkripsi simetris kunci (untuk mendistribusikan kunci) dan hash (untuk membuat tanda tangan digital). Kunci dan Algoritma kriptografi bersama-sama mengubah data. Ini adalah kunci yang mengontrol akses ke data dan karenanya harus tetap aman untuk melindungi data. Algoritma bersifat publik pengetahuan dan karenanya dapat digunakan oleh siapa saja.

Keamanan komunikasi pengembang dengan Heroku

SSH menggunakan kriptografi kunci publik RSA untuk koneksi dan otentikasi. Algoritma enkripsi termasuk DES, Blowfish, dan IDEA (default). SSH2 adalah yang terbaru versi protokol SSH yang diusulkan sebagai seperangkat standar untuk komunikasi yang aman. SSH memungkinkan Anda untuk menjalankan program baris perintah dan grafik, mentransfer file, dan bahkan membuat jaringan pribadi virtual aman.

SSH menyediakan mekanisme otentikasi berbasis kata sandi dan berbasis kunci. Meskipun otentikasi berbasis kata sandi juga aman, menggunakan otentikasi berbasis kunci direkomendasikan. Ada fitur tambahan seperti tunneling SSH dan TCP port forwarding

yang berada di luar ruang lingkup diskusi kami. Di bagian ini, kita akan fokus pada kasus penggunaan yang paling relevan tentang bagaimana klien (mesin pengembang) berkomunikasi dengan server (server Heroku) melalui SSH dan yang mendasarinya semantik komunikasi ini.

Seperti yang telah kita lihat di bagian Menyiapkan SSH di Bab 4, Menyebarkan Heroku Aplikasi, bekerja dengan platform Heroku membutuhkan saluran aman komunikasi antara mesin atau lingkungan lokal pengembang dan Platform heroku akan didirikan. Pengembang menyiapkan klien SSH di lokal mesin pengembangan. Server SSH yang berjalan pada platform Heroku menerima permintaan dari mesin klien untuk mengotentikasi klien. Pengembang menggunakan perintah heroku login untuk terhubung ke server Heroku, dan setelah dikonfirmasi, diizinkan untuk melakukan berbagai operasi, seperti membuat atau memodifikasi aplikasi web di platform. Semua komunikasi antara mesin lokal pengembang dan platform Heroku dienkripsi selama interaksi antara pengembang dan platform. Tautan berikut menunjukkan perbandingan beragam Klien SSH tersedia bagi pengembang untuk digunakan: http://en.wikipedia.org/wiki/Klien_comparison_of_SSH_.Dalam konteks contoh buku ini, kami punya menggunakan implementasi open source OpenSSH untuk mengatur SSH Tampak di dalam protokol SSH Protokol SSH bekerja dengan pertukaran dan verifikasi informasi, menggunakan publik dan kunci pribadi, untuk mengidentifikasi klien dan host. Setelah pihak yang berkomunikasi memiliki telah diidentifikasi, SSH menyediakan enkripsi untuk komunikasi selanjutnya menggunakan kriptografi kunci publik / pribadi.

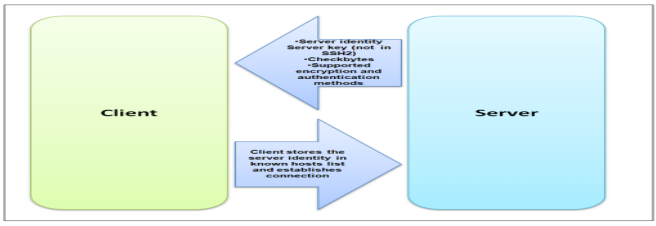
Klien istilah berarti workstation pengembang atau PC, sedangkan server istilah berarti workstation atau PC jarak jauh sekunder yang ingin Anda hubungi bekerja, seperti server sesi masuk. Dalam konteks Heroku, klien adalah mesin tempat pengembang mengetikkan perintah heroku login untuk mencoba terhubung platform Heroku. Tuan rumah adalah server Heroku yang disambungkan oleh klien setelahnya mendapatkan otentikasi. Pengembang Heroku menghasilkan "identitas" pada mesin klien

dengan menjalankan Program ssh-keygen. Program ini membuat subdirektori $ HOME / .ssh dan membuat dua file bernama identitas dan identitas. File-file ini berisi file pribadi dan kunci publik untuk akun pengembang di mesin pengembang. File yang terakhir kemudian dapat ditambahkan ke file bernama <direktori home pengguna> /. ssh / resmi_ kunci, yang perlu disalin ke semua / semua server yang diinginkan pengembang terhubung ke melalui SSH.

Dengan SSH, pengembang menghasilkan pasangan kunci publik dan kunci privat untuk lokal workstation menggunakan alat ssh-keygen. Seperti kunci ada di pengembang workstation itu sendiri dan dilindungi (terutama kunci privat), kemungkinan pihak ketiga mencuri identitas sistem dengan berbagai cara seperti memanipulasi Catatan DNS atau alamat IP fudging minimal. Satu-satunya cara Anda bisa berpura-pura untuk menjadi pengguna yang sah padahal sebenarnya tidak Anda lakukan adalah jika Anda mendapatkan akses ke pengguna tersebut kunci pribadi pada mesin lokal mereka. Itu sendiri sangat tidak mungkin.

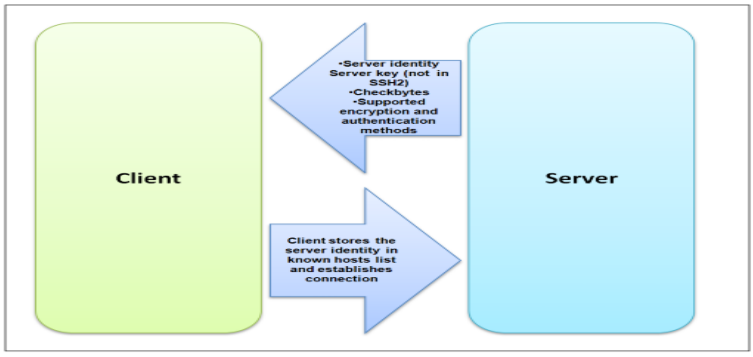
Interaksi klien dan server berikut menunjukkan bagaimana klien dan server saling mengautentikasi sebelum mengirim data terenkripsi satu sama lain menggunakan SSH. Langkah-langkah berikut dilakukan untuk otentikasi server.

1. Klien terhubung ke server dan meminta informasi. Server mengirim versi protokol yang didukung dan versi server SSH yang berjalan di server.

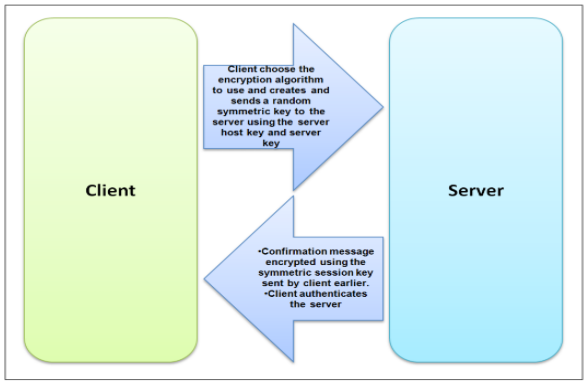


2. Berdasarkan versi protokol yang diterima, klien memutuskan apakah akan lanjutkan komunikasi atau tidak. Jika komunikasi dilanjutkan, baik klien dan server **beralih ke protokol paket biner (BPP).** Di ini server mengirimkan identitas server, kunci

server (tidak digunakan dalam SSH2), checkbytes (klien mengirim ini di balasan berikutnya sebagai jabat tangan), dan daftar metode enkripsi dan otentikasi yang didukung. BPP bertanggung jawab atas enkripsi simetris yang mendasarinya dan otentikasi semua pesan yang dikirim antara dua pihak terlibat dalam koneksi SSH.



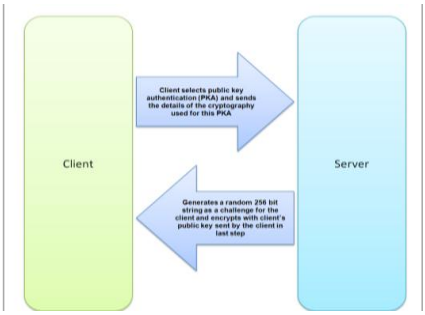
3. Klien kemudian memilih algoritma enkripsi dan membuat acak kunci simetris yang kemudian dikirim ke server. Server membalas dengan a pesan konfirmasi dienkripsi menggunakan kunci simetris yang dikirim sebelumnya oleh klien. Klien sekarang menerima pesan dan mendekripsi menggunakan kunci itu sudah memiliki dan mengautentikasi server tempat ia dikirim.



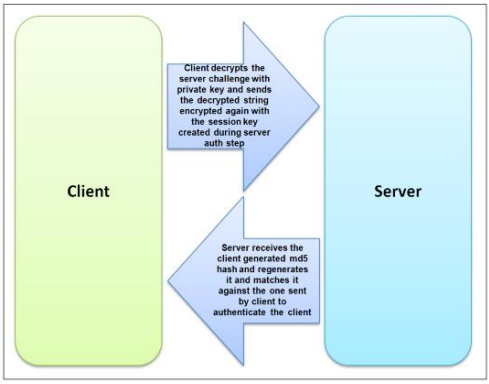
Otentikasi klien

Langkah-langkah berikut dilakukan untuk otentikasi klien:

1. Sejauh ini, klien baru saja mengotentikasi server dan kunci simetris rahasia telah dibuat. Untuk otentikasi klien, klien pertama-tama memilih publik kriptografi kunci sebagai mekanisme otentikasi yang diinginkan dan mengirimkan kriptografi yang digunakan untuk otentikasi kunci publik ini. Pada langkah yang sama, server membuat string 256-bit sebagai tantangan untuk klien dan mengenkripsi itu menggunakan kunci publik yang dikirim oleh klien sebelumnya.



2. Klien mendekripsi tantangan server dengan kunci pribadinya dan mengirimkan pesan yang didekripsi ke server dalam bentuk terenkripsi menggunakan sesi kunci (kunci simetris yang dibuat selama otentikasi server). Server menerima hash md5 yang dihasilkan oleh klien, meregenerasi hash sendiri, dan membandingkan keduanya. Jika cocok, klien ditandai sebagai diautentikasi



Setelah klien dan server saling berbagi identitas masing-masing dikonfirmasi, permintaan transfer data akan dienkripsi menggunakan kunci sesi untuk transfer data yang aman antara kedua mesin.

Keamanan aplikasi dan dasbor Heroku

Ada beberapa opsi yang tersedia di dasbor Heroku untuk dikelola tertentu aspek keamanan aplikasi. Padahal opsi ini sudah dibahas di tempat lain juga, masuk akal untuk mendiskusikan opsi-opsi ini di sini untuk membuat topik aplikasi keamanan lengkap.

Akun Heroku Anda dan dasbor

Setelah Anda masuk ke akun Heroku Anda dan menavigasi ke dasbor Heroku, Anda dapat melihat dan mengubah properti akun dari tab Akun seperti yang ditunjukkan di screenshot berikut

Keamanan aplikasi dan data tetap dihidupkan Server heroku dan pihak ketiga

Di bagian ini, kami meninjau praktik keamanan yang diikuti oleh keamanan Heroku pengembang dan data aman di server Heroku. Heroku juga menyediakan mekanisme untuk secara aman mengakses server pihak ketiga yang menyediakan layanan untuk aplikasi Anda.

Praktik keamanan Heroku

Heroku menggunakan berbagai strategi untuk mengamankan platform dan memberikan yang mulus pengalaman kepada pengembang membuat aplikasi di platform. Dengan hormat keamanan data, Heroku menggunakan berbagai langkah untuk memastikan bahwa aplikasi berjalan dalam cara mulus dan aman.

Keamanan kode sumber

Pengembang Heroku perlu mendorong kode sumber dan artefak lainnya ke Server Heroku, dan karenanya diperlukan bahwa tidak ada yang dapat mengintip sumbernya saat transit. Pengembang menggunakan protokol SSH, seperti yang dijelaskan sebelumnya, untuk mengenkripsi isi kode sumber menggunakan protokol Git saat mereka didorong ke Lingkungan heroku.

Membangun dan menggunakan keamanan

Proses pembuatan Heroku membuat executable (siput) dengan memasukkan kode dari tepercaya sumber dan menyediakan URL unik yang mewakili aplikasi yang dibangun. Kemungkinan seseorang yang memalsukan URL dan mencoba serangan DoS pada aplikasi sangat terbatas, meskipun ada saat ketika peretas mendapatkan akses ke akun pelanggan dengan melewatkan enkripsi dan dapat mengakses informasi akun menggunakan permintaan HTTP berbahaya. Dalam contoh lain, basis data pelanggan dihapus ketika kredensial mereka tidak sengaja terungkap. Terlepas dari tantangan ini, Heroku terus-menerus memperkuat barisan keamanan di sekitar platform, membuat semakin sulit untuk memecahkan metode yang digunakan untuk mengakses platform.

Heroku follows the highest standards and best practices regarding the infrastructure hosting the application. Heroku uses Amazon's EC2 platform as its IaaS layer, which it self follows the strictest of infrastructure security standards. Heroku also leverages the security features of the specific application language or framework to secure deployed applications. Aplikasi Heroku berjalan dalam lingkungan terisolasi mereka sendiri dan tidak dapat berinteraksi dengan aplikasi atau bagian lain dari sistem. Skema ini

memungkinkan keamanan yang lebih ketat kontrol dan lingkungan aplikasi yang lebih stabil.

Application security

Aplikasi apa pun yang berjalan pada platform Heroku dijalankan dalam bentuk a dyno dalam lingkungan proses eksekusi yang disebut dyno manifold. Dyno adalah dieksekusi dalam isolasi lengkap satu sama lain, artinya tidak ada dua aplikasi yang bisa melihat satu sama lain dieksekusi – bahkan ketika pada infrastruktur fisik yang sama. Lingkungan mandiri ini mengisolasi proses, memori, dan sistem file. Setiap dino memiliki batas sumber daya yang telah ditentukan yang dapat digunakan. Manajer dyno mengontrol perilaku setiap dino jika terjadi pelanggaran penggunaan sumber daya. Ini strategi memberikan perlindungan dari proses aplikasi lain dan tingkat sistem proses yang mengkonsumsi semua sumber daya yang tersedia.

Heroku membatasi aplikasi dari membuat koneksi jaringan lokal antara host oleh memiliki firewall di tempat yang didasarkan pada server host. Sebagai bagian dari penyebaran proses, aplikasi yang dihosting di platform Heroku juga disalin atau didukung dengan aman dan penyimpanan yang berlebihan. Heroku menggunakan cadangan ini untuk menyebarkan aplikasi di seluruh platform dan secara otomatis membawa mereka kembali online jika terjadi pemadaman. Di pada saat yang sama, infrastruktur Heroku dirancang untuk menskala dengan mulus dan menjadi kesalahan toleran dengan secara otomatis mengganti instance aplikasi yang gagal dan mengurangi keharusan untuk mengembalikan aplikasi dari cadangan.

Secara umum, Heroku mengambil beberapa langkah untuk melindungi privasi pelanggan dan melindungi aplikasi dan data yang disimpan dalam platform. Berbagai jenis perlindungan diaktifkan pada platform Heroku termasuk otentikasi pengguna, akses kontrol ke sumber daya, enkripsi data (dalam perjalanan atau dalam bentuk yang disimpan), dan HTTPS dukungan untuk aplikasi.

Untuk menilai keamanan platform Heroku, Heroku secara berkala mengalami uji penetrasi sistem, penilaian kerentanan platform, dan kode sumber ulasan. Heroku juga berpartisipasi dalam penilaian keamanan pihak ketiga yang mencakup semua area keamanan di platform, termasuk pengujian untuk kerentanan aplikasi dan memverifikasi isolasi aplikasi.

Keamanan data

Data pelanggan tentu saja merupakan aset terpenting bagi organisasi mana pun. Ini adalah dasar dari semua proses bisnis dan aplikasi yang dibangun di atasnya. Karena itu penting agar data pelanggan diamankan dari serangan jahat, pencurian, atau pengintaian.

Heroku memungkinkan keamanan data pelanggan dengan menjaga agar data pelanggan tetap terkendali basis data. Jika pelanggan memiliki lebih dari satu database, masing-masing database ini memiliki nama pengguna dan kata sandi yang unik untuk mengakses data. Setiap aplikasi biasanya terkait dengan database sendiri untuk menghindari aplikasi silang yang tidak sah intrusi yang dapat menyebabkan korupsi data, meskipun mungkin untuk beberapa aplikasi untuk merujuk pada sumber daya pihak ketiga yang sama. Pengembang, bagaimanapun, memiliki tanggung jawab merancang aplikasi dengan benar sehingga segala jenis pertikaian dan data dihindari konsistensi dipertahankan untuk aplikasi. Postgres – database default sekarang tersedia pada platform Heroku – memerlukan enkripsi SSL untuk terhubung ke aplikasi memanipulasi data.

Selain itu, persyaratan keamanan di sekitar data pelanggan dapat dipenuhi oleh aplikasi itu sendiri, dengan mengenkripsi data yang dipertukarkan melalui jaringan atau dibagikan antara beberapa aplikasi. Selama penyebaran, aplikasi pelanggan harus menggunakan koneksi database terenkripsi untuk menghindari pengintaian pada data yang dikirim server basis data. Aplikasi pelanggan juga diperlukan untuk mengimplementasikan keamanan mekanisme untuk data yang mungkin mereka gunakan dari solusi caching kustom atau terima dari sistem eksternal.

Setiap kali data diubah, perubahan yang sesuai dicatat secara tertulis log; ini pada gilirannya disimpan dalam sistem penyimpanan yang sangat tersedia dan tahan lama di beberapa pusat data. Jika terjadi bencana, mengakibatkan kegagalan perangkat keras atau data korupsi, log dapat diputar ulang untuk memulihkan data hingga pos pemeriksaan terakhir. Anda juga dapat membuat cadangan basis data dari waktu ke waktu untuk memenuhi bisnis Anda kebutuhan untuk keamanan dan penyimpanan data Setelah Anda menolak aplikasi, data terkait disimpan untuk periode tujuh hari, setelah itu aplikasi dan data terkait dihancurkan, membuat data tidak dapat dipulihkan. Secara internal, Heroku dapat menjatuhkan basis data Anda menggunakan pg: reset perintah jika Anda menggunakan database bersama. Konfigurasi dapat dihapus dari sistem file lokal ke aplikasi Anda. Deprovisioning perangkat keras dikelola oleh Amazon, yang menyediakan lapisan infrastruktur dasar untuk platform Heroku.

Kecuali diminta, staf internal Heroku tidak mengakses data pelanggan atau aplikasi. Hanya jika permintaan pelanggan diterima untuk memecahkan masalah dengan tingkat keparahan tinggi masalah atau perintah pemerintah sudah ada untuk mengaudit data pelanggan, staf Heroku akan mendapatkan akses terkontrol ke data dan aplikasi pelanggan. Staf Heroku menyimpan catatan dari seluruh aktivitas dengan melacak dukungan mulai dan berakhirnya waktu, alasannya untuk mengakses sumber daya, dan tindakan yang diambil dalam penyelesaian masalah.

Konfigurasi dan metadata

Sistem Heroku juga mengarsipkan konfigurasi atau metadata aplikasi Anda secara berkala dan arsipkan ke yang sama sangat tersedia, tahan lama, berlebihan infrastruktur yang digunakan untuk menyimpan informasi basis data. Dengan menggunakan arsip ini, orang dapat dengan mudah melacak perubahan konfigurasi yang dialami oleh aplikasi sejak pertama kali dipekerjakan. Pengembang aplikasi dapat dengan mudah kembali ke konfigurasi yang lebih lama untuk referensi atau verifikasi tentang bagaimana aplikasi tersebut telah berkembang dari waktu ke waktu

Keamanan infrastruktur

Ketika infrastruktur dasar berubah, Heroku terus memperbarui platform dengan perubahan baru. Ia memelihara konfigurasi dan konsistensi sistem melalui perangkat lunak manajemen konfigurasi, gambar sistem terbaru, dan penggantian sistem secara berkala dengan versi yang diperbarui. Sistem baru dikerahkan menggunakan gambar terbaru yang berisi perubahan konfigurasi terbaru dan keamanan pembaruan. Ketika sistem baru dikerahkan, sistem yang ada dinonaktifkan. Karena aplikasi pelanggan berjalan di lingkungan yang terisolasi, mereka sebenarnya tidak terpengaruh oleh pembaruan sistem inti.

Akses ke sistem operasi pangkalan yang mendasari dibatasi untuk Heroku resmi staf menggunakan nama pengguna dan otentikasi kunci untuk mendapatkan akses. Serangan brutal, pencurian, dan berbagi tidak dimungkinkan karena otentikasi kata sandi dinonaktifkan untuk pengguna.

Heroku senantiasa terlibat dengan lembaga penilai risiko internal dan eksternal dan mengumpulkan potensi kerentanan dalam platform. Masalah diprioritaskan berdasarkan seberapa jauh jangkauan dampak dan berapa banyak pengguna yang terpengaruh. Berdasarkan pada risiko yang terlibat, masalah diselesaikan dan kewarasan dipulihkan ke platform keseluruhan Heroku menggunakan konsep pengelompokan komponen untuk mengurangi risiko dan membuat tingkat keamanan yang lebih tinggi di platform. Ini mengelompokkan komponen serupa menjadi unik grup keamanan jaringan, menyediakan akses ke sumber daya spesifik (port) dan protokol untuk digunakan. Komponen dari grup keamanan yang berbeda tidak dapat saling mengakses sumber daya. Misalnya, aplikasi pelanggan milik grup jaringan yang berbeda dari infrastruktur manajemen internal Heroku; karenanya, aplikasi pelanggan tidak dapat mengakses sumber daya apa pun yang termasuk dalam infrastruktur manajemen internal Heroku.

Keamanan dalam add-on

Heroku memberi pelanggannya kemampuan untuk membangun fitur tambahan aplikasi mereka menggunakan add-ons yang merupakan paket pihak ketiga yang menyelesaikan spesifik kebutuhan aplikasi; misalnya, caching, keamanan, dan sebagainya. Add-on ini adalah ditawarkan dan dikelola oleh perusahaan pihak ketiga dan menerapkan keamanan mereka sendiri mekanisme.

Pengembang aplikasi perlu mengevaluasi keamanan aplikasi mereka perlu saat menggunakan add-on ini dan periksa apakah mereka perlu membangun tambahan kerangka kerja keamanan untuk memastikan keamanan aplikasi mereka.

Mengamankan infrastruktur logging

Log memberi tahu kami banyak hal tentang keadaan aplikasi. Tergantung pada level logging diaktifkan dalam aplikasi, seseorang dapat mengumpulkan informasi penting dan mendalam tentang negara dari berbagai komponen aplikasi. Logging melayani kebutuhan kritis untuk pemecahan masalah dan menyelesaikan masalah aplikasi yang dihadapi dari waktu ke waktu. Karena itu, sangat kritis bahwa log itu sendiri dibuat aman.

Sangat penting bahwa aplikasi pelanggan menggunakan koneksi database terenkripsi, sembunyikan data penting sebelum masuk (kartu kredit, alamat, dan kata sandi), buat log berkala (bukan hanya satu log besar), dan menunjukkan titik transisi (beralih dari satu modul ke yang lain) dengan jelas. Persyaratan pencatatan ini menyimpan data aplikasi Anda aman dengan hanya mencatat informasi yang relevan dan tidak mengungkapkan data yang mungkin menjadi risiko keamanan bagi pelanggan Anda.

Platform Heroku menyediakan banyak opsi untuk berinteraksi dengan sistem, aplikasi, dan log API. Seseorang yang dapat memecahkan masalah aplikasi dapat memilih untuk menerima jenis-jenis file log ini secara terpisah dan kemudian menggunakan add-on untuk menganalisis log data atau tinjau log secara real time menggunakan klien Heroku. Ada beberapa add-on tersedia,

misalnya, Loggly, untuk menganalisis lebih lanjut file aplikasi dan memperolehnya wawasan bermanfaat tentang masalah yang dihadapi.

Secara umum, Heroku mengambil langkah-langkah untuk melindungi privasi pelanggan dan melindungi data yang disimpan dalam platform. Berbagai jenis perlindungan diaktifkan di Internet Platform Heroku termasuk otentikasi, kontrol akses, enkripsi transportasi data, Dukungan HTTPS untuk aplikasi pelanggan, dan kemampuan bagi pelanggan untuk mengenkripsi data tersimpan.

Keamanan jaringan

Platform Heroku memanfaatkan firewall untuk membatasi akses ke sistem dari eksternal jaringan dan antar sistem secara internal. Secara default, semua akses ditolak. Heroku mengelompokkan komponen serupa ke dalam grup keamanan jaringan yang unik, menyediakan akses ke sumber daya spesifik (port) dan protokol untuk digunakan. Ini memberikan jenis isolasi untuk platform di mana komponen yang tidak memiliki fungsi platform berbeda saling mengganggu.

Platform Heroku menggunakan teknik pencegahan DDoS, seperti TCP SYN cookie dan pembatasan tingkat koneksi, selain mempertahankan beberapa tulang punggung koneksi dan kapasitas bandwidth internal yang besar untuk memenuhi tuntutan lalu lintas jaringan aplikasi. Heroku menggunakan firewall terkelola untuk mencegah IP, MAC, atau spoofing ARP di jaringannya. Ini juga menggunakan sniffing paket lanjutan, termasuk hypervisor yang melarang lalu lintas dialihkan ke antarmuka yang tidak dimaksudkan untuknya.

Heroku memanfaatkan isolasi aplikasi, pembatasan sistem operasi, dan dienkripsi koneksi untuk mengelola lebih lanjut persyaratan keamanan di setiap level.

Heroku sangat merekomendasikan agar aplikasi pelanggan tidak terlibat dalam port pemindaian atau pelanggaran keamanan berbahaya lainnya dengan cara apa pun. Pemindaian port tidak diizinkan, dan setiap instance yang dilaporkan diselidiki dan dihentikan, yang menghasilkan akses diblokir.

Standar keamanan dan kepatuhan

Untuk sebagian besar, keamanan infrastruktur fisik yang mendasari Heroku bergantung pada kebijakan keamanan infrastruktur fisik Amazon. Untuk menghitung itu, Amazoncontinu terus mengelola risiko dan menjalani penilaian berulang untuk memastikan kepatuhan dengan standar industri. Operasi pusat data Amazon memiliki beberapa sertifikasi yang sesuai dengan kreditnya, termasuk, tetapi tidak terbatas pada, ISO 27001, SOC 1 dan 2, SSAE 16 / ISAW 3402, PCI, FISMA moderat, dan SOX.

Heroku menggunakan pusat data bersertifikasi ISO 27001 dan FISMA yang dikelola oleh Amazon, yang memiliki pengalaman bertahun-tahun dalam mengelola pusat data skala besar. AWS pusat data bertempat di fasilitas nondeskrip yang berisi fasilitas penting itu memiliki kemunduran yang luas dan kontrol perimeter tingkat militer, selain yang alami perlindungan batas. Akses fisik dikontrol secara ketat menggunakan state-of-the-art sistem keamanan.

Mengamankan permintaan web

Ketika seorang pengguna mengakses aplikasi Heroku, itu bisa karena berbagai alasan. Pengguna bisa mendaftar dengan aplikasi dan memasukkan informasi pribadi utama, atau mereka dapat membayar layanan menggunakan kartu kredit. Permintaan dari pengguna dalam kedua kasus ini akan membutuhkan browser untuk mengirim informasi pribadi yang idealnya harus disembunyikan atau dienkripsi selama transmisi. Untuk itu skenario, protokol kriptografi Secure Sockets Layer (SSL) datang ke kami menyelamatkan. SSL menyediakan enkripsi data end-to-end dan integritas untuk apa saja permintaan web. Setiap permintaan web yang mengirimkan data rahasia atau sensitif harus mengaktifkan SSL untuk transmisi aman data sensitif melalui jaringan.

Heroku juga mendukung pengiriman permintaan ke aplikasi web Anda menggunakan SSL untuk mengamankan komunikasi. Berdasarkan jenis URL yang didukung di aplikasi Anda, Anda dapat menggunakan piggyback SSL atau mengkonfigurasi kapabilitas SSL untuk kustom Anda aplikasi domain.

Piggyback SSL

Jika Anda menggunakan http: // <namaappname> .herokuapp.com sebagai URL untuk aplikasi, Anda dapat "membonceng" pada fasilitas SSL yang ditawarkan secara default pada Platform heroku. Anda dapat mengakses URL yang terkait dengan aplikasi web Anda menggunakan https: // spesifikasi protokol, dan Heroku akan mengenkripsi permintaan Anda secara default.

SSL untuk domain khusus

Heroku mendukung add-on endpoint SSL untuk mengaktifkan SSL untuk domain khusus Mengkonfigurasi SSL untuk domain khusus Anda adalah prosedur yang agak panjang, dan Anda akan membutuhkan intervensi pihak ketiga untuk menjalankannya. Untuk mendapatkan SSL berfungsi untuk aplikasi domain khusus Anda, Anda harus melakukan hal berikut:

1. Beli sertifikat SSL dari penyedia:

- **Sertifikat SSL yang ditandatangani sendiri**: Salah satu yang termudah dan paling murah cara pengamanan aplikasi web Anda adalah dengan menambahkan enkripsi SSL melalui sertifikat SSL yang ditandatangani sendiri. Biasanya, sertifikat ini bisa digunakan untuk mengamankan aplikasi pementasan atau non-produksi. A ditandatangani sendiri Sertifikat SSL mengimplementasikan enkripsi penuh. Namun, saat Anda menggunakannya sertifikat SSL yang ditandatangani sendiri dengan suatu aplikasi, browser menunjukkan a peringatan tentang ketidakpercayaan situs.
- **Beli sertifikat SSL dari vendor**: Pengembang Heroku dapat membeli sertifikat SSL dari perusahaan seperti DNSimple atau ikuti a serangkaian langkah untuk mendapatkan akses ke sertifikat SSL. Biasanya kamu bisa lakukan langkah-langkah berikut untuk mendapatkan sertifikat SSL:
    1. Hasilkan kunci pribadi menggunakan openSSL.

2. Buat Permintaan Penandatanganan Sertifikat (CSR) menggunakan pribadi kunci yang dihasilkan dari langkah pertama.
3. Kirim CSR yang dibuat pada langkah kedua ke penyedia SSL.

2. Tambahkan add-on Heroku titik akhir SSL ke aplikasi Anda:
   1. Untuk menggunakan SSL untuk aplikasi Heroku Anda, Anda harus mengonfigurasi SSL add-on endpoint sebagai berikut: **$ heroku addons: add ssl: endpoint Menambahkan ssl: titik akhir pada sampelapp ... selesai, v1 ($ 20 / bln)** Layanan ssl: endpoint dikenakan biaya $ 20 per bulan.
   2. Sekarang, Anda dapat menambahkan sertifikat, termasuk perantara sertifikat dan kunci pribadi ke titik akhir SSL dengan Heroku sertifikat: tambahkan perintah:

   $ heroku certs: tambahkan server.crt bundle.pem server.key Menambahkan SSL Endpoint ke contoh aplikasi ... selesai sampleapp sekarang dilayani oleh nanking-1234.herokussl.com. Detail sertifikat: Kedaluwarsa pada: 2013-10-31 09:30:00 GMT Penerbit: C = AS; ST = CA; L = SF; O = Heroku; CN = www.sampleapp.com Mulai Pada: 2012-11-01 09:30:00 GMT ...

   URL titik akhir yang ditetapkan untuk aplikasi Anda dalam contoh ini adalah nanking-1234.herokussl.com.
3. Rincian konfigurasi titik akhir SSL dapat ditinjau the heroku: certs command sebagai berikut:

   sertifikat $ heroku Nama Umum Endpoint Berakhir Tepercaya
   ------------------- ---------------- ------------- - ----
   nanking-1234.herokussl.com www.sampleapp.com 2013-10- 31 09: 30: 00GMT Salah
4. Informasi terperinci tentang sertifikat SSL dapat diambil dengan menggunakan itu certs: info perintah sebagai berikut: sertifikat **$ heroku: info Mengambil info SSL Endpoint nanking-1234.herokussl.com untuk sampleapp ... selesai Detail**

sertifikat: Berakhir pada: 2013-10-31 09: 30: 00GMT Penerbit: C = AS; ST = CA; L = SF; O = Heroku; CN = www.sampleapp.com Mulai Pada: 2012-11-01 09: 30: 00GMT Subjek: C = AS; ST = CA; L = SF; O = Heroku; CN = www.sampleapp.com Untuk URL titik akhir herokussl.com, kunjungi melalui https, misalnya, https://nanking-1234.herokussl.com.

5. Unggah sertifikat SSL ke Heroku.
6. Ubah pengaturan DNS aplikasi untuk referensi URL titik akhir SSL baru. Setelah titik akhir SSL diberikan dan sertifikat SSL dikonfirmasi, Anda perlu mengonfigurasi DNS untuk domain khusus Anda untuk mengaktifkan perutean ke aplikasi Anda.

Setelah SSL dikonfigurasikan untuk domain khusus Anda, permintaan pengguna untuk sumber daya adalah dienkripsi selama komunikasi antara browser dan aplikasi web.

Alat keamanan aplikasi

Ada beberapa alat keamanan yang dapat membantu pengembang mengidentifikasi masalah keamanan di aplikasi web mereka. Di bagian ini, kami mengeksplorasi dua alat seperti itu – wwwhisper dan alat keamanan kertas timah yang didukung sebagai add-on pada platform Heroku.

wwwhisper

Sebagai pengembang aplikasi, Anda dapat menggunakan add-on seperti wwwhisper untuk mengotorisasi akses ke RoR atau aplikasi Heroku berbasis Rack lainnya. Administrator dari aplikasi dapat menggunakan antarmuka web untuk menentukan alamat email dari para pengguna yang diizinkan mengakses aplikasi Anda. wwwhisper memberikan yang halus dan mulus kontrol akses ke aplikasi Heroku Anda

wwwhisper memanfaatkan Persona – sistem masuk lintas peramban untuk Web (didukung di semua peramban modern) –yang menghapus perlunya kata sandi khusus situs menetapkan kepemilikan alamat email tertentu.

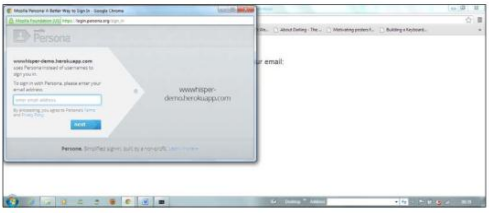
Middleware Rack menyediakan integrasi dengan layanan keamanan wwwhisper. Sebagai akibatnya, biaya integrasi dijaga agar tetap minimum, dan tidak perlu dilakukan kode aplikasi berubah atau panggil API wwwhisper apa pun.

Contoh aplikasi wwwhisper

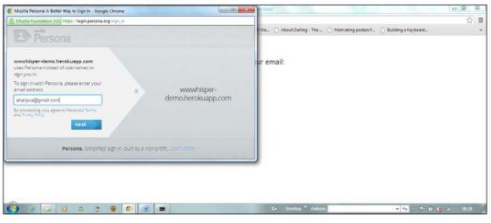
Ada contoh aplikasi wwwhisper yang tersedia di wwwhisper-demo.herokuapp. com yang menunjukkan penggunaan pengaya ini. Meskipun semua orang dapat mengakses situs ini dengan secara default, Anda masih harus masuk untuk mengakses aplikasiUntuk mengakses aplikasi, kunjungi URL yang disebutkan. Anda akan melihat yang berikut di layar Anda:

Sekarang, lakukan langkah-langkah berikut:

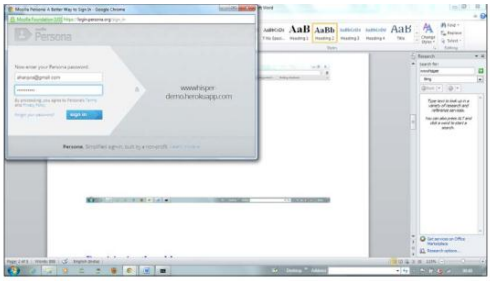
1. Klik Masuk untuk diarahkan ke halaman login Persona, seperti yang ditunjukkan di tangkapan layar berikut:



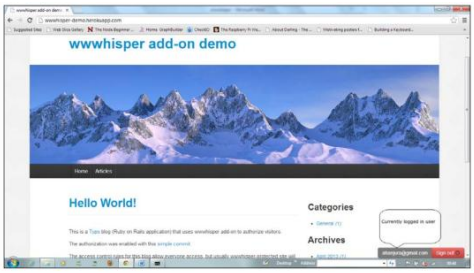
2. Jika Anda sudah memiliki akun di Persona, masukkan alamat email sebagai ditunjukkan pada tangkapan layar berikut:



3. Masukkan kata sandi untuk akun Persona seperti yang ditunjukkan pada tangkapan layar berikut:



4. Anda akan diarahkan ke aplikasi demo wwwhisper. Saat ini pengguna yang masuk ditampilkan di sudut kanan bawah layar:



Dapatkan wwwhisper

Anda dapat mengunduh wwwhisper ke aplikasi Heroku melalui Heroku CLI. Untuk meninjau fitur terperinci dari wwwhisper, kunjungi https://addons.heroku.com/wwwhisper. Layanan wwwhisper menyimpan email pengguna yang diizinkan mengakses email Anda aplikasi. Email-email ini hanya digunakan untuk mengotorisasi akses ke aplikasi dan tidak untuk tujuan komunikasi atau pemasaran. wwwhisper tidak bertahan riwayat atau pola penelusuran aplikasi untuk pengguna aplikasi.

Dianjurkan agar Anda menggunakan aplikasi yang dilindungi wwwtidak di HTTPS. wwwhisper hanya mengizinkan akses ke konten yang dilayani oleh aplikasi Heroku. Jika pengembang aplikasi menempatkan konten sensitif pada server eksternal seperti Amazon S3 yang tidak memerlukan otorisasi, wwwhisper tidak dapat membatasi akses ke konten tersebut.

wwwhisper tersedia dalam tiga rencana – starter, dasar, dan plus. Kamu bisa memilih salah satu paket berdasarkan kebutuhan

bisnis Anda. Dengan pemula, pengguna tunggal gratis kit, Anda dapat menangani hingga 10.000 permintaan otorisasi per bulan. Rencana dasar menyediakan akses ke hingga 10 pengguna dan 1 juta permintaan otorisasi per bulan dan paket plus yang lebih tinggi memungkinkan hingga 100 pengguna dan 10 juta permintaan otorisasi per bulan.
Untuk menambahkan wwwhisper, ketikkan yang berikut:

$ heroku addons: add wwwhisper [--admin = your_email]

Parameter --admin adalah parameter opsional yang menentukan siapa yang seharusnya awalnya diizinkan mengakses aplikasi. Tanpa opsi admin, Heroku email pemilik aplikasi digunakan. Anda dapat menggunakan situs admin wwwhisper untuk memberikan akses ke pengguna lain.

Untuk menambahkan paket wwwhisper tertentu, ketikkan yang berikut:

$ heroku addons: add wwwhisper: plus

Setelah add-on ditambahkan ke aplikasi Anda, konfigurasi WWWHISPER_URL pengaturan akan tersedia dalam konfigurasi aplikasi. URL ini dapat digunakan untuk berkomunikasi dengan layanan wwwhisper. Saat mengembangkan aplikasi, aplikasi e-mail dari pengguna yang diautentikasi dapat diambil dari variabel lingkungan Rack bernama REMOTE_USER. Anda dapat memverifikasi ini menggunakan Heroku berikut config: get command:

$ heroku config: dapatkan WWWHISPER_URL https: // pengguna: kata sandi @ domain

Menghapus wwwhisper

Anda dapat menghapus wwwhisper melalui CLI. Ini akan menghancurkan data terkait. Pertimbangkan baris perintah berikut:

$ heroku addons: hapus wwwhisper

Mengaktifkan wwwhisper di aplikasi Anda

Mengaktifkan wwwhisper di aplikasi Anda Pengembang perlu menyertakan alat keamanan wwwhisper dalam konfigurasi untuk mengaktifkan penggunaannya oleh aplikasi. Langkah-langkah untuk mengaktifkan alat di berbagai jenis aplikasi adalah sebagai berikut.

Untuk aplikasi Ruby, kita perlu menjalankan langkah-langkah berikut:

1. Tambahkan entri berikut ke dalam Gemfile aplikasi Ruby: gem 'rack-wwwhisper', '~> 1.0'
2. Perbarui dependensi aplikasi dengan bundler menggunakan perintah berikut:$ bundle install.

   Tempatkan baris berikut di akhir config / environment / production.rb untuk a Aplikasi rel:
   config.middleware.insert 0, "Rack :: WWWhisper"
   Baris ini akan menjadikan wwwhisper middleware pertama dalam rantai middleware Rack.

Untuk aplikasi berbasis Rack lainnya

Untuk aplikasi berbasis Rack, tambahkan dua baris berikut ke config.ru: membutuhkan 'rack / wwwhisper' gunakan Rack :: WWWhisper

Poskan pemberdayaan wwwhisper

Setelah wwwhisper diaktifkan di aplikasi Anda, Anda dapat memverifikasi apakah itu benar-benar berfungsi dengan memasukkan URL https://yourapp-name.herokuapp.com/ di browser Anda. Anda harus disajikan halaman login. Masuk dengan email Anda dan kemudian kunjungi https://yourapp-name.herokuapp.com/wwwhisper/admin/ untuk menentukan lokasi yang dapat diakses oleh pengunjung dan mereka (jika ada) yang harus terbuka untuk semua orang.

BAB 8
MENGATASI MASALAH APLIKASI HEROKU

Dalam beberapa bab terakhir, kami belajar cara membangun, menyebarkan, dan menjalankan aplikasi Heroku. Kami juga mengeksplorasi cara menguji aplikasi Anda secara lokal menggunakan mandor dan setelah itu selesai, bagaimana cara sukses menyebarkan aplikasi Anda ke platform Heroku. Kami juga telah melihat berbagai opsi yang tersedia untuk menjalankan dan mengelola aplikasi kami melalui dasbor Heroku – toko serba ada untuk berbagai kebutuhan manajemen aplikasi. Sekarang, kita berada pada titik di mana aplikasi aktif dan berjalan. Tiba-tiba aplikasi berhenti bekerja atau mulai membuat kesalahan pada Anda. Saatnya untuk memecahkan masalah apa yang salah. Bab ini membahas aspek yang sangat penting dari penerapan aplikasi di Heroku, yaitu, pemecahan masalah aplikasi ketika mulai bertingkah buruk.

Dalam bab ini, kita akan membahas topik-topik berikut tentang pemecahan masalah di Heroku :

- Menggunakan infrastruktur logging Heroku untuk memecahkan masalah aplikasi Anda
- Memecahkan masalah aplikasi Heroku menggunakan alat dan teknik yang tersedia
- Mempelajari cara memecahkan masalah downtime aplikasi
- Memahami masalah dan cara yang paling umum ditemui mengatasinya

- Memanfaatkan fitur pemeriksaan produksi untuk mengidentifikasi masalah sebelum muncul
- Menggunakan konfigurasi yang disarankan untuk aplikasi Anda
- Mengelola jendela perawatan
- Memahami kesalahan Heroku dan cara mengatasinya menggunakan halaman kesalahan khusus

Kebutuhan untuk pemecahan masalah

Heroku adalah salah satu platform paling tangguh untuk mengembangkan aplikasi di cloud saat ini. Ini menawarkan perpaduan unik dari layanan platform dari penyebaran aplikasi ke pemantauan, dan dari merutekan permintaan web hingga mencatat semua yang terjadi dalam aplikasi. Heroku sangat tersedia dan dapat diukur dan idealnya cocok untuk penyebaran aplikasi web dari segala kompleksitas. Ini juga menawarkan pengembang berbagai pilihan dalam hal bahasa pemrograman untuk digunakan untuk menulis aplikasi web.

Namun, seperti lingkungan komputasi yang kuat lainnya, Heroku terkadang menghadapi masalah operasional atau terkait aplikasi yang menyebabkan aplikasi produksi atau pengembangan Anda berperilaku tidak menentu. Masalah dapat berasal dari salah satu aspek berikut dari aplikasi Heroku Anda :

- Masalah operasional
  - Masalah konfigurasi
  - Masalah basis data
  - Meminta masalah antrian
  - Masalah skalabilitas
  - Masalah DNS
  - Perpustakaan tidak ada
- Masalah terkait memori
- Masalah khusus bahasa atau kerangka kerja
- Masalah kontrol akses dan kerentanan keamanan
- Masalah penyebaran
- Masalah komunikasi

Cara yang paling umum bagi Heroku untuk memberi tahu Anda bahwa ada masalah adalah melalui log Heroku. Biasanya, Heroku menambahkan informasi kesalahan khusus ke log Anda untuk membantu Anda memecahkan masalah. Itu mengumpulkan log dari berbagai sumber dan menunjukkan kepada Anda pandangan terkonsolidasi dari perilaku aplikasi. Anda dapat memfilter area log tertentu yang Anda temukan relevan dengan pemecahan masalah Anda. Beberapa fasilitas berguna lainnya yang ditawarkan Heroku untuk masalah pemecahan masalah adalah pemantauan proses dan perintah restart proses otomatis, pemeriksaan aplikasi, dan banyak lagi. Ada beberapa teknik yang bisa digunakan untuk memecahkan masalah yang ada dan menyelesaikan masalah yang mengganggu dengan cepat. Lebih sering daripada tidak, kode kesalahan Heroku dalam file log aplikasi menambahkan informasi yang sangat berharga untuk memungkinkan pemecahan masalah yang efektif. Properti tahan erosi Heroku membantu Anda menjalankan aplikasi tanpa downtime ditambah dengan penggunaan sumber daya yang efektif dan penskalaan layanan yang transparan.

Jendela Anda ke aplikasi yang sedang berjalan – Log

Log adalah file pada disk Anda (atau data dalam memori) yang ditulis oleh aplikasi saat dijalankan. Menulis logfile diaktifkan dengan menggunakan berbagai kerangka kerja logging di aplikasi Anda (misalnya, log4j untuk Java) dan juga log sistem sistem sendiri yang melacak berbagai peristiwa sistem. Basis data, sistem antrian, dan hampir semua komponen aplikasi memiliki log. Apa pun yang ditulis aplikasi Anda ke kesalahan standar atau aliran output biasanya ditangkap dalam log aplikasi.

Logging adalah sumber daya yang paling dikenal luas untuk pemecahan masalah aplikasi dalam penyebaran apa pun. Log lebih sering daripada bukan sumber daya pertama dan paling kritis yang digunakan untuk memahami status aplikasi ketika crash atau mulai berperilaku tidak menentu.

Log dirancang untuk mengumpulkan informasi dari elemen yang berbeda dari aplikasi Anda dan memberikan tampilan waktu memerintahkan tentang bagaimana aplikasi dieksekusi secara real time dan log melacak peristiwa saat itu terjadi.

Sedikit lagi tentang Logplex - sistem logging Heroku

Sistem Logplex membentuk dasar dari infrastruktur logging Heroku. Ini mengumpulkan dan mendistribusikan pesan log dari aplikasi dan bagian lain dari infrastruktur Heroku. Pengembang dapat melacak aliran aplikasi menggunakan pesan log ini dan mempersempit cakupan kesalahan yang ditemui ke subset seluruh kode aplikasi. Pada sistem Heroku, entri log ini tersedia melalui API publik dan alat baris perintah Heroku. Karena sifat platform Heroku yang terdistribusi, mencoba mengakses pesan log secara manual di berbagai komponen aplikasi dan membuat masuk akal dari hal-hal tersebut praktis tidak mungkin. Anda tidak bisa mendapatkan tampilan terpadu dari keseluruhan perilaku sistem jika Anda menyusun pesan-pesan ini secara manual dan mencoba memecahkan masalah. Sistem logging terintegrasi Logplex memberikan alternatif yang kuat dalam kasus ini.

Sumber dan saluran air

Sistem Logplex bertindak seperti saluran yang merutekan pesan dari sumber entri log, yaitu, produsen pesan log (misalnya, aplikasi yang berjalan pada dyno, platform Heroku itu sendiri) untuk masuk ke saluran pembuangan entri, yaitu, konsumen dari mencatat pesan (misalnya, sistem arsip atau pemroses pos yang melakukan penambangan informasi).

Batas pesan

Sistem Logplex tidak menyediakan penyimpanan untuk seluruh data log yang dihasilkan, tetapi hanya menyimpan data terbaru yang cukup baik untuk mengekstrak informasi yang relevan dari aplikasi dijalankan. Biasanya, ukuran data ini sekitar 1.500 pesan log gabungan. Jika Anda memiliki kebutuhan bisnis untuk menyimpan lebih dari batas yang telah ditentukan, Anda harus

memilih layanan penyimpanan alternatif, misalnya, saluran syslog untuk mengarsipkan semua pesan. Selain itu, ekosistem pengaya Heroku memungkinkan Anda untuk bekerja dengan lebih banyak alat logging yang canggih untuk menyimpan, menganalisis, dan membuat aplikasi cerdas berbasis log untuk memberi tahu Anda dalam kondisi yang luar biasa / kesalahan.

Mengambil Log Heroku

Heroku log messages can be reviewed by typing the heroku logs command on the prompt (after you have logged in to Heroku). Consider the following code snippet :

```
$ heroku logs
2013-04-26T15:23:21.278990+00:00 app[web.1]: Rendered includes/_header (0.1ms)
2013-04-26T15:23:21.298234+00:00 app[web.1]: Completed in 74ms (view: 31, DB: 40) | 200 OK
[http://xxx.heroku.com/]
2013-04-26T15:13:46.723498+00:00 heroku[router]: at=info method=GET path=/posts
host=xxx.herokuapp.com fwd="x.x.x.x" dyno=web.1 connect=1ms service=18ms status=200 bytes=975
2013-04-26T15:13:47.893472+00:00 app[worker.1]: 2 jobs processed at 16.6761 j/s, 0 failed ...
```

Output log sebelumnya termasuk entri log dari salah satu dinamika web aplikasi, router Heroku, dan salah satu dari dinamika pekerja aplikasi. Perintah log mengambil 100 entri log secara default.

Mendapatkan pesan log 'n' terakhir

Untuk mengambil sejumlah pesan log, gunakan opsi baris perintah --num atau -n. Misalnya, untuk mengambil 300 baris terakhir data log, ketikkan $ heroku log -n 300 atau $ heroku logs -num 300.

Mendapatkan pesan log langsung

Jika Anda perlu mengamati aplikasi langsung dan memecahkan masalah saat terjadi, menggunakan perintah ekor Heroku adalah ide yang bagus. Perintah ekor memberikan tampilan waktu-nyata dari pesan log dari aplikasi dan membantu Anda melihat lebih dekat peristiwa yang terjadi. Untuk melihat pesan log yang dihasilkan, ketik $ heroku log --tail atau $ heroku log -t.

Beberapa kerangka kerja memungkinkan buffer log, yaitu, mengumpulkan satu set pesan log sebelum mengirimkannya ke output standar untuk ditampilkan. Jika Anda tidak memerlukan perilaku ini dan ingin melihat log ditampilkan secara waktu nyata tanpa buffering, Anda perlu mengonfigurasi aplikasi Anda untuk menonaktifkan buffering. Kerangka kerja yang berbeda memiliki cara yang berbeda untuk melakukannya; misalnya, di Ruby, Anda perlu menambahkan potongan kode berikut Anda config.ru fie untuk menonaktifkan buffering dan mengizinkan log muncul secara real time:

$stdout.sync = true

Juga, penting untuk dipahami bahwa banyak kerangka kerja yang pesan log-nya dialihkan ke tujuan selain output standar sistem mereka. Dalam kasus seperti itu, mudah untuk bingung dan mulai mencari pesan log di mana tidak ada. Jika Anda ingin mengubah perilaku default, Anda perlu mengkonfigurasi tujuan logging yang benar untuk aplikasi Anda.

Menyiapkan level logging

Secara default, level logging pada platform Heroku diatur ke INFO. Jika Anda menginginkan DEBUG level logging, Anda harus secara manual mengatur level ke DEBUG sebagai berikut:

konfigurasi heroku: tambahkan LOG_LEVEL = DEBUG
heroku addons: perbarui logging: diperluas
log heroku –tail

Kemudian, jalankan aplikasi Anda dan bersiap untuk memukul halaman dengan kesalahan. Anda harus bisa untuk melihat kesalahan di konsol Anda.

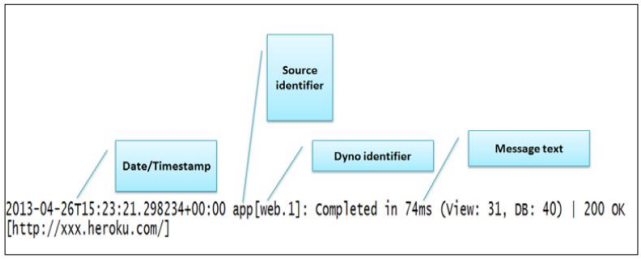
Membedah pesan log Heroku

Penting bagi Anda untuk memahami arti setiap komponen dari pesan log. Ini akan membantu Anda mempersempit analisis Anda ke segmen log yang tepat dan memahami pesan log secara lengkap dan cepat.

Format pesan log Heroku adalah sebagai berikut:

sumber timestamp [dyno identifier]: pesan log

Contoh pesan log adalah sebagai berikut:



Deskripsi sintaks sebelumnya adalah sebagai berikut:

- Stempel waktu berisi tanggal dan waktu ketika pesan log itu dihasilkan oleh suatu komponen. Waktu yang ditunjukkan memiliki presisi mikrodetik.
- Identifir sumber menunjukkan bagian mana dari infrastruktur Heroku log pesan berasal dari. Ini bisa jadi aplikasi jika berasal dari salah satu aplikasi dyno, atau bisa juga Heroku jika berasal dari yang mendasarinya Komponen infrastruktur Heroku seperti router permintaan Heroku.
- Dyno identifir mewakili nama atau proses dino tertentu membuat pesan log ini.
- Pesan log aktual berisi informasi yang ditulis oleh aplikasi atau Heroku atau komponen lain dari sistem. Pesan log bisa sampai Berukuran 1024 byte. Pesan yang lebih panjang terpotong.

Jenis pesan log

Log di Heroku dapat berasal dari banyak sumber di platform Heroku:

- **Pesan log yang dihasilkan aplikasi**: Pesan log ini biasanya berasal dari dalam aplikasi dan server Anda, ditambah perpustakaan yang digunakan oleh aplikasi. Ini biasanya adalah pesan tingkat atas dan lebih dekat dengan pengguna. Jenis-jenis pesan ini menyediakan informasi tingkat tinggi tentang cacat kode aplikasi Anda.

- **Log sistem yang dihasilkan Heroku**: Ini adalah pesan yang dihasilkan oleh infrastruktur Heroku. Biasanya, pesan-pesan ini adalah tentang tindakan yang dilakukan oleh platform atas nama aplikasi Anda. Jenis pesan yang akan diklasifikasikan dalam tipe ini adalah dyno reboot pesan atau pesan kesalahan sistem yang dilempar oleh infrastruktur Heroku yang mendasarinya.
- **Log Antarmuka Pemrograman Aplikasi (API):** Log pesan di bawah kategori ini mencakup yang terkait dengan beberapa langkah administratif yang diambil oleh pengembang saat mengerjakan aplikasi, misalnya, menyebarkan aplikasi atau meningkatkan skala pekerja web.

Pemilih log

Sistem logging Heroku mengumpulkan dan mendistribusikan pesan log dari sumber yang berbeda. Ini memberikan pandangan yang tidak jelas tentang peristiwa log yang dihasilkan di seluruh infrastruktur Heroku dan aplikasi Anda. Seringkali, untuk masalah pemecahan masalah, hanya beberapa jenis pesan yang penting. Dalam situasi seperti itu, pencocokan log disediakan oleh sistem logging Heroku datang untuk menyelamatkan. Untuk mengambil log dengan sumber tertentu atau dino tertentu atau keduanya, Heroku memberikan argumen --source (-s) dan --ps (-p).

Contoh pencocokan log

Mari kita lihat beberapa contoh dari pemilih log yang didukung di Heroku berikut ini:

- Untuk mengekstrak pesan yang berisi informasi terkait komponen router, ketikkan perintah berikut:

  $ heroku log --ps router

  2013-05-07T06: 12: 06.209875 + 00: 00 heroku [router]: at = info method = DAPATKAN path = / artikel / css / sample.css host = sampleapp.herokuapp. com fwd = "207.207.207.207" dyno = web.2 terhubung = 1ms layanan = 12ms status = 200 byte = 12

  2013-05-07T06: 12: 06.209875 + 00: 00 heroku [router]: at =

info method = GET path = / artikel / instal host = sampleapp.herokuapp. com fwd = "207.207.207.207" dyno = web.3 terhubung = layanan 1ms = status 12ms = 200 byte = 1403

- Untuk mengekstrak pesan yang dikirim oleh sumber aplikasi, ketikkan perintah berikut:

  $ heroku log --sumber aplikasi

  Aplikasi 2013-05-07T02: 12: 47.209875 + 00: 00 [web.1]: Rendered common / _ search.html.erb (1.0ms) 2013-05-07T02: 12: 47.209875 + aplikasi 00: 00 [web. 1]: Menyelesaikan 200 OK dalam 48ms (Views: 23.7ms | ActiveRecord: 24.3ms)

  2013-05-07T02: 12: 47.209875 + 00: 00 aplikasi [pekerja.1]: [Pekerja (tuan rumah: 465bf64e-61c8-46d3-b480-362bfd4ecff9 pid: 1)] 1 pekerjaan diproses pada 33.0330 j / s, 0 gagal ... 2013-05 07T02: 13: 01.209875 + 00: 00 app [web.6]: Memulai DAPATKAN "/ artikel / instal" untuk 4.1.81.209 pada 2013-05-07 02 : 13: 01 +0000

- Untuk mengekstrak pesan yang dikirim oleh aplikasi, tetapi berasal dari dino tertentu, ketikkan perintah berikut:

  $ heroku log --source app --ps pekerja

  2013-05-07T02: 13: 59.239875 + 00: 00 aplikasi [pekerja.1]: [Pekerja (host: 260bf64e-61c8-46d3-b480-362bfd4ecff9 pid: 1)] Artikel # show_result selesai setelah 0,0421

  2013-05-07T02: 13: 59.239875 + 00: 00 aplikasi [pekerja.1]: [Pekerja (tuan rumah: 260cf64e-61c8-46d3-b480-362bfd4ecff9 pid: 1)] 3 pekerjaan diproses pada 21,6842 j / s, 0 gagal ...

- Untuk mendapatkan aliran aktual dari keluaran yang dipasangkan, Anda dapat menggunakan –tail argumen baris perintah; ketik perintah berikut:

  $ heroku log --source heroku –tail

  Perintah ini akan menyatukan pesan untuk sistem Heroku dan menampilkan aliran pesan log sistem secara terus menerus ketika aplikasi terus berjalan.

Dapatkan lebih banyak dari logging - logging lainnya alat

Heroku menyediakan serangkaian pengaya untuk menyediakan layanan manajemen log dan analitik canggih untuk aplikasi Anda. Anda bisa menggunakan satu atau lebih dari add-on yang tersedia dan membuat kemampuan pemantauan aplikasi yang unggul untuk aplikasi Anda. Beberapa dari layanan ini adalah sebagai berikut:

- **Papertrail**: Ini adalah layanan manajemen log real-time pembunuh yang memungkinkan Anda memecahkan masalah kesalahan aplikasi, membuat peringatan pemantauan, dan melacak tiket layanan dan kehilangan email. Papertrail menyediakan akses mudah melalui alat baris perintah, API, atau browser web. Ini juga terintegrasi dengan alat-alat canggih seperti Elastic Mapreduce, S3, dan Campire selain memiliki kemampuan untuk mencari basis data lansiran aplikasi dan acara yang terus meningkat untuk mendapatkan informasi penting tentang aplikasi tersebut.
- **Logentries**: Ini adalah sistem manajemen log pembangkit tenaga listrik lain yang memungkinkan pengguna mendapatkan notifid tentang kesalahan Heroku kritis termasuk pemberitahuan seluler. Logentries menyediakan visualisasi untuk memudahkan debugging log Heroku konsolidasi, menunjuk secara spesifik pada kesalahan kritis, di samping memberikan kemampuan untuk mengekspor / mengunduh log ke mesin Anda sendiri.
- **Loggly**: Ini adalah layanan manajemen log online solid yang digunakan oleh lebih dari 3.000 pelanggan, yang tidak memerlukan konfirmasi, memiliki API REST yang kaya untuk membangun aplikasi khusus berdasarkan informasi pencatatan, dan memungkinkan pencarian (termasuk pencarian wildcard) seluruh penyebaran Heroku Anda untuk masalah spesifik. Loggly memberikan kemampuan analitik aplikasi yang luar biasa untuk mengukur kinerja selain memiliki dukungan debugging aplikasi waktu-nyata untuk memecahkan masalah saat terjadi. Melalui serangkaian alat yang kaya seperti dasbor dan bagan,

Loggly membantu Anda mempersempit bagian yang salah dari aplikasi Anda dengan mudah.

- **Flydata**: Ini adalah layanan penyimpanan log lain yang berguna untuk aplikasi Heroku Anda yang memungkinkan Anda mencadangkan seluruh log Heroku Anda ke penyimpanan S3 Amazon, pada saat yang sama memungkinkan analisis data log pada platform Redshift Amazon. Itu juga mengirimi Anda email yang memberitahukan Anda tentang kesalahan Heroku kritis yang ditemukan di logfies Anda.
- **Keen IO**: Keen IO adalah platform yang dapat diskalakan dan digerakkan oleh analitik yang memberikan kemampuan untuk melacak peristiwa tertentu dalam siklus hidup aplikasi Anda apakah itu transaksi pembelian atau registrasi pengguna baru atau kesalahan Heroku kritis yang dilemparkan oleh aplikasi Anda. Tertarik IO menggunakan JSON untuk menyimpan data acara aplikasi Anda dan menyediakan API permintaan yang sangat ramah pengguna untuk mendapatkan informasi yang Anda butuhkan. IO yang tajam memungkinkan Anda mengekstrak dan mem-port data data aplikasi Anda dengan mudah serta memberikan metrik dan bagan yang dapat disematkan yang dapat Anda lihat di dalam aplikasi Anda.

  Jika salah satu dari alat atau layanan eksternal ini tidak dapat dikonsumsi pesan yang dihasilkan oleh Logplex, mereka dapat kehilangan beberapa di antaranya pesan. Namun, Logplex tidak memasukkan entri peringatan kapan saja itu harus meninggalkan pesan karena downtime layanan yang dikonsumsi.

Teknik untuk memecahkan masalah aplikasi Anda

Ada berbagai jenis kesalahan yang bisa Anda temui di platform Heroku saat mengerjakan aplikasi Anda. Bagian ini menjelaskan berbagai jenis masalah yang bisa dihadapi dan cara mengatasinya dengan menggunakan teknik dan alat yang telah teruji waktu.

Memecahkan masalah downtime aplikasi

Tidak peduli seberapa tersedia aplikasi Anda, downtime adalah kenyataan dari aplikasi produksi. Aplikasi dapat berhenti berfungsi karena berbagai alasan. Anda dapat memiliki desain yang buruk yang menyebabkan permintaan macet atau tidak cukup penyimpanan, atau jaringan lebih lambat yang menyebabkan kinerja aplikasi yang buruk. Seringkali, itu tidak terlalu buruk dan Anda mungkin harus me-restart aplikasi untuk menggunakan versi baru aplikasi Anda atau database baru. Semua situasi ini dapat menyebabkan downtime aplikasi. Ketika downtime adalah yang direncanakan, semuanya lebih baik, tetapi bagaimana jika downtime tidak beralasan dan tidak disengaja.

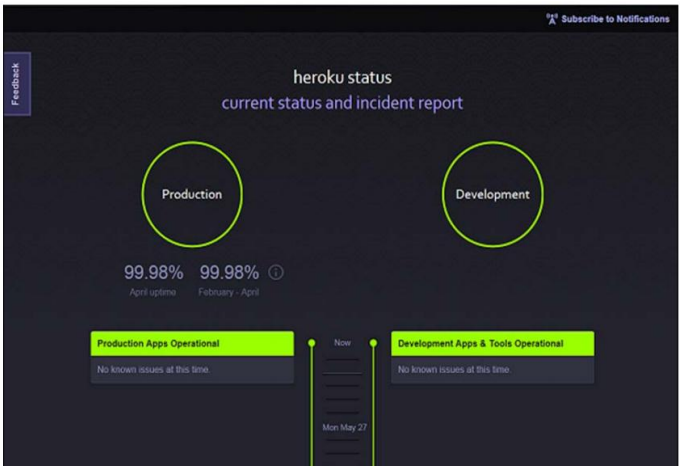
Seperti ditunjukkan sebelumnya, kesalahan bisa dari aplikasi atau platform Heroku di bawahnya. Kesalahan terkait aplikasi dapat berarti kesalahan logis atau runtime dalam kode sumber atau bisa juga salah satu kesalahan platform Heroku. Untuk kasus terakhir,Kode kesalahan Heroku yang muncul di log aplikasi dapat membantu Anda memecahkan masalah tertentu. Jika log menunjukkan salah satu kode kesalahan Heroku, dokumentasi kode kesalahan di https:// devcenter.heroku.com /articles/error-codes adalah sumber daya yang sangat bagus untuk dilihat.

Cara lain untuk menghilangkan kesalahan adalah dengan mencoba mereplikasi skenario kesalahan dengan memulai ulang aplikasi dan menjalankan perintah ekor untuk melacak pesan log yang dihasilkan karena tindakan pengguna terus menerus pada aplikasi secara real time. Anda dapat memulai ulang aplikasi dengan mengetikkan baris perintah berikut:

$ heroku restart

Secara umum, sangat masuk akal untuk meninjau status Heroku lingkungan sebelum menggunakan aplikasi Anda. Anda dapat mengunjungi halaman status Heroku di http://status.heroku.com untuk mendapatkan pandangan tentang masalah terbaru yang dihadapi pada lingkungan produksi dan pengembangan di Heroku.

Tangkapan layar berikut menunjukkan halaman status Heroku yang berisi informasi tentang berbagai lingkungan dan laporan tentang kesalahan yang ditemui:



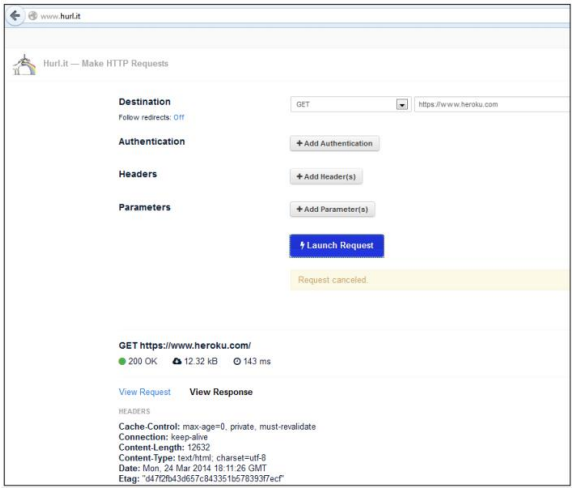
Heroku menyediakan banyak fitur menarik yang membantu Anda membatasi waktu henti aplikasi selama jendela perawatan yang direncanakan:

- Banyak pemecahan masalah dilakukan untuk Anda oleh platform itu sendiri dalam situasi di mana intervensi manual akan diperlukan sebaliknya. Misalnya, tidak seperti kebanyakan platform berbasis server, manajer dinamo Heroku menjaga aplikasi Anda tetap berjalan terlepas dari kegagalan yang terputus-putus. Heroku restart dinyalakan Anda secara otomatis dan mengirimkan dino Anda ke host lain secara transparan dan secara instan setiap kali terjadi kegagalan pada perangkat keras yang mendasarinya.
- Heroku secara internal mengelola setiap pemutakhiran atau tambalan yang diperlukan untuk lingkungan operasinya tanpa berdampak pada dinamika berjalan kecuali untuk memulai ulang saat diperlukan.
- Basis data yang berjalan pada basis data PostgreSQL dikelola sepenuhnya dan dipantau oleh tim platform Heroku dan setiap kegagalan perangkat keras ditangani sepenuhnya oleh tim basis data Heroku. Tidak diperlukan intervensi dari pengembang aplikasi.

Men-debug permintaan HTTP dan API

Jika Anda tidak dapat membuat permintaan HTTP ke beberapa URL, mungkin ada banyak alasan kegagalannya. Salah satu cara mudah untuk mendapatkan tampilan terperinci dari respons HTTP dari server dan mengapa halaman yang diminta tidak dapat dilayani adalah dengan menggunakan alat hurl berbasis web. Hurl membuat permintaan HTTP, secara opsional membiarkan Anda mengatur informasi tajuk dan memberi Anda detail respons. Anda dapat menggunakan alat ini untuk menguji bahkan API Anda yang membuat permintaan HTTP ke beberapa server mengharapkan server untuk melayani beberapa data kembali.

Anda juga dapat menggunakan curl – alat baris perintah untuk tujuan yang sama. Pertimbangkan tangkapan layar berikut:



Untuk menggunakan ikal setelah Anda menginstalnya, ketikkan baris perintah berikut:

$ curl -v http://helloworld.herokuapp.com/

Dalam hal ini, Anda dapat dengan mudah mengetahui apakah masalahnya ada pada aplikasi Anda atau dengan konfigurasi domain kustom Anda.

Ada banyak cara lain untuk menangani permintaan HTTP yang salah di Heroku. Anda dapat membatalkan permintaan yang berjalan lama dengan mengonfigurasi batas waktu permintaan atau

memindahkannya ke pekerjaan latar belakang, misalnya, mengunggah atau mengunduh dokumen berukuran besar atau membuat beberapa antrian di belakang penyeimbang beban di mana satu antrian untuk permintaan yang bergerak lambat dan lainnya untuk permintaan cepat.

Memvalidasi pembentukan proses Anda

Terkadang, respons aplikasi tampaknya sangat lambat karena dino yang ada kewalahan oleh terlalu banyak permintaan. Akibatnya, permintaan diantri menunggu waktu dyno. Masalahnya dapat diatasi dengan menyesuaikan formasi dyno dengan meningkatkan jumlah dyno yang menangani aplikasi Anda pada fl dengan menggunakan perintah skala heroku berikut:

web skala $ heroku = 2

Perintah ini menciptakan dua contoh lagi dari dyno web untuk menangani req tambahan Anda.

Memeriksa basis data Anda

Jika Anda mencurigai bahwa basis data aplikasi Anda adalah pihak yang salah yang menyebabkan waktu respons lambat, Anda dapat melihat status permintaan untuk aplikasi Anda menggunakan perintah heroku pg: ps. Perintah ini akan mencantumkan semua pertanyaan yang saat ini dijalankan terhadap basis data aplikasi Anda.

Jika Anda mengalami masalah yang berkaitan dengan memori, cobalah untuk mencari kueri yang menganggur atau memiliki struktur yang sama meskipun mereka memiliki klausa mana yang berbeda. Pertanyaan-pertanyaan ini mungkin adalah yang menyebabkan kesalahan kehabisan memori Heroku PostgreSQL. Periksa rencana kueri untuk kueri tersebut dengan menjalankan EXPLAIN pada dugaan kueri. Jika Anda merasa bahwa permintaannya sangat mahal, itu bisa menjadi tabel penuh memindai atau bergabung dengan beberapa tabel, Anda dapat mencoba menyetelnya. Untuk mengatasi kesalahan yang ada dan selanjutnya mengonfirmasi bahwa kueri memang penyebabnya,

bunuh kueri menggunakan perintah tambahan Heroku Postgres sebagai berikut:

$ heroku pg: bunuh <proses id>

Ketika semuanya gagal

Bukan tidak biasa untuk menghadapi situasi ketika rilis baru aplikasi Anda tidak berfungsi sebagaimana dimaksud dan Anda ingin kembali ke versi sebelumnya sebagai langkah sementara. Untuk melakukannya pada fl adalah pertanyaan lain. Di sinilah letak keajaiban perintah rollback heroku. Ini memungkinkan Anda kembali ke versi aplikasi Anda sebelumnya.

Untuk memeriksa riwayat rilis aplikasi Anda, Anda dapat mengetik yang berikut:

$ heroku rilis

=== lembut-mesa-5445 Rilis

v2 Aktifkan Logplex ahanjura@gmail.com 2012/12/03 03:39:37

v1 Rilis awal ahanjura@gmail.com 2012/12/03 03:39:36

Anda juga dapat mengeluarkan perintah berikut:

$ heroku rollback

Bergulir kembali lembut-mesa-5445 ... selesai, v2

! Peringatan: rollback memengaruhi kode dan konfigurasi; itu tidak menambah atau hapus addons. Untuk membatalkan, jalankan: heroku rollback v3

Atau, Anda dapat menentukan dan mengeluarkan hal-hal berikut:

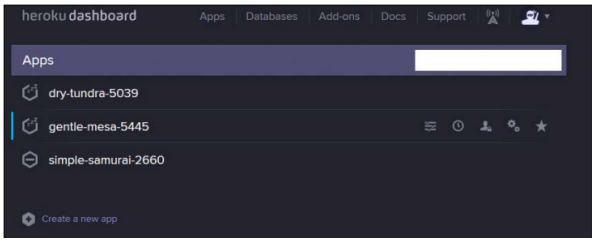
$ heroku rollback <nama versi>

Pemeriksaan produksi

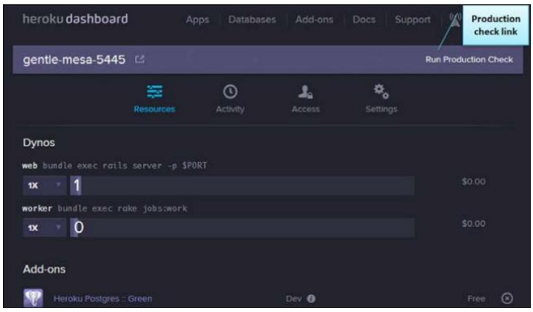
Heroku baru-baru ini memperkenalkan konsep pemeriksaan produksi – pemeriksaan kesehatan langsung untuk aplikasi Anda. Anda dapat menggunakannya dengan masuk ke dasbor Heroku.

Lakukan langkah-langkah berikut untuk melakukan pemeriksaan produksi:

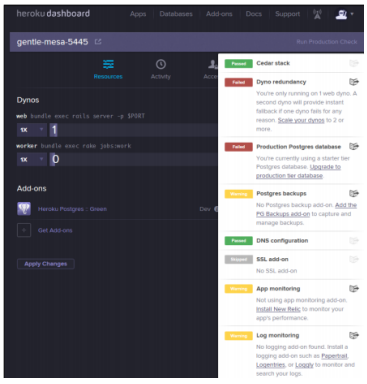
1. Masuk ke akun Heroku Anda dengan mengunjungi https://www.heroku.com.
2. Klik pada aplikasi spesifik yang ingin Anda jalankan pemeriksaan produksi:



3. Klik tautan Jalankan Pemeriksaan Produksi di sudut kanan atas halaman dasbor aplikasi:



4. Tinjau status produksi saat ini dari berbagai bagian aplikasi Anda. Mengklik bagian tertentu akan memberikan rincian tentang kesalahan dalam komponen itu. Pertimbangkan tangkapan layar berikut:



Pemeriksaan produksi akan menjalankan serangkaian tes pada aplikasi dan menunjukkan status proses untuk masing-masing komponen aplikasi Anda. Setelah menunjukkan komponen yang gagal, Anda dapat mengisolasi komponen yang relevan dan menggali lebih dalam ke akar penyebab dengan melihat lebih jauh pada log dan berpotensi menggunakan teknik lain yang dijelaskan dalam bagian ini.

Fitur cek produksi yang baru diperkenalkan menguji aplikasi target terhadap serangkaian kriteria yang sangat baik dan sangat direkomendasikan. Kriteria tolok ukur ini dipilih oleh Heroku untuk memastikan ketersediaan tinggi untuk aplikasi. Ini juga memvalidasi bahwa Anda memiliki alat perangkat lunak yang diperlukan untuk memantau parameter

diperlukan untuk ketersediaan sistem. Anda disarankan untuk menggunakan fitur pemeriksaan produksi dari waktu ke waktu, misalnya, ketika basis pengguna aplikasi Anda diperkirakan akan naik secara drastis atau Anda telah meluncurkan fitur produk baru yang berpotensi dapat mengganggu infrastruktur Anda yang ada saat aplikasi mulai menangani volume permintaan yang jauh lebih besar di lingkungan produksi.

Sekarang kita telah melihat teknik yang tersedia untuk memecahkan masalah aplikasi Anda pada platform Heroku, mari kita mencoba untuk memahami cara menggunakan konfigurasi Heroku yang direkomendasikan untuk menjalankan aplikasi Anda dengan cara yang paling berkinerja, kuat, dan dapat diskalakan.

Konfigurasi Heroku yang disarankan

Tumpukan

Tumpukan Heroku Cedar adalah tumpukan platform Heroku yang saat ini dan paling banyak digunakan. Ini didasarkan pada Ubuntu 11 sebagai sistem operasi dasar dan menghadirkan platform yang toleran terhadap kesalahan yang dapat diskalakan dan tersedia untuk menjalankan aplikasi Heroku Anda.

Jika aplikasi Anda berjalan pada tumpukan Heroku lama atau Anda bermigrasi ke Heroku, pastikan Anda menggunakan tumpukan Cedar untuk aplikasi Anda. Tautan di https://

devcenter.heroku.com/articles/cedar-migration menyediakan informasi terperinci tentang cara memigrasi aplikasi Anda ke tumpukan Cedar untuk mendapatkan kinerja dan fitur terbaik. Jika Anda menggunakan tumpukan yang lebih lama, masuk akal untuk bermigrasi ke tumpukan Cedar dan memanfaatkan kekuatan penuh platform Heroku.

Jika Anda sudah menggunakan Heroku, Anda dapat memeriksa tumpukan saat ini yang digunakan oleh aplikasi Anda dengan masuk ke Heroku dan pergi ke folder aplikasi yang ditentukan dan mengetik perintah heroku stack berikut. Tanda * menunjukkan tumpukan yang saat ini digunakan.

$ heroku stack -sampel contoh
=== sampleapp Stack yang Tersedia
bamboo-mri-1.9.2
bamboo-ree-1.8.7
*** cedar**

Proses pembentukan

Anda dapat membangun ketersediaan ke dalam aplikasi Anda dengan menjalankan beberapa dyno. Misalnya, jika Anda menjalankan setidaknya dua dino untuk aplikasi web Anda, dino ini cenderung dieksekusi pada mesin yang berbeda, sehingga meningkatkan redundansi aplikasi kritis misi Anda. Disarankan agar Anda menggunakan dua atau lebih dinamika web untuk aplikasi Anda untuk memungkinkan ketersediaan layanan yang berkelanjutan untuk pengguna Anda. Dengan menjalankan dino aplikasi pada mesin yang berbeda, Heroku memastikan bahwa setidaknya beberapa dino tersedia untuk melayani permintaan pengguna ketika beberapa dino mengalami masalah dan harus dinyalakan kembali. Anda juga harus mempertimbangkan meningkatkan aplikasi Anda sesuai dengan perlu memastikan bahwa sistem Anda dapat mengikuti peningkatan permintaan selama masa puncak. Memiliki redundansi yang cukup sangat penting untuk efisiensi sistem Anda dalam menangani permintaan.

Terkadang, Anda mungkin ingin memantau dinamika web yang sedang berjalan dan melihat apakah mereka membuat kemajuan dalam melayani permintaan pengguna. Untuk memeriksa status saat ini dari formasi proses Anda, gunakan perintah heroku ps sebagai berikut:

$ heroku ps -a sampleapp
=== web:
web.1: untuk 37m

Seringkali, peningkatan beban permintaan pengguna yang tiba-tiba dapat mendatangkan malapetaka ke aplikasi Anda jika aplikasi tidak menskala secara otomatis karena peningkatan beban. Misalnya, bayangkan menjalankan aplikasi e-commerce berbasis Heroku yang menjual hadiah untuk teman dan keluarga dan coba tebak, Anda melihat peningkatan volume permintaan secara sporadis karena pengguna membeli hadiah untuk Hari Natal. Sebelum pemuatan permintaan mencapai ambang batas aplikasi Anda, Anda harus mempertimbangkan untuk menambahkan proses web tambahan untuk menangani beban kerja yang berpotensi besar ini. Anda mungkin perlu meningkatkan proses latar belakang Anda dan pekerjaan lain juga, tergantung pada formasi saat ini berjalan pada platform.

Melakukannya di Heroku itu sederhana. Ketik heroku ps: perintah skala sebagai berikut. Perintah ini mulai 3 dino lagi untuk menangani permintaan HTTP untuk aplikasi.

$ heroku ps: skala web + 3
Menskalakan proses web ... selesai, sekarang jalankan 4

Layanan basis data

Heroku menawarkan database yang andal dan kuat sebagai layanan di PostgreSQL. Anda dapat menggunakan Heroku PostgreSQL untuk kebutuhan penyimpanan data aplikasi Anda dalam pengembangan serta lingkungan produksi. Untuk aplikasi tingkat produksi yang serius, Anda dapat membeli salah satu dari delapan paket tambahan PostgreSQL yang tersedia di Platform heroku. Heroku juga mendukung beberapa arsitektur penyimpanan data lainnya seperti basis data dalam memori (Memcached),

penyimpanan data non-relasional (CouchDB), dan penyimpanan data berorientasi dokumen (MongoDB) untuk menyediakan penyimpanan data yang sangat tersedia.

Keuntungan tertentu menggunakan Heroku PostgreSQL meliputi:

- Dukungan pada berbagai platform perangkat keras dan OS.
- API langsung untuk terhubung dari berbagai bahasa dan kerangka kerja pemrograman dan kemampuan untuk menghasilkan string konfigurasi untuk bahasa-bahasa ini secara otomatis.
- Rentang rencana tambahan penyimpanan data yang tersedia – hingga hari ini, rencana tingkat produksi entry level yang disebut Crane menyediakan hingga 410 MB RAM, sedangkan paket Mecha kelas atas, cocok untuk penyebaran basis data besar, menyediakan hingga 16 GB RAM . Harga saat ini dari rencana tingkat produksi ini berkisar dari $ 50 per bulan hingga $ 6.400 per bulan.
- Selain itu, seiring dengan meningkatnya kebutuhan aplikasi Anda dan Anda perlu mendukung basis pengguna yang lebih besar, Anda dapat mengatur skala basis data PostgreSQL Anda secara horizontal dengan menambahkan pengikut hanya baca yang tetap mengikuti perkembangan data dengan master database. Selain itu, beberapa paket PostgreSQL tingkat lanjut menyediakan fitur-fitur berikut:
    - Waktu kerja yang diharapkan 99,95 persen Batas koneksi hingga 500 koneksi basis data
    - Baris tidak terbatas Snapshots otomatis harian dengan pemeriksaan retensi satu bulan
    - Pemeriksaan kesehatan otomatis
    - Klip data pengikut basis data Garpu basis data °
    - Langsung akses psql / libpq
    - Layanan basis data yang dikelola sepenuhnya Postgres 9.2, tidak dimodifikasi untuk kompatibilitas yang terjamin
    - Postgres ekstensi (hstore)
    - Tulis-depan log yang didukung setiap 60 detik

Untuk menambahkan add-on PostgreSQL ke aplikasi Anda, Anda bisa mengetikkan yang berikut:

$ heroku addons: tambahkan heroku-postgresql

Untuk memeriksa konfigurasi database Anda saat ini, gunakan perintah heroku pg: info sebagai berikut. Perintah menampilkan rencana Anda telah berlangganan, status ketersediaan database, ukuran database dan versi database di samping perincian lainnya.

$ heroku pg: info -a sampleapp
=== HEROKU_POSTGRESQL_RED
Paket: Ronin
Status: tersedia
Ukuran Data: 7,3 MB
Tabel: 0
Versi PG: 9.1.4
Fork / Ikuti: Tersedia
Dibuat: 2013-05-13 11:23 UTC
Pemeliharaan: tidak diperlukan

Pertimbangan domain dan keamanan

Jika Anda menggunakan nama domain khusus, Anda harus memvalidasi sebelumnya jika URL aplikasi terkonfirmasi dengan benar dan akan menyelesaikan ke alamat yang benar selama pencarian DNS. Seringkali, konfigurasi yang salah akan mendarat Anda di halaman kesalahan "tidak ada situs web ditemukan".

Aspek kunci lain dari resolusi DNS adalah kenyataan bahwa jika Anda menggunakan Secure Sockets Layer (SSL) untuk mengamankan komunikasi browser dengan server, konfigurasi DNS Anda harus mengarahkan pengguna ke titik akhir SSL dengan benar. Pada platform Heroku, hal yang sama akan terlihat seperti endpoint-name.herokussl.com. Anda harus melihat alias memetakan www.sampleapp.com menjadi app-name.herokuapp.com atau endpoint-name.herokussl.com.

Anda harus menggunakan cara standar untuk mengakses berbagai jenis domain. Domain kosong, root, dan telanjang

seharusnya tidak dikonfigurasikan menggunakan catatan-A. Anda harus mengonfigurasi domain Anda secara tepat menggunakan fungsi mirip CNAME atau pengalihan subdomain.

Juga, Heroku secara default memungkinkan pengguna mengakses situs-situsnya seperti https: // status. heroku.com menggunakan https untuk lebih mengamankan komunikasi dengan aplikasi Anda.

Untuk memeriksa apakah catatan DNS terkonfigurasi dengan benar untuk aplikasi Anda, gunakan utilitas baris perintah host berikut:

$ host www.sampleapp.com

www.sampleapp.com adalah alias untuk dora-1234.herokussl.com.

dora-1234.herokussl.com adalah alias untuk xxxx-xxxx.us-east-1.elb.

amazonaws.com.

xxxx-xxxx.us-east-1.elb.amazonaws.com memiliki alamat x.x.x.x â € ¦

Output dari perintah ini memberikan detail tentang lokasi server aktual aplikasi web kami.

Pemantauan kesehatan proaktif

Desainer dan pengguna aplikasi cloud mengharapkan kegagalan di beberapa titik. Oleh karena itu, sebagian besar penyedia layanan cloud merencanakan untuk menghadapi situasi seperti bencana di muka ketika menggunakan layanan mereka. Heroku menyediakan add-on luar biasa yang disebut New Relic (http://www.newrelic.com) yang membantu memantau kesehatan aplikasi Anda secara proaktif. New Relic membantu Anda memonitor aplikasi web / seluler dan server tanpa konfigurasi. Ini membantu dalam mengidentifikasi kemacetan sebelum mereka mulai memengaruhi kinerja aplikasi Anda secara negatif. Anda dapat mengambil langkah-langkah dan memodifikasi aplikasi Anda seperti yang diperlukan untuk menghindari masalah ini.

Heroku merekomendasikan agar Anda menggunakan konfigurasi ini dan alat terkait untuk mengurangi kemungkinan masalah untuk aplikasi web Anda.

Ketika aplikasi web Anda mulai melakukan kesalahan atau perlu upgrade karena fitur baru sedang diperkenalkan, Anda mungkin ingin mengarahkan pengguna aplikasi Anda ke halaman informasi saat Anda menyelesaikan masalah atau menyelesaikan upgrade. Di sinilah halaman kesalahan khusus berguna, terutama selama waktu henti yang direncanakan, juga disebut jendela pemeliharaan.

Jendela perawatan

Selama pemeliharaan aplikasi atau migrasi, Anda dapat menampilkan halaman web khusus untuk pengguna aplikasi Anda. Heroku menyajikan halaman statis setiap kali pengguna mencoba mengakses aplikasi selama jendela pemeliharaan.

Memeriksa status perawatan

Jika Anda perlu memeriksa status saat ini dari jendela pemeliharaan untuk aplikasi Anda, Anda dapat menggunakan perintah pemeliharaan heroku. Perintah ini akan menampilkan arus status jendela pemeliharaan, yaitu, aktif jika pemeliharaan sedang berlangsung dan dimatikan jika aplikasi berjalan dalam mode produksi / pengembangan reguler:

$ heroku maintenance
Mati

Mengaktifkan mode pemeliharaan

Untuk mengaktifkan mode pemeliharaan untuk aplikasi Anda, Anda dapat menggunakan perintah berikut dari antarmuka klien Heroku (CLI):

$ heroku maintenance: on
Mengaktifkan mode perawatan untuk aplikasi sampel ...
selesai

Menonaktifkan mode pemeliharaan

Untuk menonaktifkan mode pemeliharaan untuk aplikasi Anda, Anda dapat menggunakan perintah berikut dari antarmuka klien Heroku (CLI):

$ heroku maintenance: mati

Menonaktifkan mode perawatan untuk aplikasi sampel ... selesai

Jendela pemeliharaan - di belakang layar

Selama jendela pemeliharaan, dinamika web atau pekerja yang sudah berjalan di platform Heroku tidak berhenti. Alih-alih, perute HTTP memblokir permintaan apa pun agar tidak mencapai dinamika ini. Anda dapat memilih untuk mengurangi skala dino yang berjalan di platform Heroku selama jendela pemeliharaan karena Anda tetap tidak akan menerima banyak permintaan. Namun, pastikan untuk meningkatkan kembali setelah jendela perawatan selesai dan aplikasi mulai beroperasi pada tingkat normal.

Menyesuaikan konten situs

Untuk menyesuaikan konten yang akan digunakan untuk tampilan selama jendela pemeliharaan, Anda dapat menggunakan parameter konfigurasi MAINTENANCE_ PAGE_URL.

Misalnya, untuk mengatur URL halaman pemeliharaan, Anda dapat mengetik berikut ini:

$ heroku config: set MAINTENANCE_PAGE_URL = http: //s3.amazonaws.com/johndoe/ maintenance_pagehtml

Dalam contoh sebelumnya, johndoe adalah nama ember yang ditentukan dari pengguna yang memiliki penyimpanan di AWS.

Menyesuaikan halaman kesalahan

Anda juga dapat menyesuaikan konten yang ditampilkan saat aplikasi Anda melakukan kesalahan. Ini dapat dicapai dengan menggunakan parameter konfigurasi ERROR_PAGE_ URL.

Untuk mengatur ERROR_PAGE_URL, ketikkan yang berikut:

$ heroku config: set ERROR_PAGE_URL = http: //s3.amazonaws.com/johndoe/error_ page.html

Anda perlu memastikan sumber daya (html / gambar / css / lainnya) yang perlu Anda gunakan untuk menampilkan halaman dapat dibaca oleh semua orang.

Menguji pemeliharaan kustom dan halaman kesalahan

Untuk menguji apakah halaman pemeliharaan berfungsi seperti yang dirancang, aktifkan mode pemeliharaan aplikasi dan kemudian gunakan perintah buka heroku berikut:

$ heroku maintenance: on

Mengaktifkan mode perawatan untuk aplikasi sampel ... selesai

$ heroku terbuka

Halaman kustom akan dilayani dan log aplikasi akan menunjukkan kode kesalahan Heroku H80 untuk permintaan web yang menunjukkan bahwa halaman pemeliharaan disajikan kepada pengguna.

$ heroku log -p router -n 1

2013-05-08T14: 44: 12-06: 03 heroku [router]: at = info code = H80

desc = "Mode perawatan" metode = DAPATKAN

Untuk menguji apakah konfigurasi halaman kesalahan berfungsi, Anda dapat membuat permintaan untuk aplikasi yang habis waktu tersebut. Aplikasi mengembalikan kesalahan waktu permintaan H12. Anda dapat memeriksa log sambil membuat permintaan ini dan log akan terlihat seperti berikut:

$ heroku log --tail

2013-05-08T18: 04: 40-07: 00 app [web.1]: Tidur 35 detik sebelum saya melayani halaman ini

2013-05-08T18: 05: 10-07: 00 heroku [router]: at = kode kesalahan = H12 desc = "Metode batas waktu permintaan" = jalur GET = / host = sampleapp.herokuapp.com fwd = X.X.X.X

dyno = web.1 connect = 3ms service = 30001ms status = 503 bytes = 0
2013-05-08T18: 05: 15-07: 00 app [web.1]: Selesai tidur

Halaman kesalahan khusus ditampilkan di browser Anda. Hanya kesalahan tingkat sistem seperti tidak ada respons atau respons yang salah hasil dalam tampilan halaman kesalahan Heroku yang dijelaskan sebelumnya. Kesalahan aplikasi akan menampilkan halaman kesalahan aplikasi dan bukan halaman kesalahan Heroku.

Ketika meminta waktu habis

Salah satu alasan permintaan waktu habis adalah antrian permintaan klien untuk sebuah dino, menghasilkan eksekusi permintaan berurutan. Dalam hal ini, permintaan apa pun yang saat ini dalam antrian harus menunggu yang sebelumnya dieksekusi hingga selesai. Waktu yang dibutuhkan untuk mengeksekusi permintaan sebanding dengan posisi permintaan dalam antrian. Semakin jauh permintaan, semakin lama waktu yang dibutuhkan untuk mengeksekusi. Skenario ini adalah tipikal dari bahasa yang mendukung penanganan permintaan utas tunggal.

Masalah seperti itu dapat dikurangi (setidaknya sampai batas tertentu) dengan melakukan hal berikut:

- Berbagi dino antar proses. Anda dapat menggunakan server web alternatif seperti Unicorn yang membantu berbagi antrean dyno antar proses. Ini membantu menghapus backlog permintaan dengan cepat dan membuat throughput respons yang lebih baik.
- Menjalankan lebih banyak proses dyno untuk meningkatkan konkurensi. Ini secara khusus mengurangi kemungkinan antrian permintaan seperti dijelaskan sebelumnya.
- Tuning aplikasi Anda, terutama bidang yang paling banyak menyita waktu atau perlu sinkronisasi.

Klasifikasi kesalahan dalam Heroku

Pemecahan masalah aplikasi terdistribusi dapat menjadi perhatian utama bagi sebagian besar pengembang. Log yang didistribusikan, beberapa pekerjaan, dan lebih dari satu antarmuka pengguna dapat menyebabkan malam sulit bagi seseorang yang mencoba memecahkan masalah aplikasi. Jadi, menjadi sangat penting bahwa lingkungan eksekusi aplikasi Anda memberikan petunjuk yang cukup tentang potensi penyebab kesalahan atau situasi fatal.

Heroku kesalahan aplikasi biasanya muncul sebagai halaman kesalahan untuk pengguna ketika permintaan web dibuat. Heroku akan mengembalikan halaman kesalahan standar dengan kode status HTTP 503. Selain itu, Heroku menambahkan informasi kesalahan khusus ke log aplikasi.

Platform Heroku menggunakan konvensi berikut untuk mengklasifikasikan berbagai jenis kesalahan ini:

Kode kesalahan awalan	Jenis kesalahan
H	HTTP error
R	Runtime error
L	Kesalahan logging

Misalnya, berikut ini adalah kutipan dari proses web macet. Kode kesalahan ditunjukkan dalam warna yang berbeda.

2013-03-04T11: 25: 10-07: 00 heroku [web.1]: Proses keluar

2013-03-04T11: 25: 12-07: 00 heroku [router]: at = kode kesalahan = H10 desc = "App crashed" method = GET

Kesalahan HTTP yang didukung saat ini ditunjukkan pada tabel berikut:

Kode kesalahan	Resolusi pesan	kesalahan
H10	Aplikasi mengalami	gangguan, lalu restart.
H11	Backlog Run terlalu dalam	lebih banyak dyno, tune database, atau buat kode lebih cepat

H12	Permintaan batas waktu	Atur timer aplikasi atau tekan lama durasi pekerjaan ke latar belakang.
H13	Koneksi ditutup tanpa tanggapan	Hindari batas waktu.
H14	Tidak ada proses web yang berjalan	Dinamika booting atau skala menggunakan proses web the heroku ps: perintah skala.
H15	Koneksi idle .	Biasanya karena koneksi tidak aktif, no resolusi khusus
H16	Redirect ke herokuapp.com	Ini bukan benar-benar kesalahan tetapi sebuah pesan informasi yang menunjukkan itu permintaan dibuat ke tumpukan Cedar aplikasi menggunakan heroku lama. domain com akan dialihkan ke herokuapp.com.
H17	HTTP diformat dengan buruk tanggapan	Tidak ada resolusi khusus. Kesalahannya menunjukkan HTTP yang tidak valid atau salah tanggapan dari dino yang menangani permintaan.
H18	Permintaan terputus	Disebabkan karena penutupan klien atau soket server. Tidak khusus resolusi kecuali mencoba kembali permintaan.
H19	Batas waktu koneksi backend	Biasanya disebabkan karena lebih banyak permintaan dibuat dari yang bisa ditangani oleh dino.Biasanya diselesaikan dengan menjalankan lebih banyak dino.
H20	Batas waktu boot aplikasi	Terjadi jika dinamika web tidak dapat menjangkauan Status "boot" atau "up" dalam 75 detik permintaan enqueuing oleh router. Tidak resolusi khusus.
H21	Sambungan backend ditola	Biasanya gejala dari aplikasi sedang kewalahan dan tidak dapat menerima yang baru koneksi. Jalankan lebih banyak dino
H22	Batas koneksi tercapai	karena lonjakan klien HTTP koneksi ke aplikasi yang dihasilkan

		dalam kelebihan. Tidak ada resolusi khusus.
H80	Mode pemeliharaan	Ini bukan kesalahan tetapi indikasi bahwa aplikasi sedang berjalan di mode pemeliharaan.
H99	Kesalahan platform .	Dicatat karena Heroku internal kesalahan platform. Ini bisa menjadi intermiten kesalahan atau masalah serius. Khas diselesaikan dengan mencoba lagi nanti, memeriksa situs status Heroku untuk Heroku-lebar masalah, atau menaikkan tiket dengan Heroku

Kesalahan runtime yang didukung saat ini ditunjukkan pada tabel berikut:

Kode kesalahan	Pesan eror	Resolusi
R10	Batas waktu booting	Disebabkan ketika proses web gagal mengikat ke yang ditunjuk Pelabuhan. Biasanya diselesaikan dengan mengurangi ketergantungan pada sistem eksternal atau kode rekayasa ulang untuk mengurangi nya ukuran atau ketergantungan pada perhitungan pengambilan waktu seperti pertanyaan eksternal.
R11	Ikatan buruk	Sekarang sudah usang. Biasanya terjadi saat proses mencoba mengikat ke port yang tidak ditunjuk.
R12	Keluar dari batas waktu	Terjadi ketika suatu proses tidak berhenti setelah SIGTERM dikirim untuk itu. Dapat diatasi dengan mematikan proses menggunakan secara paksa SIGKILL.
R13	Lampirkan kesalahan	Dihasilkan ketika proses dimulai dengan heroku run gagal melampirkan ke klien yang memohon. Dapat diatasi dengan coba lagi.
R14	Kuota memori terlampaui	Terjadi ketika sebuah dyno melewati batas 512 yang telah dirancang

		sebelumnya Penggunaan memori MB. Aplikasi dapat terus berlanjut berjalan, tetapi kinerjanya akan menurun seiring waktu. Potensi Resolusi adalah dengan menggunakan dinamika ukuran memori 2X, buat lebih banyak dynos, dan bagikan.
R15	Kuota memori jauh melebihi	Hasil ketika sebuah dyno menggunakan jumlah yang terlalu besar RAM terhadap yang dialokasikan 512 MB. Biasanya berakhir dengan pembunuhan proses yang salah. Resolusi umum adalah untuk me-reboot banyak dynos untuk membagikan beban permintaan.
R16	Terpisah	Terjadi ketika proses terlampir terus berjalan merata setelah dikirim SIGHUP ketika koneksi eksternal Tutup. Jika kesalahan ini terjadi dan Anda ingin prosesnya keluar, Anda bisa membersihkannya secara paksa.

Kesalahan logging yang didukung saat ini ditunjukkan pada tabel berikut:

Kode kesalahan	Pesan eror	Resolusi
L10	Tiriskan buffer meluap	Lonjakan pesan log yang dihasilkan oleh dyno, menghasilkan sistem Logplex menjatuhkan beberapa dari mereka untuk mengikuti. Resolusi dapat berupa instrumentasi kode untuk mengurangi emisi jumlah pesan log atau memperbaiki loop buggy, misalnya.
L11	Penyangga ekor meluap	Mirip dengan kesalahan sebelumnya, perbedaannya adalah bahwa sesi ekor yang dijalankan

		oleh Anda tidak dapat mengikuti volume log yang dihasilkan oleh aplikasi. Anda mungkin perlu untuk mengurangi volume logging yang dipancarkan oleh aplikasi atau dapatkan bandwidth yang lebih baik untuk menerima lebih banyak pesan log di delta waktu yang sama.

Anda dapat merujuk ke deskripsi terperinci untuk masing-masing kode kesalahan di

https://devcenter.heroku.com/articles/error-codes

Ringkasan

Dalam bab ini, kami meninjau berbagai cara pemecahan masalah aplikasi Heroku. Kami melihat cara logging bekerja di Heroku dan bagaimana memanfaatkan utilitas Heroku CLI untuk meninjau log aplikasi dan memecahkan masalah masalah yang ditentukan. Kami juga meninjau teknik yang tersedia untuk memecahkan masalah downtime aplikasi, men-debug permintaan HTTP, dan memproses pembentukan aplikasi Anda. Kami juga menemukan bagaimana Anda bisa gulung balik aplikasi Anda ke kondisi kerja sebelumnya ketika setiap teknik pemecahan masalah lainnya gagal. Kami menemukan konfigurasi aplikasi yang direkomendasikan yang membantu aplikasi menghindari kesalahan dan waktu henti. Kami belajar cara menangani jendela perawatan dan menghindari batas waktu. Akhirnya, kami menggali jauh ke dalam sistem kode kesalahan Heroku – the konvensi penamaan kesalahan dan katalog kode kesalahan Heroku.

Pada bab selanjutnya, kita akan mempelajari fitur-fitur canggih dari platform Heroku dan bagaimana memanfaatkan fitur-fitur tersebut untuk membangun dan menggunakan aplikasi web yang kaya fitur, performan, dapat diskalakan, dan siap produksi.

BAB 9
PENGGUNAAN HEROKU TINGKAT LANJUT

Kami berada di akhir perjalanan kami bersama Heroku. Dalam buku ini, kami telah belajar bahwa Heroku menyenangkan dan mudah digunakan. Melalui perjalanan ini, kami telah belajar bagaimana membangun, menyebarkan, dan memecahkan masalah aplikasi Heroku menggunakan berbagai macam add-on dan alat yang tersedia. Fokus kami adalah pada platform Heroku, yaitu, apa yang ditawarkannya kepada pengembang dalam hal layanan, dan bagaimana pengembang dapat memanfaatkan layanan dan meluncurkan aplikasi yang dibangun di atas tumpukan Heroku. Selama perjalanan kami, kami juga belajar tentang bagaimana platform Heroku diamankan bagi pengembang untuk membangun aplikasi yang andal bagi pengguna akhir. Dengan menunjukkan berbagai teknik, kami juga belajar cara menggunakan platform Heroku lebih efektif, baik itu konfigurasi DNS untuk aplikasi Anda, penggunaan konfigurasi 2X dyno, atau mengatur dan mengelola penyimpanan data seperti sebagai Heroku PostgreSQL.

Dalam bab ini, kita akan melihat beberapa aspek yang berguna dan canggih dari platform Heroku:

- Kami meninjau beberapa fitur eksperimental yang sangat berguna yang disediakan oleh Heroku Labs, khusus untuk membangun, menyebarkan, dan memantau aplikasi Anda.

- Kami akan membaca primer tentang teknologi Websockets dan penggunaannya di Heroku yang membantu pengembang membangun aplikasi web real-time berkinerja tinggi
- Kami akan belajar bagaimana memanfaatkan API Platform Heroku untuk secara terprogram melakukan operasi terkait aplikasi apa pun di lingkungan Heroku, sebuah fitur yang membantu Anda membangun dan menjual layanan yang dibangun di atas lingkungan cloud Heroku
- Terakhir, kita akan mempelajari aspek yang sering diabaikan namun sangat penting dari pengembangan aplikasi berbasis cloud, yaitu, pengembangan kolaboratif, dan memahami cara berbagi aplikasi Anda dengan orang lain untuk produktivitas yang lebih tinggi Jadi, mari kita mulai.

Bereksperimen dengan Heroku Labs

Meskipun kami telah fokus pada berbagai fitur utama platform Heroku di bab-bab sebelumnya, kami masih memiliki aspek yang sangat kuat dari platform Heroku untuk diungkap: Lab Heroku. Seperti namanya, Heroku Labs adalah lingkungan eksperimental - lingkungan komputasi canggih untuk meneliti dan bereksperimen dengan fitur potensial yang dipertimbangkan untuk rangkaian fitur Heroku.

Heroku Labs menyediakan tempat terbuka bagi Anda untuk bereksperimen dengan kemampuan baru dan juga menguji alternatif yang dibuat untuk mengatasi berbagai keterbatasan yang diketahui pada platform Heroku. Sebagai contoh, Heroku baru-baru ini mulai mendukung Websockets di Heroku (saat ini fitur ini dalam versi beta publik) untuk membangun aplikasi web real-time yang tampil di platform Heroku. Ini adalah kemampuan yang sama sekali baru untuk platform Heroku.

Sebelumnya dalam buku ini, kami belajar dan bereksperimen dengan set fitur platform Heroku yang saat ini tersedia. Di bagian ini, kita akan melangkah lebih jauh dan melihat penawaran Heroku Labs. Anda dapat mencoba fitur-fitur ini dengan aplikasi Anda dan membuat aplikasi web real-time yang lebih cepat, portabel, dan berperilaku baik.

Menggunakan fitur Heroku Labs

Heroku Labs adalah lingkungan eksperimental canggih yang menyediakan banyak fitur berguna bagi para pengembang untuk bereksperimen. Di bagian ini, kami menjelajahi beberapa fitur terkemuka yang tersedia di lingkungan Heroku Labs yang berguna saat mengembangkan atau memelihara aplikasi web pada platform Heroku.

Pemasangan mulus menggunakan jalur pipa

Sebagian besar tim operasi telah menguasai seni menciptakan metodologi promosi kode langkah-demi-langkah untuk memfasilitasi tingkat pengujian yang sesuai sebelum perangkat lunak diekspos kepada pengguna akhir dalam produksi. Proses yang biasa dilakukan adalah kode pengembang dipromosikan ke testbed kualitas di mana ia dibangun dan diuji dan dari sana produk dipromosikan menjadi penerimaan pengguna dan penerimaan pelanggan sebelum menjangkau pengguna akhir. Kebanyakan alur kerja pengiriman merupakan variasi kecil dari proses proses ini. Namun, sebagian besar tim masih merasa cukup sulit untuk membuat alur kerja penyebaran yang konsisten antara lingkungan ini.

Lebih sering daripada tidak, pemilik aplikasi harus mengelola secara manual semua alur kerja yang penting dan memastikan bahwa hanya komponen atau modul yang diinginkan yang dipromosikan di sepanjang jalur pengiriman. Segera setelah kami membuat ketergantungan pada intervensi manual untuk proses penyebaran ini, kami membuat proses promosi kode rentan terhadap kesalahan manusia seperti mendorong kode yang salah atau mempromosikan kode ke test bed yang salah. Ini bukan hanya overhead untuk tim operasi,tetapi juga merupakan hambatan untuk tujuan menyediakan kemampuan penyebaran perangkat lunak aplikasi web Anda secara terus menerus dan mulus.

Heroku Labs menawarkan fitur eksperimental utama untuk menghindari sebagian besar masalah ini. Fitur yang disebut saluran pipa ini membantu Anda mengesampingkan aturan workflw penyebaran yaitu, dengan fitur ini, Anda dapat menolak panduan

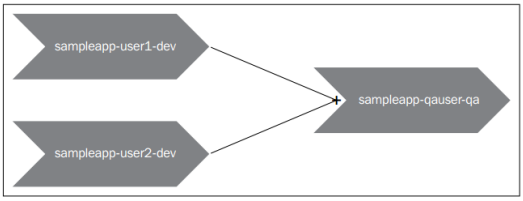
penerapan aplikasi Anda atau aturan. Fitur pipa Heroku melakukan instruksi penerapan aplikasi Anda secara otomatis, sehingga mengurangi kemungkinan kesalahan manusia paling umum. Saluran pipa membantu Anda menentukan aturan untuk memindahkan kode Anda dari satu lingkungan ke lingkungan lain.

Beberapa skenario penggunaan umum yang sesuai untuk penggunaan pipa Heroku adalah sebagai berikut:

1. Tim Operasi (OPS) mempromosikan kode dari lingkungan pengujian tunggal kepada yang lain.

   Dalam hal ini, tim OPS mempromosikan kode sampleapp-uat (uji penerimaan pengguna) ke lingkungan sampleapp-custtest (tempat uji pelanggan) melalui pipa. Dalam contoh ini, lingkungan custtest dianggap sebagai ujung hilir dari lingkungan uat.

2. Perubahan kode dari beberapa pengembang digabungkan dan dipromosikan ke lingkungan jaminan kualitas (QA).



   Di sini, tim ops mempromosikan kode dari dua lingkungan pengembangan ke lingkungan QA. Dalam contoh ini, lingkungan qauser-qa dianggap sebagai hilir dari lingkungan user1-dev dan user2-dev.

Mengaktifkan fitur saluran pipa

Menggunakan fitur saluran pipa dari Heroku Labs sangat mudah:

1. Aktifkan fitur Pipeline menggunakan lab: aktifkan perintah sebagai berikut:
   $ heroku labs: aktifkan jalur pipa
   Mengaktifkan saluran pipa untuk name@company.com ... selesai

PERINGATAN: Fitur ini eksperimental dan dapat berubah atau menjadi dihapus tanpa pemberitahuan ....
Untuk informasi lebih lanjut, lihat https: //devcenter.heroku.com / articles / using-pipelines-to-deploy-betweenaplikasi.

2. Instal plugin Heroku CLI untuk jalur pipa seperti yang ditunjukkan pada perintah berikut:
   $ heroku plugins: install git: //github.com/heroku/herokupipeline.git
3. Gunakan pipa heroku: tambahkan perintah untuk mengaitkan hilir aplikasi dengan aplikasi Anda saat ini:
   $ heroku pipeline: tambahkan sampleapp-custtest
   Menambahkan aplikasi hilir: sampleapp-custtest
   Dalam contoh ini, sampleapp adalah aplikasi hilir yang terkait dengan aplikasi Anda saat ini.
4. Untuk memeriksa aplikasi hilir untuk aplikasi Anda saat ini, gunakan perintah pipa heroku:
   $ heroku pipeline
   Pipa: sampleapp-uat ---> sampleapp-custtest
   Dalam contoh ini, sampleapp-uat memiliki aplikasi hilir sampleapp-custtest yang terkait dengannya.
5. Setelah pipa ditolak, Anda dapat menggunakan pipa: promosikan perintah untuk menjalankan instruksi promosi kode dan salin slug aplikasi Anda ke aplikasi hilir yang ditolak sebagai versi baru atau lepaskan menggunakan perintah berikut:
   $ heroku pipeline: promosikan
   Mempromosikan sampleapp-uat ke sampleapp-custtest ... selesai, v3
6. Anda dapat menemukan perbedaan antara hulu dan hilir aplikasi dengan menggunakan pipa heroku: perintah diff sebagai berikut:
   $ heroku pipeline: diff
   Membandingkan sampleapp-uat dengan sampleapp-custtest ... selesai, sampleapp-uat maju dengan 1 komit:
   31cd650 2013-11-27 Kondisi overflow tetap (pengguna1)
   Menggunakan fitur Saluran Pipa tidak menyiratkan promosi otomatis repo Git Anda, variabel konfigurasi, add-on yang

diinstal, dan dependensi terkait lingkungan. Aspek-aspek ini dari aplikasi web Anda perlu dikelola secara terpisah dari promosi kode Anda.

Pemantauan kinerja

Heroku luar biasa. Tidak hanya membuat Anda membangun, menyebarkan, dan memecahkan masalah aplikasi Anda dengan mudah, ia juga memberi Anda alat yang dapat Anda aktifkan (misalnya, fitur Heroku Labs) atau konfigurasi (add-on) dan kemudian gunakan kemampuan baru secara instan . Salah satu kemampuan utama yang Anda perlukan untuk bekerja dengan aplikasi web Anda adalah kemampuan untuk memantau kinerja dinamika web Anda. Sebagai pengembang, Anda perlu tahu seberapa baik kinerja web Anda. Apa kemacetan - peningkatan beban CPU, penggunaan memori yang sangat tinggi, atau yang lainnya.

Heroku Labs menyediakan fitur eksperimental yang disebut metrik runtime log yang melakukan hal itu; itu memungkinkan Anda mengukur CPU dan memori (nyata dan swap) untuk dinamika aplikasi web Anda.

Setelah Anda mengaktifkan fitur ini, data tentang memori dan penggunaan swap di samping beban CPU dari dyno dipancarkan ke dalam aliran log aplikasi. Metrik dicatat secara optimal setiap 20 detik.

Anda dapat menggunakan perintah log Heroku untuk meninjau statistik ini atau mengumpankan log ke add-on yang dapat menggunakan log.

Mengaktifkan pemantauan

Anda dapat mulai menggunakan fitur metrik runtime log dengan menggunakan heroku
labs: aktifkan perintah sebagai berikut:

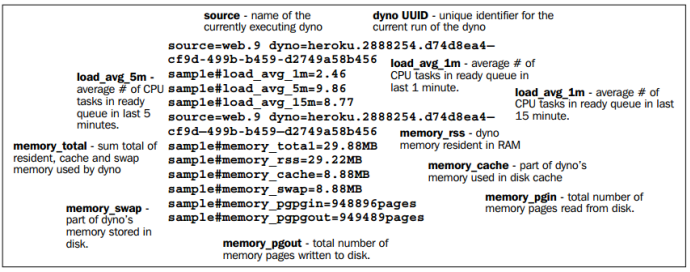
```
$ heroku labs: aktifkan log-runtime-metrics
Mengaktifkan metrik-runtime-metrik untuk sampelapp ... selesai
```

Setelah Anda mengaktifkan fitur ini, jalankan perintah restart heroku seperti yang ditunjukkan pada baris perintah berikut untuk mengaktifkannya untuk aplikasi Anda:

$ heroku restart

Log snapshot

Diagram berikut menunjukkan format metrik log yang dihasilkan yang menjelaskan berbagai bagian pesan:



Dyno tidak mempertimbangkan tugas yang diblokir pada operasi I / O saat menghitung rata-rata beban. Artikel http://www.linuxjournal.com/ artikel / 9001 memberikan tinjauan umum tentang konsep rata-rata beban.

Menonton aplikasi Anda dengan cermat menggunakan ID Permintaan

Mengingat ukuran dan kompleksitas aplikasi web modern, pencatatan memainkan bagian penting dalam masalah pemecahan masalah yang dihadapi selama penanganan Anda permintaan klien. Bahkan kemudian, Anda mungkin merasa sulit untuk menemukan utas spesifik yang didapat diblokir atau dihentikan karena kesalahan di suatu tempat di sistem. Penyelesaian masalah masalah seperti itu seringkali dapat mengganggu. Untungnya, Heroku Labs benar-benar menyediakan fitur berguna yang memungkinkan Anda untuk menghubungkan peristiwa log di tingkat router dengan log acara di tingkat web dyno. Fitur ini, yang disebut **ID Permintaan**, adalah percobaan fitur yang membantu Anda melacak permintaan web Anda ke log dyno web dan mengerti akar penyebab masalah runtime.

Mendukung ID Permintaan

Heroku mendukung konsep ID Permintaan dengan menyediakan fungsionalitas tambahan di tingkat router Heroku. Setiap kali permintaan web datang ke router Heroku, itu router menghasilkan pengidentifikasi permintaan unik – request_id – dan mengaitkannya dengan permintaan yang masuk. Request_id ini dicatat dalam log router, dan router menyimpannya di header HTTP sebelum mengirim permintaan HTTP ke web dyno. Itu fild request_id di header HTTP dapat diambil oleh aplikasi dan dicatat untuk melacak permintaan web tertentu.

Anda dapat mengaktifkan fitur ini dengan menggunakan lab: aktifkan perintah sebagai berikut:

$ heroku labs: aktifkan http-request-id
Mengaktifkan http-request-id untuk sampleapp ... selesai

Setelah fitur ini diaktifkan, log router Anda akan membuat ID permintaan unik dan lampirkan identifir dengan permintaan web sebelum meneruskannya:

2013-12-10T07: 53: 10 + 00: 00 heroku [router]: at = info method = GET path = / host = sampleapp.herokuapp.com request_id = 6d1410f3cfc85a8af9bac12bfd0930e3 fwd = "209.87.1.23" dyno = web.3 terhubung = 0ms layanan = 47ms status = 200 byte = 1132

Dengan akses ke ID permintaan sedikit, pemecahan masalah permintaan web menjadi banyak lebih mudah. Yang harus Anda lakukan adalah sebagai berikut:

1. Cari log router dan cari kesalahan spesifik (H13) dan ID permintaan (6d1410f3cfc85a8af9bac12bfd0930e3), misalnya:
   2013-12-16T11: 25: 12-03: 00 heroku [router]: at = kode kesalahan = H13 desc = "Koneksi ditutup tanpa respons" metode = GET path = / host = sampleapp.herokuapp.com request_id = 6d1410f3cfc85a8af9bac12bfd0930e3 fwd = "209.87.1.23"

dyno = web.1 connect = 1234ms layanan = 4567ms status = 503 byte = 0

2. Cari log web dyno dan cari ID permintaan khusus Anda diambil pada langkah pertama:
   2013-12-16T11: 25: 12-03: 00 heroku [web.1]: request_id = 6d1410f3cfc85a8af9bac12bfd0930e3
   2013-12-16T11: 25: 12-03: 00 heroku [web.1]: /lib/fft/calc.rb:27:
   [BUG] Kesalahan segmentasi

Seperti yang Anda lihat, ada kesalahan segmentasi pada baris 27 di calc.rb Fie. Anda sekarang dapat dengan mudah merujuk ke kode sumber spesifik dan menyelidikinya kemungkinan alasan untuk kesalahan tersebut.

Memperkenalkan Websockets

Protokol **Websocket (WS)** adalah protokol baru yang mendukung saluran dua arah untuk komunikasi antara server dan klien. Protokol ini mengatasi beberapa kelemahan serius dari metode komunikasi HTTP sebelumnya. Ini termasuk duplex penuh, saluran komunikasi dua arah antara server dan klien berakhir TCP. RFC 6455 (http://tools.ietf.org/html/rfc6455) menyediakan spesifikasi terperinci dari protokol ini. Protokol Websocket telah distandarisasi sekitar dua tahun lalu oleh **Internet Engineering Task Force (IETF)** dan sejak itu telah menemukan beberapa kegunaan dalam aplikasi web real-time (game, obrolan waktu-nyata, dan banyak lagi).

Websocket versus HTTP

Protokol Websocket juga memiliki beberapa keunggulan berbeda dibandingkan HTTP, terutama dalam hal pemberitahuan dua arah antara server dan klien. Setelah koneksi soket Web dibuat antara klien dan host server, data dapat mengalir di kedua arah secara bersamaan melalui TCP. Ini mengelak teknik komunikasi yang

digunakan menggunakan HTTP untuk mensimulasikan notifikasi peristiwa waktu nyata, baik itu polling, polling panjang, atau komet. Teknik-teknik lama ini membutuhkan biaya besar. Ini karena untuk permintaan HTTP yang lebih kecil, overhead penggunaan HTTP tampaknya menyebabkan masalah kinerja yang tidak perlu dibandingkan dengan menggunakan Soket web, di mana overhead mungkin hanya beberapa byte per bingkai. Protokol Websocket adalah kandidat yang ideal untuk aplikasi web waktu-nyata latensi rendah.

Websocket bukan HTTP

Protokol Websocket adalah protokol ringan berlapis di atas TCP secara langsung. Selain fakta bahwa HTTP membantu mengatur koneksi soket Web dengan permintaan peningkatan HTTP (peningkatan generik, tidak khusus untuk soket Web), HTTP dan soket Web adalah dua protokol yang sama sekali berbeda.

Kasus penggunaan Websocket

Orang dapat secara luas mengklasifikasikan sebagian besar kasus penggunaan untuk Websockets ke dalam
kategori berikut:

- Pemberitahuan push real-time
- Pengalaman pengguna interaktif real-time
- Acara yang dikirim oleh server

Aplikasi khas menggunakan Websockets

Beberapa aplikasi umum yang menggunakan Websockets adalah sebagai berikut:

- Game multi pemain
- Pengeditan kolaboratif dokumen / kode sumber
- Umpan sosial
- Obrolan multimedia online
- Data tunangan real-time (misalnya, ticker saham)
- Aplikasi berbasis lokasi

Soket Web yang mendukung di aplikasi Anda

Jika Anda menemukan aplikasi web tempat Anda bisa menggunakan Websockets, pertanyaan pertama yang harus ditanyakan adalah: bisakah saya menggunakan Websockets dengan desktop atau browser perangkat seluler saya? Situs web berikut menyediakan matriks kompatibilitas untuk berbagai browser yang mendukung / tidak mendukung protokol Websocket pada saat ini:

http://caniuse.com/Websockets

Membuat koneksi Websocket

Untuk membuat koneksi Websocket, klien dapat melakukan permintaan jabat tangan klien. Permintaan peningkatan HTTP digunakan untuk memulai koneksi Websocket seperti yang ditunjukkan pada contoh berikut:

DAPATKAN / mengobrol HTTP / 1.1
PEMBAWA ACARA: server.sampleapp.com
Tingkatkan: websocket
Koneksi: Tingkatkan
Sec-WebSocket-Key: gRfTgYJhYRBGfDRHJuYKW ==
Asal: http://sampleapp.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13

Jika server menerima permintaan (jika tidak, HTTP 500 dikirim kembali), jabat tangan server akan dikembalikan sebagai berikut:

HTTP / 1.1 101 Protokol Perpindahan
Tingkatkan: websocket
Koneksi: Tingkatkan
Sec-WebSocket-Accept: j8pUytRNy8UyUjILoHrTHjLo + xOo =

Pada jabat tangan yang sukses antara klien dan server, koneksi TCP digunakan untuk membuat permintaan HTTP awal ditingkatkan ke koneksi Websocket. Keduanya server dan klien sekarang dapat mengirim pesan secara bersamaan.

Kerugian menggunakan Websockets

Meskipun Websockets benar-benar cocok untuk pemberitahuan acara dua arah secara real-time antara klien dan tuan rumah, tetapi juga memiliki kelemahannya sendiri. Karena Websockets memerlukan koneksi terus-menerus antara klien dan server dan server perlu mengelola semacam keadaan, protokol tersebut menghabiskan banyak sumber daya.

Namun, untuk jenis aplikasi yang baru saja kita lihat, keuntungannya jauh melebihi rasa sakitnya. Juga, HTTP yang lebih lama dan API REST yang lebih baru memang memiliki kredibilitas mereka sendiri dalam kasus penggunaan tertentu untuk komunikasi web; mereka harus dipertimbangkan saat memutuskan mode komunikasi yang diinginkan antara server Anda dan klien.

Heroku dan Websockets

Heroku memberikan dukungan Websocket ke domain herokuapp.com, domain khusus, dan titik akhir SSL khusus melalui fitur Heroku Labs Websockets eksperimental.

Satu peringatan utama adalah bahwa untuk menyediakan titik akhir SSL khusus dengan dukungan Websocket, fitur Websockets harus diaktifkan sebelum titik akhir disediakan. Jika Anda mengaktifkan fitur Websockets untuk aplikasi dengan titik akhir SSL khusus yang sudah ada, Heroku melempar kesalahan tentang titik akhir SSL sedang digunakan seperti yang ditunjukkan di baris perintah berikut:

```
$ heroku labs: aktifkan Websockets
Mengaktifkan Websockets untuk contoh aplikasi ... gagal
! Tidak dapat menambahkan fitur Websockets ketika ssl-
endpoint sedang digunakan.
```

Cara yang tepat untuk bekerja dengan titik akhir SSL yang ada adalah dengan menghapusnya menggunakan heroku addons: remove ssl: endpoint command, tambahkan dukungan Websockets untuk aplikasi dan kemudian sediakan kembali titik akhir SSL. Tak perlu dikatakan bahwa Anda perlu memperbarui catatan DNS untuk menunjuk ke domain kustom terkait nama host baru didapat selama proses penyediaan titik akhir.

Mengaktifkan dukungan Websocket

Heroku memungkinkan Anda menggunakan dukungan Websocket eksperimental menggunakan lab: aktifkan perintah pada CLI Heroku. Untuk mengaktifkan dukungan Websocket untuk aplikasi Anda, ketikkan perintah berikut:

$ heroku labs: aktifkan Websockets -sampel app
Mengaktifkan Soket Web untuk sampel app ... selesai
PERINGATAN: Fitur ini eksperimental dan dapat berubah atau dihapus tanpa pemberitahuan.

Setelah fitur ini diaktifkan untuk aplikasi Anda, catatan DNS untuk domain herokuapp.com Anda secara otomatis diperbarui untuk menunjuk ke titik akhir yang mampu Websocket. Ketika datang ke penggunaan domain khusus, mereka harus dikonfirmasi dengan benar untuk memastikan ada dukungan Websocket yang konsisten pada semua domain terkait.

Mematikan Soket Web

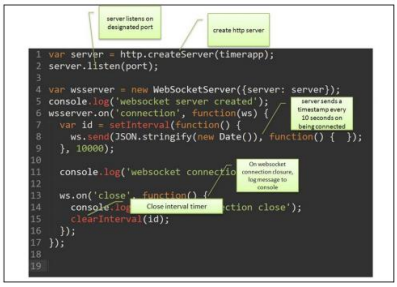
Menonaktifkan dukungan Websocket untuk aplikasi Heroku Anda sangat mudah. Cukup gunakan labs: nonaktifkan perintah sebagai berikut untuk menghentikan Websockets yang mendukung:

$ heroku labs: nonaktifkan websockets -a sampleapp
Menonaktifkan soket web untuk aplikasi sampel ... selesai
Contoh Websockets

Contoh Websockets berikut menunjukkan kode klien dan server minimal yang ditulis dalam JavaScript. Dalam kode, aplikasi penghitung waktu sederhana membuka Websocket ke server dan menerima cap waktu dari server setiap detik, yang kemudian ditampilkan pada halaman.

Kode server

Server ditulis dalam JavaScript sisi server sebagai aplikasi Node.js. Dalam aplikasi ini, ketika server terhubung ke klien, itu menciptakan metode yang berjalan setiap 10.000 ms, mengirimkan timestamp ke browser melalui Websocket seperti yang ditunjukkan pada tangkapan layar berikut.




```
var server = http.createServer(timerapp);
server.listen(port);

var wsserver = new WebSocketServer({server: server});
console.log('websocket server created');
wsserver.on('connection', function(ws) {
  var id = setInterval(function() {
    ws.send(JSON.stringify(new Date()), function() {  });
  }, 10000);

  console.log('websocket connectio

  ws.on('close', function() {
    console.lo                    ction close');
    clearInterval(id);
  });
});
```

Kode klien

Dengan asumsi bahwa browser mendukung fitur Websockets, klien membuat koneksi dengan server. Kode klien berikut yang ditulis dalam JavaScript mem-parsing acara server dan menambahkannya ke elemen daftar HTML yang kemudian ditampilkan di browser:


```
var host = location.origin.replace(/^http/, 'ws')
var wsc = new WebSocket(host);
wsc.onmessage = function (event) {
  var li =                nt.createElement('li');
  li.inner                JSON.parse(event.data);
  document.querySelector('#pings').appendChild(li);
};
```

Contoh klien dan server hanyalah contoh sederhana dari vanilla tentang bagaimana penggunaan Websockets API. Anda dapat menggunakan contoh-contoh ini sebagai dasar untuk mengembangkan kasus penggunaan yang lebih kompleks.

Ini melengkapi ulasan kami tentang beberapa fitur Heroku Labs yang menonjol. Selanjutnya, kita akan melihat Heroku Platform API – cara yang ampuh untuk membangun, menyebarkan, dan mengelola aplikasi Heroku secara terprogram. Untuk ikhtisar singkat tentang API Platform Heroku, silakan merujuk ke Bab 2, Di Dalam Heroku. Dalam bab ini, kita akan memahami fasilitas yang tersedia di API platform Heroku untuk menulis dan memelihara aplikasi Heroku. Kami juga akan meninjau beberapa sampel tentang bagaimana pengembang dapat memanggil platform API untuk melakukan berbagai operasi yang terkait dengan aplikasi web Heroku.

Panggilan API Platform Heroku tinju Anda

Beberapa perpustakaan klien sudah tersedia untuk membuat panggilan API Platform Heroku. Heroku secara alami mendukung perpustakaan klien Heroics untuk Ruby. Pada saat penulisan teks ini, ada pustaka klien yang tersedia untuk Node.js, Scale, dan Go.

Karena klien API menggunakan HTTP untuk berinteraksi dengan Heroku Platform API, yang kita butuhkan untuk membuat dukungan API untuk bahasa pemrograman baru adalah perpustakaan HTTP dengan metode yang diperlukan untuk mengirim dan menerima permintaan ke atau dari API platform. Atau, kita dapat menggunakan alat keriting untuk menunjukkan penggunaan platform API. Dipersenjatai dengan akun Heroku dan versi terinstal dari alat keriting, kami sekarang akan bereksperimen dengan set sampel API Platform Heroku dan menunjukkan bagaimana orang dapat dengan mudah dan intuitif membangun aplikasi Heroku secara terprogram. Ini adalah cara ketiga membuat aplikasi Heroku selain Heroku CLI dan dashboard Heroku.

Sebelum kita mulai

Nah, sebelum Anda dapat mengakses API platform, klien Anda harus diautentikasi. Token otentikasi kemudian digunakan dalam panggilan yang akan datang ke platform untuk memverifikasi akses klien.

Kunci API tersedia di bagian Akun di dasbor Heroku Anda. Anda juga dapat menggunakan perintah heroku auth: token untuk mengambil token ini:

```
$ heroku auth: token
0123456-89ab-cdef-0123-456789 abcdef
```

Anda juga dapat memperoleh kunci otorisasi dari perintah ini untuk skrip dengan mengakses kunci menggunakan perintah heroku auth: token dan mengonversinya menjadi nilai base-64. Jika Anda menggunakan curl bersama dengan Heroku toolbelt, curl dapat menangani detail otentikasi dengan membaca netrc fie.

Metode API yang didukung

Klien dapat mengakses beberapa metode API Platform Heroku untuk memanipulasi aplikasi tergantung pada kebutuhan pemanggil. Daftar metode yang tersedia untuk klien ditunjukkan pada tabel berikut:

Metode Tujuan
POST Menciptakan objek baru
GET Mengambil objek atau daftar objek
PUT Mengganti objek yang ada
HAPUS Menghapus objek yang ada
HEAD Mengambil metadata tentang objek yang ada
PATCH Memperbarui objek yang ada

Jika klien tidak mendukung metode tertentu, itu dapat menimpa header permintaan dengan menggunakan metode POST dan mengatur header X-Http-Method-Override ke metode yang dimaksud. Sebagai contoh, untuk melakukan doa PATCH, klien dapat melakukan doa POST dengan Http-Method-Header diatur ke PATCH dalam permintaan klien.

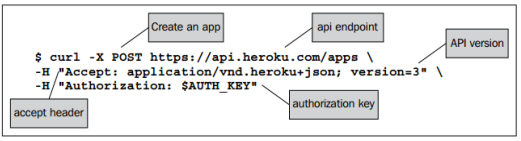
Contoh penggunaan API platform

Anda dapat melakukan hampir semua yang Anda bisa dengan Heroku CLI atau dasbor dengan menggunakan platform API. Keuntungan utama menggunakan platform API adalah memungkinkan pengembang untuk memiliki kontrol penuh atas seluruh siklus hidup aplikasi Heroku. Pengembang dapat melakukan operasi terkait aplikasi apa pun dari kenyamanan cuplikan kode.

Dalam contoh berikut, kami akan membuat aplikasi, mencantumkan informasi tentang aplikasi, memperbarui informasi aplikasi, dan menghapus aplikasi tersebut menggunakan API platform. Dalam prosesnya, kami akan meninjau konten permintaan dan respons khas panggilan platform API. Kami akan menggunakan alat keriting untuk menunjukkan penggunaan API Platform Heroku dalam contoh-contoh ini.

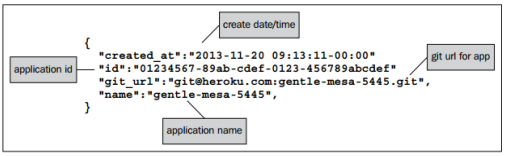
Membuat aplikasi

Untuk membuat aplikasi Heroku, gunakan perintah ikal berikut. Diagram berikut menunjukkan komponen berbeda dari perintah curl yang diperlukan untuk membuat aplikasi:



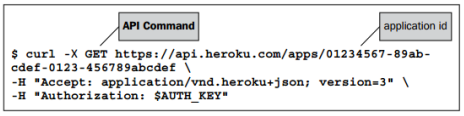
Buat respons API aplikasi

Setelah berhasil menyelesaikan suatu tindakan, platform API mengembalikan respons JSON dengan perincian aplikasi Heroku yang baru dibuat sebagai berikut:

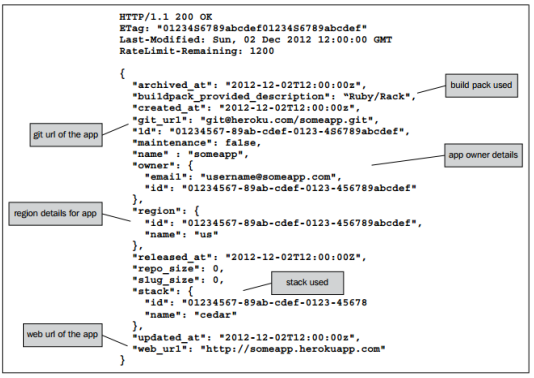


Mengambil informasi aplikasi

Permintaan API meneruskan ID aplikasi ke titik akhir API bersama dengan perintah GET untuk mengambil detail lebih lanjut tentang aplikasi:



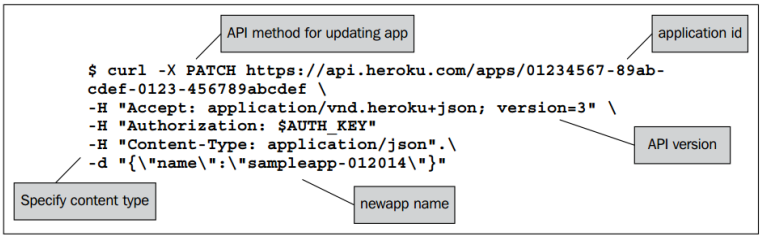
Respons API mencakup semua detail aplikasi sebagai berikut:



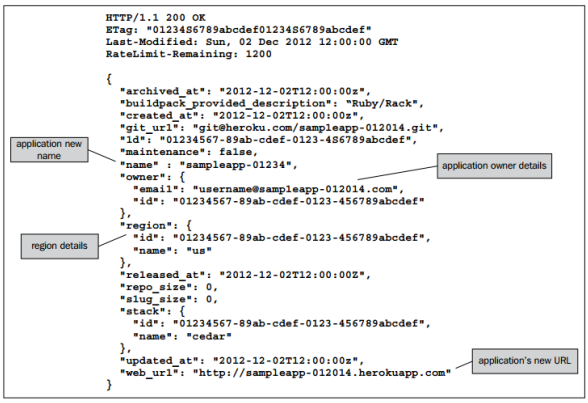
Memodifikasi informasi aplikasi

Anda dapat mengubah properti aplikasi, termasuk namanya, dengan menggunakan perintah PATCH. Berikan ID aplikasi ke titik akhir API dan tentukan nama baru, versi API, dan jenis konten.

Anda dapat mengeluarkan perintah ikal berikut untuk memperbarui nama aplikasi Heroku:



Respons API terhadap perintah pembaruan akan serupa dengan respons JSON dari panggilan GET sebagai berikut:



Menghapus aplikasi

Anda dapat menghapus aplikasi Heroku menggunakan perintah DELETE API, meneruskan ID aplikasi sebagai parameter.

Fiure berikut menunjukkan perintah curl untuk menghapus aplikasi. Anda harus melewati ID aplikasi (atau nama) ke titik akhir API selain menggunakan perintah DELETE untuk menghapus aplikasi:

Menafsirkan respons API

Respons API Heroku Platform yang berhasil dapat dikategorikan berdasarkan statusnya sesuai tabel berikut:

Status Berarti
200 ok permintaan API berhasil
201 dibuat Permintaan API menghasilkan sumber daya yang dibuat atau sumber daya baru ditambahkan ke aplikasi (untuk penyediaan tambahan)
202 diterima Permintaan API yang oleh platform tetap pemrosesan tidak belum selesai
206 Sebagian Konten Permintaan API berhasil tetapi respons yang diterima hanya sebagian dari seluruh respons atau bagian dari rentang respons keseluruhan yang diharapkan

Platform API dapat merespons klien dengan kesalahan jika terjadi kesalahan selama pemrosesan server; sebagai alternatif, permintaan klien itu sendiri bisa salah dan harus diperbaiki sebelum mengirim permintaan lagi. Oleh karena itu, respons yang keliru diklasifikasikan berdasarkan apakah kesalahan yang berhubungan dengan klien atau yang terkait dengan server.

Operasi kesalahan

Respons kesalahan terkait klien biasanya merupakan hasil dari permintaan klien yang cacat atau tidak memadai. Ini bisa berupa permintaan yang tidak valid atau yang tidak sah. Respons kesalahan klien dapat terjadi sebagai akibat dari beberapa tindakan yang diperlukan terkait dengan akun yang terlibat atau sumber daya yang diminta tidak ada. Parameter bisa tidak valid, atau akun mungkin perlu verifikasi sebelum mengeluarkan panggilan API. Alasannya bisa banyak, tetapi membantu untuk memeriksa beberapa hal untuk memastikan bahwa kesalahan diminimalkan:

- Pastikan informasi penagihan dimasukkan untuk akun, sehingga Heroku dapat menyimpan catatan penggunaan API.
- Berikan bentuk permintaan API yang benar.
  Respons kesalahan terkait server hanya dari dua jenis: kesalahan internal (HTTP 500) atau kesalahan tidak tersedia layanan (HTTP 503).

Format kesalahan

Ketika API platform meminta sumber daya tertentu gagal dan respons kembali dengan status gagal, badan JSON yang dikembalikan berisi rincian lebih lanjut tentang kesalahan. Responsnya berisi pengidentifikasi atau ID unik untuk kesalahan dan pesan terkait di pesan fild. Pesan fild membantu entitas panggilan atau klien memahami akar penyebab permintaan gagal.

Format respons kesalahan yang diterima ditunjukkan pada tabel berikut:

Nama	Tipe	Deskripsi
ID	Tali	Identifier unik untuk kesalahan
Pesan	Tali	teks pesan kesalahan terperinci untuk pengguna akhir

Contoh respons kesalahan

Kapan pun klien mengeluarkan lebih dari batas permintaan API, API platform mengembalikan respons yang menunjukkan bahwa klien telah melampaui batas laju API. Ini mungkin mengharuskan klien menunggu beberapa saat untuk mengeluarkan kembali permintaan atau memodifikasi rencana penagihan untuk membeli batas API yang lebih besar. Apapun yang klien pilih, klien bisa mengeluarkan permintaan hanya ketika batas API dipulihkan atau ditingkatkan.

Misalnya, respons berikut dikeluarkan dari platform API untuk melebihi batas akses API:

{**"id": "menyetujuilimit", "message": "Akun Anda mencapai batas tingkat API \ n Silakan tunggu beberapa menit sebelum membuat permintaan baru** "}

Peringatan

Ketika panggilan platform API menghasilkan peringatan, tajuk penambahan respons API berisi uraian terperinci tentang peringatan untuk membantu klien mengambil tindakan yang tepat saat mengirim permintaan lebih lanjut. Add header berisi ID fild dan deskripsi yang sesuai dari pengidentifikasi peringatan unik.

Berbagi aplikasi Anda di Heroku

Salah satu faktor pendorong untuk popularitas dan kesuksesan banyak teknologi saat ini adalah konsep pengembangan sosial atau kolaboratif. Ini mencakup tidak hanya kemampuan untuk menulis aplikasi, tetapi juga kesempatan untuk memelihara dan memutakhirkannya.

Heroku percaya pada filosofi kolaborasi sosial untuk membangun aplikasi web yang benar-benar bebas bug dan terpelihara. Heroku menyediakan fasilitas untuk berkolaborasi dan berbagi aplikasi Anda dengan pengguna lain yang dapat melakukan hampir semua hal yang dapat dilakukan oleh pemilik aplikasi.

Prasyarat untuk kolaborasi

Anda dapat menambahkan pengguna lain sebagai kolaborator untuk aplikasi Anda dalam dua cara berikut:

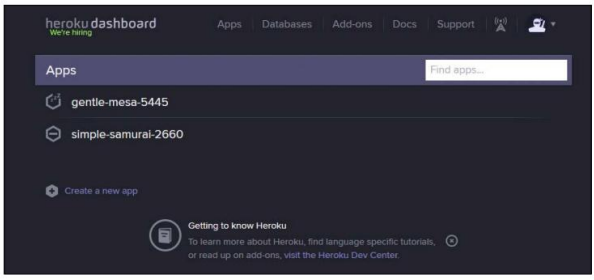
- Dasbor Heroku
- The Heroku CLI

Prasyarat untuk berbagi aplikasi Anda termasuk memiliki akun Heroku dan Heroku toolbelt (http://toolbelt.heroku.com). Menginstal toolbelt Heroku memberi Anda akses ke Heroku CLI dan sistem kontrol revisi Git.

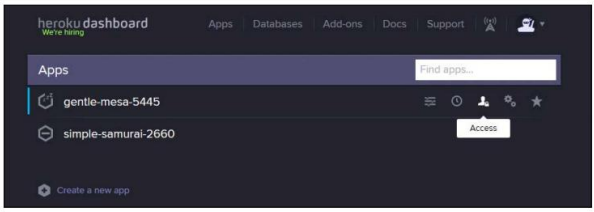
Menambahkan kolaborator aplikasi ke dasbor Heroku

Lakukan langkah-langkah berikut untuk menambahkan kolaborator aplikasi ke dasbor Heroku:

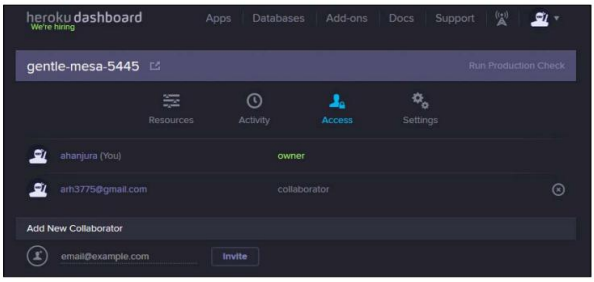
1. Untuk menambahkan kolaborator menggunakan dasbor Heroku, Anda perlu login terlebih dahulu ke akun Heroku Anda. Setelah Anda masuk, Anda akan melihat daftar aplikasi Heroku Anda, seperti yang ditunjukkan pada tangkapan layar berikut:



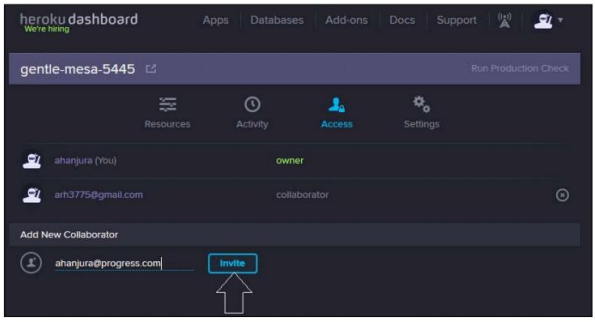
3. Untuk setiap aplikasi yang ditampilkan dalam daftar, Anda akan melihat serangkaian ikon untuk berbagai operasi yang dapat dilakukan pada aplikasi. Klik ikon Access seperti yang ditunjukkan pada tangkapan layar berikut:



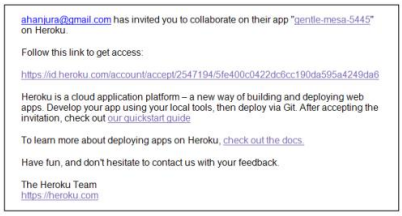
4. Layar berikut akan menampilkan daftar pengguna (pemilik dan kolaborator untuk aplikasi ini). Selain itu, Anda dapat menambahkan kolaborator baru dengan mengirim undangan melalui email:



5. Tambahkan kolaborator dengan memasukkan alamat email yang valid dan mengklik Undang:

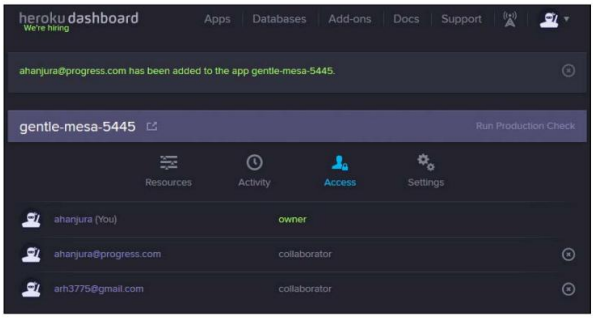


6. Email dikirimkan kepada pengguna yang ditambahkan sebagai kolaborator, memberi tahu pengguna baru bahwa ia telah diberikan akses ke aplikasi. Jika tidak ada akun Heroku yang sesuai dengan spesifikasi email, email undangan dikirim seperti yang ditunjukkan pada tangkapan layar berikut:



Kolaborator dapat mengklik tautan yang dikirim melalui email dan mendaftar untuk Heroku jika pengguna belum memiliki akun Heroku.

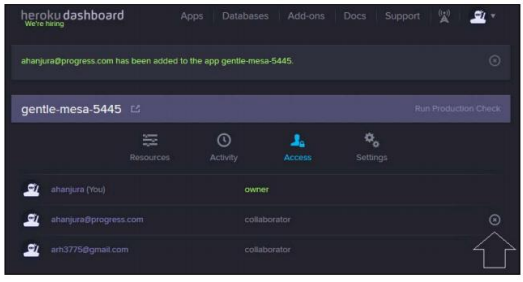
7. Pengguna ditambahkan ke daftar kolaborator seperti yang ditunjukkan pada tangkapan layar berikut:



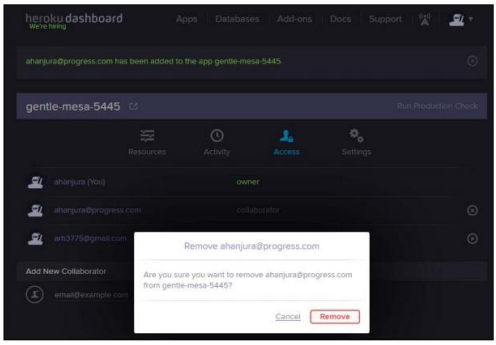
Menghapus kolaborator

Menghapus kolaborator dapat dilakukan melalui langkah-langkah berikut:

1. Klik pada tanda silang di kanan ekstrim nama kolaborator seperti yang ditunjukkan pada tangkapan layar berikut:



2. Klik pada kotak dialog konfirmasi untuk menghapus kolaborator untuk menyelesaikan operasi penghapusan:



Menambahkan kolaborator melalui Heroku CLI

Anda dapat menambahkan pengembang lain untuk berkolaborasi pada aplikasi Anda menggunakan berbagi: tambahkan perintah sebagai berikut:

$ heroku sharing: tambahkan johndoe@sampleapp.com
Menambahkan johndoe@sampleapp.com ke kolaborator lembut-mesa-5445 ...selesai

Kolaborator yang ditambahkan dalam proses ini dapat melakukan semua operasi yang dapat dilakukan pemilik, kecuali beberapa yang termasuk menambah atau menghapus add-on berbayar, menghapus atau mengganti nama aplikasi, atau melihat faktur akun.

Daftar kolaborator

Untuk mendaftar semua kolaborator untuk suatu aplikasi, gunakan perintah berbagi heroku berikut:

berbagi $ heroku
=== Kolaborator sampelapp
madana@sampleapp.com
johndoe@sampleapp.com

Menghapus kolaborator

Untuk mencabut akses kolaborator, gunakan berbagi heroku berikut: hapus perintah:

$ heroku sharing: hapus johndoe@sampleapp.com
Menghapus johndoe@sampleapp.com dari kolaborator lembut-mesa-5445 ...selesai

Setelah berhasil mencabut akses kolaborator, kolaborator tidak dapat lagi menerapkan perubahan kode apa pun atau mengubah konfigurasi aplikasi.

Tindakan kolaborator

Setelah kolaborator ditambahkan ke aplikasi, kolaborator dapat melakukan beberapa operasi pada aplikasi, termasuk memodifikasi aplikasi itu sendiri dan melihat detail aplikasi. Di bagian ini, kita melihat langkah-langkah khusus yang dapat diambil kolaborator untuk memodifikasi aplikasi dan melihat detailnya.

Bekerja pada aplikasi

Sekarang kolaborator telah menerima akses ke aplikasi, sekarang saatnya untuk menggunakan aplikasi. Untuk menggunakan aplikasi dan memodifikasinya, kolaborator perlu melakukan yang berikut:

1. Mengkloning aplikasi secara lokal. Gunakan heroku git: clone --app sampleapp perintah untuk mengkloning aplikasi.
2. Sebagai alternatif, Anda bisa mendapatkan akses ke repositori kanonik pemilik aplikasi dan kemudian menggunakan heroku git: remote untuk menambahkan remote Git ke checkout Anda.
3. Edit kode sumber.
4. Gunakan git push heroku untuk menyebarkan komit lokal. Pastikan perubahan diperbarui ke repositori kanonik aplikasi (menggunakan perintah master asal git push) serta ke remote yang dibuat sebelumnya. Ini untuk menghindari masalah sinkronisasi kode dengan sesama kolaborator.

5. Jika Anda mendapatkan kesalahan yang mirip dengan yang berikut ini, akan ada potensi konflik antara remote dan repositori lokal Anda di luar perubahan yang telah Anda lakukan:

   $ git push heroku kesalahan: 'ref / kepala / master' remote bukan subset lokal yang ketat ref 'refs / heads / master'. mungkin Anda tidak up-to-date dan perlu menarik dulu?

6. Tarik ke bawah perubahan terbaru menggunakan perintah git pull –rebase dari master jarak jauh, gabungkan dengan kode Anda, lalu coba dorong kembali kode tersebut.

Melihat aplikasi

Saat mendapatkan akses ke aplikasi Anda, kolaborator dapat memverifikasi detail aplikasi menggunakan perintah info heroku yang sudah dikenal sebagai berikut:

$ heroku info --app sampleapp
=== sampleapp
URL web: http://sampleapp.herokuapp.com/
Git Repo: git@heroku.com: sampleapp.git
Ukuran Repo: 1024k
Ukuran siput: 512k
Email Pemilik: owner@sampleapp.com
Kolaborator: madana@sampleapp.com
johndoe@sampleapp.com

Ringkasan

Dalam bab penutup ini, kami belajar tentang beberapa fitur eksperimental dari platform Heroku yang membantu kami membangun, menyebarkan, dan memantau aplikasi kami dengan lebih baik. Kami mengeksplorasi dukungan eksperimental Heroku untuk protokol komunikasi dua arah yang kuat yang disebut Websockets – pilihan alami untuk mengembangkan aplikasi web waktu-nyata. Kami mendapat pengantar tentang penggunaan API Platform Heroku untuk membangun aplikasi Heroku secara terprogram. Kami juga membahas cara kerja kolaborasi pada

platform Heroku, yaitu, bagaimana Anda dapat menggunakan dasbor Heroku dan / atau Heroku CLI untuk menambahkan kolaborator Anda dan membagikan aplikasi Anda dengan mereka secara instan.

Ini tidak mengakhiri perjalanan kami dari Heroku. Ini baru permulaan. Permintaan akan platform yang kuat, terukur, dan berkinerja tinggi sebagai layanan lebih besar dari sebelumnya. Semakin banyak penyedia yang memungkinkan pengembang untuk membangun aplikasi yang lebih cepat menggunakan alat pengembangan cloud terbaru. Fakta bahwa platform pengembang untuk cloud berkembang pesat memberi tahu kita bahwa lebih banyak inovasi sedang berlangsung. Sumber terbuka baru

platform seperti Flynn (www.flynn.io) dan versi open source dari Heroku, Openruko (https://github.com/openruko), juga menarik banyak kontribusi dan perhatian pengembang. Buku ini telah mencoba membiasakan pembaca dengan bagaimana sebagai pengembang, Anda dapat memanfaatkan kemampuan platform Heroku dan membangun aplikasi web yang kuat, dapat diukur, dan berkinerja tinggi. Semoga sudah memenuhi kebutuhan itu sampai batas tertentu.

TENTANG PENULIS

Nama saya Dian Resha Agustina, lahir di Bandung pada tanggal 20 Agustus 1995. Pendidikan yang saya tempuh pada strata-1 di Univeristas Bandar Lampung, Jurusan Informatika. Kemudian saya lulus pada 2020 dengan predikat terbaik di Magister Teknik Informatika kampus IIB Darmajaya. Saya sudah menekuni bidang teknokogi semenjak mengambil kejuruan Animasi di SMKN 5 Bandar Lampung yaitu di tahun 2010 sampai dengan saat ini. Saya berharap untuk terus dapat memberikan ilmu bermanfaat yang saya punya di bidang teknologi tentang cloud computing, grafika Komputer, pemrograman game, pengolahan citra digital, human-computer interaction ataupun lainnya. Saya berprofesi sebagai dosen di salah satu kampus swasta di Indonesia. Anda dapat kontak saya di email dian.resha@ubl.ac.id atau dianfikubl@gmail.com juga channel youtube Gadis Informatika dan instagram @dianresha.

Onno Widodo Purbo (lahir 17 Agustus 1962) sebagai pakar di bidang teknologi informasi asal Indonesia. Selain pakar, Onno juga dikenal sebagai penulis, pendidik, dan pembicara seminar. Sebagai aktivis Onno dikenal dalam upayanya memperjuangkan Linux. Karya inovatifnya diantaranya adalah Wajanbolic, sebagai upaya koneksi internet murah tanpa kabel dan RT/RW-Net sebagai jaringan komputer swadaya masyarakat untuk menyebarkan internet murah, serta penerapan Open BTS.