BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

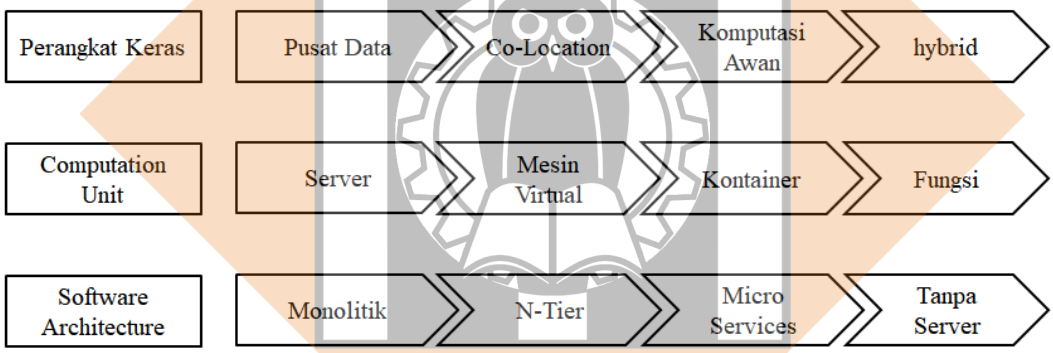
Selama beberapa dekade terakhir, telah terjadi inovasi di industri *information technology* (IT). Fokus utama dari inovasi teknologi ini bertujuan untuk mendukung kelincahan dan ketahanan bisnis, dan tentu saja mendorong efisiensi biaya. Teknologi komputasi tanpa *server* telah mendukung industri IT untuk mengembangkan dan menggunakan perangkat lunak tanpa perlu khawatir mengenai perangkat keras yang berjalan di bawahnya.

Revolusi di bidang komputasi *server* adalah komputasi awan, yang mana telah diadopsi secara meluas sebagai *Infrastructure as a Service* (IaaS) dan *Platform* as a Service (PaaS). Komputasi awan IaaS memberi kemampuan untuk menyediakan sumber daya perangkat keras sesuai kebutuhan dengan tingkat waktu ketersediaan lebih cepat dibandingkan apabila disediakan sendiri. Sebagai ilustrasi untuk menyiapkan perangkat keras di komputasi awan diperlukan waktu kurang lebih dua menit [1], lebih cepat dibandingkan menyiapkan perangkat keras sendiri yang membutuhkan waktu beberapa minggu agar perangkat tersebut dapat digunakan, mulai dari proses pembelian, hingga melakukan instalasi dan konfigurasi.

Komputasi awan IaaS juga menghilangkan kebutuhan modal yang diperlukan untuk belanja perangkat keras. Ini semua disebabkan oleh teknologi virtualisasi yang memungkinkan untuk berbagi sumberdaya perangkat keras. Komputasi awan PaaS menyediakan pondasi untuk pengembangan aplikasi *server* terstandarisasi dengan memanfaatkan komponen yang sudah ada. Berdasarkan data yang

diterbitkan oleh *IDC* diperkirakan pada tahun 2020, 67% dari total belanja Industri IT akan digunakan untuk layanan berbasis komputasi awan [2].

Revolusi komputasi awan memberi pengaruh pada arsitektur perangkat lunak, unit komputasi dan perangkat keras. Arsitektur perangkat lunak bergerak dari monolitik menuju tanpa *server*, karena didukung oleh perubahan satuan unit komputasi dari mesin virtual, kontainer dan fungsi yang berjalan dalam kontainer. Di sisi perangkat keras *enterprise*, evolusi telah terjadi dari pusat data, *co-location*, komputasi awan dan *hybrid*. Komputasi awan *hybrid* memungkinkan *enterprise* untuk menggabungkan pusat data dan komputasi awan, serta menikmati keleluasaan komputasi tanpa *server* [3]. Gambar 1.1 menunjukkan evolusi teknlogi komputasi.



Gambar 1.1 Evolusi Teknologi Komputasi Awan dan Pengaruhnya Terhadap Arsitektur Perangkat Lunak, Satuan Unit Komputasi dan Perangkat Keras

Komputasi tanpa *server* adalah evolusi model layanan komputasi awan –dari IaaS ke PaaS hingga fungsi sebagai layanan. FaaS selangkah lebih maju dalam hal mengabstraksi seluruh *runtime* pemrograman untuk menyediakan opsi yang jauh lebih mudah, sehingga pengguna cukup berfokus pada kode aplikasi tanpa perlu mengelola infrastruktur dan *platform* yang mendasarinya. Tabel 1.1 menunjukan beberapa perbedaan antara layanan IaaS, PaaS dan FaaS. Pada layanan FaaS

satuan *deployment* terkecil disebut fungsi yang mengeksekusi kode logika bisnis saat dibutuhkan.

Tabel 1.1 Perbedaan antara IaaS, PaaS dan FaaS

	IaaS	**Paas**	**Faas**
Unit *Deployment*	Sistem Operasi	Aplikasi	Fungsi
Menyediakan	Mesin virtual	*Platform* Pengembangan Aplikasi	Kode eksekusi sesuai permintaan
Abstrak	Perangkat keras	Sistem Operasi dan *middleware*	Runtime Programming

Beberapa penyedia layanan komputasi awan terkemuka seperti Amazon, Microsoft, Google telah meluncurkan layanan komputasi tanpa *server*, antara lain Amazon memiliki AWS Lambda (diluncurkan pada tahun 2014), lalu Microsoft dan Google yang memliki layanan Azure Functions (diluncurkan pada tahun 2015) dan Google Functions (diluncurkan pada tahun 2016) [4].

Berikut adalah perbandingan ketiga layanan komputasi tanpa *server* yang disediakan oleh Amazon, Microsoft dan Google seperti pada tabel 1.2.

Tabel 1.2 Perbandingan Spesifikasi Komputasi Tanpa Server AWS, Google dan Microsoft Azzure

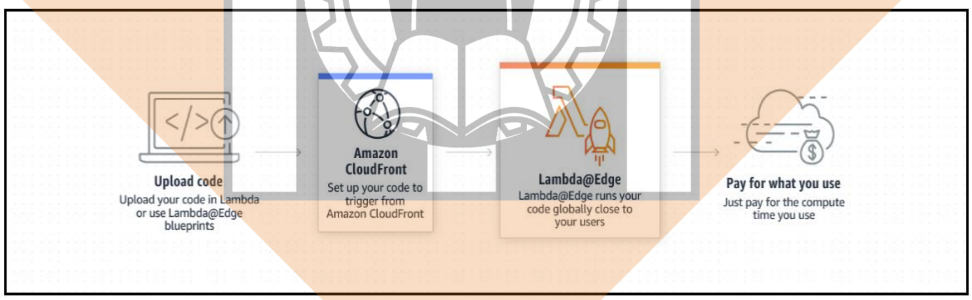
	AWS Lambda	**Google Functions**	**Azure Functions**
Tahun Peluncuran	2014	2016	2015
Skalabilitas	Otomatis	Otomatis	Otomatis
Maksimal Fungsi	Tidak terbatas	20 per project	Tergantung *trigger*
Abstrak *Programming languages supported*	JavaScript, Python, Java, Node.js,	Javascript	C#, F#, NodeJS, Python, PHP, Bash
Concurrent Excecution	100 paralel eksekusi per *account*	Tidak terbatas	Tergantung dari service aplikasi

Lisensi	*Closed source*	*Open Source*	*Open Source*
Pricing Model	*Pay as code*	*Pay as code*	*Pay as code*
Harga	Gratis 1 juta *request* pertama dan $ 0,02 1000 *request* berikutnya dan $ 0,00001667 per GB data transfer	Gratis 1 juta *request* pertama dan $ 0,02 1000 *request* berikutnya dan $ 0,00001667 per GB data transfer	Gratis 1 juta *request* pertama dan $ 0,04 1000 *request* berikutnya dan $ 0,0000231 per GB data transfer

Dari tabel 1.2 terlihat bahwa AWS Lambda menyediakan maksimal fungsi tidak terbatas dibandingkan dengan Azzure dan Google Functions. Azzure Functions menyediakan dukungan bahasa pemrograman yang lebih banyak. Skema harga yang ditawarkan adalah per *request functions* dengan akses gratis untuk satu juta *request* pertama [5].

Gambar berikut adalah *flow process* dari layanan komputasi tanpa server dari ketiga penyedia layanan komputasi tanpa server di atas [6].

1. AWS Lambda

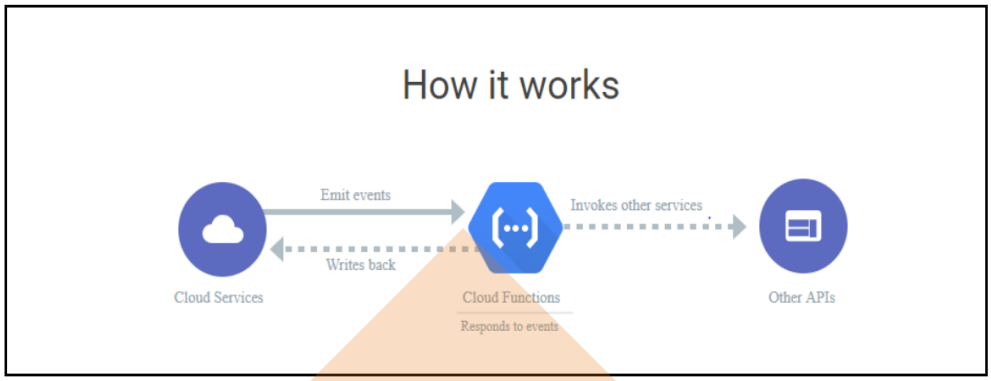


Gambar 1.2 *Flow Proccess* Pada AWS Lamda

Proses dimulai ketika kode diunduh menggunakan Amazon CloudFront yang kemudian diteruskan ke Lamda@Edge untuk menjalankan fungsi dari kode tersebut. AWS Lamda juga memiliki *feature* yang memungkinkan fungsi tersebut dijalankan di *environment* AWS terdekat dengan pengguna aplikasi tersebut, sehingga diharapkan bisa mempercepat

response. Biaya layanan dihitung berdasarkan waktu komputasi yang diperlukan untuk menjalankan fungsi dibuat oleh kode tersebut.

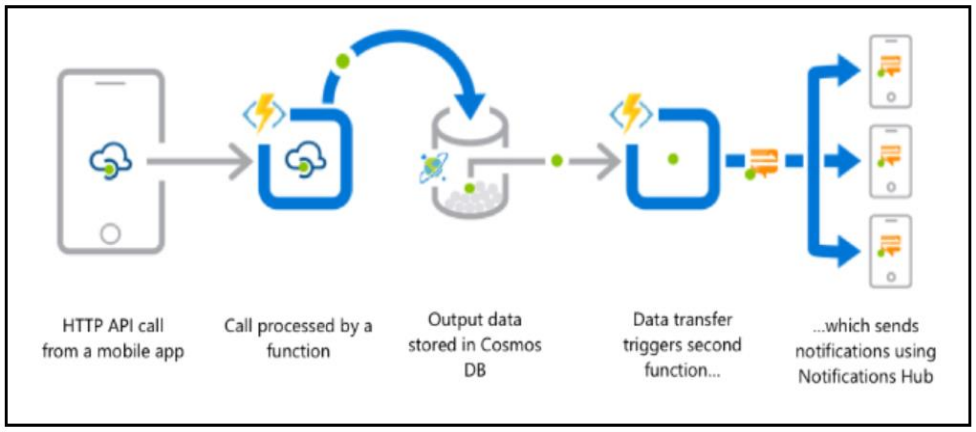
2. Google Functions



Gambar 1.3 *Flow Proccess* Pada Google Functions

Fungsi pada Google Functions terhubung ke layanan *service* komputasi awan AWS yang digunakan, misal ke IaaS dengan menggunakan Google Cloud Platform (GCP). Fungsi dipicu ketika suatu *event* terpilih telah terjadi, lalu kemudian fungsi tersebut akan bereaksi dengan memanggil layanan lainnya melalui Application Programming Interface (API). Contoh kasus penggunaan adalah skalabilitas GCP pada saat beban puncak, dimana fungsi dibuat untuk menyiapkan *instance* GCP baru ketika beban sistem sedang tinggi.

3. Azzure Functions



Gambar 1.4 *Flow Procces* Pada Azzure Functions

Fungsi dipanggil menggunakan API *call* lalu kemudian hasil luaran dari fungsi tersebut akan disimpan di Cosmos DB untuk kemudian diteruskan ke fungsi berikutnya akan mengirim notifikasi melalui Notifications HUB.

Layanan komputasi tanpa *server* yang disediakan oleh penyedia komputasi awan ini tidak semua dapat diadopsi oleh perusahaan dan organisasi, baik karena regulasi dan ketersediaan pusat data yang sudah ada atau karena layanan komputasi awan yang saat ini digunakan belum memiliki layanan FaaS.

Kebutuhan untuk menggunakan layanan FaaS di pusat data yang sudah ada dinilai cukup tinggi karena:

1. Transisi dari pusat data yang telah ada ke komputasi awan.
2. Kesiapan aplikasi sebelum pindah ke komputasi tanpa *server*.
3. Kesiapan sumber daya manusia menuju era komputasi tanpa *server*.

Kebutuhan akan layanan FaaS di pusat data sendiri dapat dicarikan solusinya melalui implementasi FaaS menggunakan *Open Source*. Ada beberapa *Open Source* FaaS yang tersedia antara lain Kubeless, OpenWhisk, Knative dan Fission [7], namun untuk menentukan pilihan OSS mana yang akan digunakan terutama untuk IT administrator dalam suatu perusahaan tentu merupakan suatu masalah

tersediri, antara lain waktu dan resiko. Masalah selanjutnya adalah bagaimana proses implementasinya dilakukan dengan mengacu kepada ketersediaan dokumentasi yang ada.

Permasalahan bagaimana menentukan pilihan OSS FaaS beserta panduan dalam implementasinya akan dijadikan sebagai topik utama dalam tugas akhir ini, yaitu **Perancangan dan Implementasi "*Functions as a Service*" untuk Komputasi Tanpa *Server* menggunakan *Open Source*.**

1.2 Perumusan Masalah

Adapun rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana implementasi FaaS menggunakan perangkat lunak terbuka (OSS- *Open Source Software*).
2. Cara Implementasi FaaS dalam mesin virtual untuk diaplikasikan dalam komputasi awan IaaS dan pusat data sendiri.
3. Bagaimana menguji kemampuan FaaS dalam melayani kebutuhan komputasi.

1.3 Tujuan dan Maanfaat

Penelitian ini ditujukan untuk menghasilkan luaran berikut ini:

1. Ada banyak pilihan OSS untuk layanan FaaS yang tersedia. Diperlukan perbandingan pilihan yang ada untuk menentukan FaaS yang sesuai dengan kebutuhan perusahaan. Penelitian ini bertujuan untuk memilih dan mengimplementasi FaaS berbasis OSS bagi IT Sistem Administrator dalam perusahaan.
2. Pengetahuan tentang arsitektur FaaS penting untuk pengembang aplikasi agar dapat menulis kode pemrograman yang sesuai. Penelitian ini juga

ditujukan untuk memberikan pengetahuan kepada pengembang aplikasi untuk memulai penulisan kode aplikasi dalam FaaS.

Jika penelitian ini dapat dilakukan dengan baik maka diperoleh manfaat sebagai berikut:

1. Memperoleh panduan untuk memilih, mengimplementasikan, menguji dan mengoptimalkan FaaS dalam Perusahaan.
2. Memberikan acuan arsitektur dan implementasi FaaS sesuai kebutuhan perusahaan menggunakan OSS.

1.4 Ruang Lingkup

Penelitian ini akan difokuskan untuk mencapai luaran sebagai berikut:

1. Perbandingan pilihan OSS FaaS yang berjalan diatas Kubernetes antara lain Knative, Kubeless, Fision, OpenWhisk dan OpenFaaS
2. Panduan implementasi mulai dari otomasi proses *provisioning* infrastruktur di komputasi awan IaaS dan pusat data sendiri, proses instalasi komponen OSS FaaS yang dipilih hingga uji coba fungsi dan performa.

Penelitian ini tidak mencakup luaran sebagai berikut

1. Perubahan kode sumber pada *Core* FaaS yang digunakan.
2. Panduan implementasi berbasis Windows dan Unix

1.5 Metodologi Penelitian

Metodologi yang digunakan dalam menyelesaikan tugas akhir ini adalah sebagai berikut:

A. Studi Kepustakaan

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang

diperlukan untuk perancangan dan implementasi sistem. Informasi didapatkan dari jurnal, buku, dan materi-materi lain yang berhubungan dangan *"Function as a Service"* untuk komputasi tanpa *server*.

B. Analisis Kebutuhan Sistem

Pada tahap ini lakukan analisis kebutuhan sistem, mulai dari pilihan OSS FaaS yang digunakan sampai kebutuhan perangkat keras dan lunak yang nanti dibutuhkan pada saat implementasi.

C. Perancangan Sistem

Tahapan perancangan sistem ini dilakukan untuk merancang bagaimana implementasi dan pengujian sistem FaaS nantinya akan dilakukan.

D. Implementasi

Pada tahap ini dilakukan implementasi sesuai rancangan yang telah dibuat. Proses implementasi ini juga menggunakan *infrastructure as a code* sehingga memudahkan proses implementasi dan pembuatan dokumentasi.

E. Pengujian

Pada tahap ini dilakukan uji coba terhadap sistem, tujuannya untuk memastikan sistem berjalan sebagaimana mestinya dan menguji FaaS dalam melayani komputasi.

1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir ini merupakan gambaran umum mengenai isi dari keseluruhan pembahasan, yang bertujuan untuk memudahkan pembaca dalam mengikuti alur pembahasan. Adapun sistematika penulisan laporan tugas akhir ini adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisikan latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah, metodologi dan sistematika penulisan laporan.

Bab II Landasan Teori

Bab ini menguraikan tentang teori yang berhubungan dengan judul tugas akhir, seperti komputasi tanpa *server*, fungsi sebagai layanan dan hal-hal terkait mengenai pembuatan system tersebut.

Bab III Analisis dan Perancangan Sistem

Bab ini menjelaskan mengenai perancangan yang dilakukan untuk membangun *function as a service.*

Bab IV Implementasi dan Pengujian Sistem

Pada bab ini menjelaskan bentuk implementasi sistem serta pengujiannya.

Bab V Penutup

Pada bab ini berisi kesimpulan dari tugas akhir dan saran untuk pengembangan lebih lanjut.