BAB I
PENDAHULUAN

Komputasi awan beberapa tahun belakangan menjadi topik yang menarik. Komputasi awan adalah generasi lanjut dalam pengelolaan sumber daya (resource). Saat ini komputasi awan mulai berkembang dengan pesat karena para pelanggan tidak perlu memikirkan masalah pemeliharaan, skalabilitas, dan keamanan server namun pelanggan dapat memegang penuh atas resource dari server tersebut. Pelanggan individu maupun perusahaan dapat menyewa layanan cloud tersebut melalui pihak ketiga yang menyediakan layanan cloud melalui internet. Layanan Cloud dibagi menjadi 3 bagian utama yaitu Software as a Service (SaaS), Platform as a Service (Paas), dan Infrastructure as a Service (IaaS) (Somasundaram & Govindarajan, 2014).

Komputasi awan menjadi solusi popular untuk media penyimpanan data dan eksekusi aplikasi oleh pengguna dengan ketentuan pay-per-use yaitu membayar sesuai dengan penggunaan resource saja. Publik cloud banyak disediakan oleh beberapa perusahaan seperti Amazon, Google, Mirosoft, atau Rackspace. Bagi perusahaan yang ingin membuat sistem cloud mereka sendiri terdapat beberapa solusi seperti Eucalyptus, OpenStack, dan Proxmox yang berfungsi sebagai virtual environment untuk sistem cloud. Pada model infrastruktur sebagai layanan, salah satu masalah yang cukup besar adalah manajemen penggunaan sumber daya pada mesin virtual, di mana sumber daya pada mesin virtual dapat digunakan layaknya sumber daya pada mesin nyata (Rathore, 2012).

Alasan mengapa semua skalabilitas sampai dalam tahap teknologi informasi adalah karena fakta bahwa pelanggan menuntut deployment services yang cepat di layanan cloud. Dengan munculnya Amazon EC2 Instances proses pembuatan environment membutuhkan waktu lebih singkat, tidak seperti dulu, dimana pelanggan membutuhkan waktu sekitar 1 minggu untuk mulai dapat menjalankan server[1]. Penyedia IAAS memberikan fleksibilitas penuh kepada pelanggan untuk penyediaan komputasi, penyimpanan serta sumber daya jaringan di layanan cloud. Pelanggan selalu menyukai

1Elastic Compute Cloud (EC2), Amazone Web Services, Amazon.com, http://aws.amazon.com/ec2

pendekatan tersebut dan menuntut respon yang cepat dari vendor IAAS. interaksi langsung dengan penyedia cloud membuat pelanggan tertarik untuk melakukan one on one provisioning namun karena keterbatasan resources tools, hal ini tidak dapat diterapkan. Selain itu, klien yang memiliki private cloud memiliki kontrol penuh terhadap cloud spaces dibandingkan dengan grid atau cluster computing; sehingga aplikasi dapat disesuaikan kapan saja dan sesuai. Tetapi hal ini menimbulkan permasalahan yang lebih kompleks seperti kompleks pada konfigurasi dan proses deploy aplikasi dari sisi pengguna itu sendiri (Gentzsch, 2001).

Aplikasi yang berjalan pada layanan cloud berjalan selama beberapa jam dan sistem pembayaran berdasarkan penggunaan resources, hal ini juga selalu dalam monitoring devops. Keseluruhan proses ini dikelola oleh devops untuk meningkatkan efisiensi dari aplikasi milik pengguna (Beach, 2014). Kebutuhan otomatisasi tidak hanya berhenti di sini, karena kebutuhan terus meningkat untuk memudahkan dan melakukan standarisasi seluruh prosedur dari permintaan. Dulu proses deployment application dilakukan dengan menggunakan batch script yang penggunaanya mulai tidak reliable (Singh, Thakur, Chaurasiya, & Nagdev, 2015).

Dalam penelitian ini, penulis menggunakan manajemen konfigurasi pada model layanan cloud IAAS. Penelitian juga akan membuktikan bagaimana application provisioning dapat dilakukan pada layanan cloud dengan rancangan arsitektur dan memberikan pengetahunan praktis proses automisasi pada server.

1.2. Rumusan Masalah

Rumusan masalah dari latar belakang diatas adalah bbagaimana application provisioning dapat dilakukan pada layanan cloud dengan rancangan arsitektur dan memberikan pengetahunan praktis proses automisasi pada server

1.3. Batasan Masalah

Adapun batasan-batasan masalah pada penelitian ini adalah :

1. Penelitian menggunakan google cloud platform
2. Penelitian menggunakan ansible untuk configuration management

3. Parameter yang digunakan untuk perbandingan adalah waktu proses penyelesain tugas dari masing-masing server.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Implementasi configuration management untuk mengelola banyak server
2. Menangani permasalah waktu server provisioning pada layanan cloud

1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah menjadi acuan dalam pengembangan implementasi layanan cloud dan proof of concept automating provisioning pada layanan cloud.

1.6. Target Luaran

Dokumentasi dalam pengembangan implementasi layanan cloud dan proof of concept automating provisioning pada layanan cloud.

BAB II
TINJAUAN PUSTAKA

Tinjauan pustakan yang digunakan pada penelitian ini meliputi beberapa penelitian yang sudah dilakukan sebelumnya, sebagai berikut.

Karena besar dan kompleks topologi layanan cloud sehingga sulit untuk melakukan configuration management dengan cara lama, komunitas DevOps kemudian membuat tools yang memiliki fitur untuk menangani proses automisasi dan kompleks dengan tool seperti Marionette collective dan Spiceweasel [2]. Alat-alat ini memberikan lebih banyak kemudahan untuk mengelola cluster dan node collection . Marionette collective menyediakan framework yang dapat digunakan untuk menghubungkan baik puppet maupun chef sebagai configuration managemen. Turnbull, mengatakan bahwa spiceweasel terdiri dari command berbasis Chef untuk memudahkan pengelolaan satu kelompok node, tetapi jauh lebih sederhana dan lebih terbatas dibandingkan dengan Marionette collective. Konfigurasi dari kumpulan node dapat digambarkan menggunakan dokumen yang ditulis dalam JavaScript Object Notation (JSON). Kemudian, beberapa command Chef yang dihasilkan berdasarkan dokumen ini untuk melakukan proses deployment ke layanan cloud (Turnbull & McCune, 2011).

Penelitian yang disajikan dengan konsep untuk mengintegrasikan manajemen konfigurasi dengan manajemen Cloud model-driven. Tujuannya adalah untuk menyatukan kelebihan dari kedua service management paradigms. Hal ini dicapai berdasarkan TOSCA, standar yang muncul untuk mewujudkan model-driven cloud management. Konsep yang berbeda digambarkan bagaimana memasukkan definisi konfigurasi pada CSAR. Kedua pendekatan integrasi dapat dikombinasikan untuk membuat CSAR yang sangat portabel. Konsep integrasi ini mengikuti prinsipdesain TOSCA dan pelaksanaannya tidak memerlukan perpanjangan TOSCA sendiri. Integrasi configuration definitions menjadi dasar untuk menciptakan CSAR portabel (Wettinger et al., 2013).

2http://wiki.opscode.com/display/chef/Spiceweasel

Cloud computing menjadi awal perkembangan proses provisionig dan deployment. Proses deployment pada infrastruktur AWS memberikan kemudahan dan mempercepat produksi aplikasi. Peneliti menggunakan ansible untuk configuration management tools pada infrastruktur amazon web services, pengujian terhadap ansible memiliki fleksibilitas tinggi dalam melakukan seluruh prosedur deployment application (Singh et al., 2015).