BAB 1
PENDAHULUAN

1.1 LATAR BELAKANG

Sistem komputasi terdistribusi menjadi suatu kebutuhan dalam implementasi aplikasi web beserta *software* pendukung seperti *web server* [1]. Peningkatan jumlah *web server* yang semakin banyak digunakan juga harus diikuti dengan peningkatan kualitas ataupun kuantitas sumber daya, seperti menggunakan teknologi kontainerisasi [2]. Salah satu *software* yang mengadopsi teknik kontainerisasi dan banyak diterapkan di lingkungan *web server* yaitu Kubernetes [3]. Kubernetes adalah *platform open-source* yang dirancang untuk mengotomatiskan penerapan, penskalaan, dan pengoperasian aplikasi kontainer. Dengan Kubernetes, *kontainer* dapat dengan mudah ditingkatkan (*scaled up*), dihancurkan, dan dibuat ulang. Dibandingkan dengan mesin virtual biasa, Kubernetes dapat digunakan lebih cepat, lebih efisien, dan andal [4].

Kubernetes juga sebagai orkestrasi kontainer yang dapat menyediakan *service* pada *server* dan penskalaan (*scaling*) aplikasi terkontainer di seluruh cluster [5]. *Service* pada Kubernetes *cluster* berfungsi sebagai jembatan untuk dapat mengakses satu atau beberapa *pod* [6]. *Service* didefinisikan sebagai sekumpulan *pod* logis sebagai *end-points* dan kebijakan untuk mengaksesnya. *Service* mengelompokkan *pod* berdasarkan labelnya dan bukan berdasarkan alamat IP mereka, sehingga menyembunyikan perubahan alamat IP *pod* dari klien [5]. Agar aplikasi/*klien* yang berasal diluar *cluster* Kubernetes dapat mengakses *pod*, *service* perlu di ekspos ke luar jaringan sehingga dapat diakses *pod* yang ada di belakang *service* tersebut [6].

Service dapat diekspos ke luar *cluster* Kubernetes dengan beberapa cara, yaitu *Node Port* yang bekerja dengan mengekspos *service* pada *port* yang sama dari setiap *node cluster* dan meneruskan permintaan ke *service* tujuan, *Load Balancer* diekspos secara eksternal hanya ketika *cluster* berjalan di *cloud public* dan meneruskan permintaan ke *Node Port* lalu ke *service* tujuan [5] [6]. Jika menggunakan *Node Port* ataupun *Load Balancer*, klien harus mengetahui semua IP *address* dari *node* dan *Load Balancer*. Kubernetes menyediakan cara lain yang

disebut *Ingress* untuk mengakses *service* dari luar *cluster* Kubernetes [5] yang dapat diakses dengan menggunakan domain atau nama, sehingga klien hanya butuh mengetahui satu lokasi IP address *ingress* [6].

Ingress adalah kumpulan aturan untuk koneksi masuk untuk mencapai *service* tertentu di *cluster* yang didefinisikan sebagai *backend* untuk *ingress*. Agar *ingress* dapat berfungsi di *cluster* Kubernetes, *ingress controller* harus dijalankan dan diimplementasikan di *cluster* menggunakan *controller ingress* yang tersedia seperti Kong dan Istio [5]. Ingress controller bekerja pada layer 7, sedangkan lalu lintas HTTP yang digunakan untuk web server bekerja pada protokol TCP dan UDP pada layer 4. Sehingga pada bagian tersebut, ingress controller tidak memungkinkan untuk melakukan perutean lalu lintas, maka diperlukannya penelitian lebih lanjut bagaimana kinerja ingress controller pada layer 4 seperti untuk parameter delay dan throughput. Ingress controller juga bertindak sebagai reverse proxy dan load balancer. Setiap ingress controller memiliki konfigurasi yang berbeda seperti jenis proxy yang digunakan dan konfigurasi load balancer-nya. Di mana konfigurasi pada ingress controller yang berbeda ini dapat mempengaruhi kinerja layanan. Konfigurasi tersebut juga menyebabkan setiap ingress controller memiliki keunggulan performansi yang berbeda-beda pada setiap variasi jumlah request klien [6].

Berdasarkan uraian di atas, maka diperlukan penelitian *controller ingress* sehingga dapat mengetahui *controller* mana yang memiliki performa lebih baik dan responsif penggunaannya saat *cluster* menerima banyak permintaan dari *klien* [6]. Berdasarkan referensi dari salah satu suatu artikel, maka pada penelitian ini akan menguji *ingress controller* dengan membandingkan kinerja dari *ingress controller* yaitu Kong dan Istio pada Kubernetes *cluster*. Penelitian ini menggunakan skenario pengujian peningkatan jumlah *klien* yaitu 50, 150, 250, 350, 450, 550, 650, dan 750 *klien*, serta skenario pengujian dengan kube *ingress* istio dan kube *ingress* kong. Parameter yang akan diuji yaitu *delay*, *throughput*, CPU *usage*, dan *memory usage*.

1.2 RUMUSAN MASALAH

Rumusan masalah dari penelitian ini adalah:

1) Bagaimana kinerja *ingress controller* Kong dan Istio pada Kubernetes *cluster* berdasarkan parameter *delay*, *throughput*, CPU *usage*, dan *memory usage*?
2) Bagaimana pengaruh skenario peningkatan jumlah *klien* menggunakan *ingress controller* Kong dan Istio pada Kubernetes *cluster* berdasarkan parameter *delay*, *throughput*, CPU *usage*, dan *memory usage*?

1.3 BATASAN MASALAH

Batasan masalah dari penelitian ini adalah:

1) Pengujian dilakukan pada Kubernetes *Cluster* yang terdiri dari 1 (satu) *master node* dan 4 (empat) *worker node.*
2) Pengujian penelitian ini menggunakan *ingress controller* yaitu Kong dan Istio *ingress controller*.
3) Parameter yang digunakan yaitu *delay*, *throughput*, CPU *usage*, dan *memory usage*.
4) Penelitian ini menggunakan pengujian jumlah klien dari 50, 150, 250, 350, 450, 550, 650, dan 750 klien.
5) Pengujian penelitian ini dilakukan menggunakan *Google Cloud Platform* dengan arsitektur jaringan klien-server.
6) Pengambilan data yang dianalisis menggunakan *tools Wireshark* dan Htop.

1.4 TUJUAN

Tujuan dari penelitian ini adalah:

1) Menganalisis kinerja *ingress controller* Kong dan Istio pada Kubernetes *cluster* berdasarkan parameter *delay*, *throughput*, CPU *usage*, dan *memory usage*.
2) Menganalisis pengaruh skenario peningkatan jumlah *klien* menggunakan *ingress controller* Kong dan Istio pada Kubernetes *cluster* berdasarkan parameter *delay*, *throughput*, CPU *usage*, dan *memory usage.*

1.5 MANFAAT

Penelitian ini diharapkan dapat memberikan gambaran mengenai kinerja dari controller ingress Kong dan Istio pada Kubernetes cluster. Dengan mengetahui pengaruh penggunaan controller ingress Kong dan Istio pada Kubernetes cluster diharapkan dalam implementasinya dapat memberikan informasi kinerja controller ingress sehingga mampu meningkatkan kinerja pada Kubernetes cluster.

1.6 SISTEMATIKA PENULISAN

Penelitian ini terbagi menjadi beberapa bab. Bab 1 berisi tentang latar belakang, rumusan masalah, manfaat dan tujuan penelitian, batasan masalah dan sistematika penulisan. Bab 2 membahas tentang konsep Kubernetes, service pada Kubernetes, ingress, Kong ingress controller, Istio ingress controller, serta parameter kinerja. Cara penelitian seperti alat penelitian, jalan penelitian yang meliputi parameter simulasi, pemodelan sistem, parameter kinerja, serta pengambilan hasil data dibahas pada bab 3. Bab 4 membahas tentang hasil simulasi dan analisis sistem berdasarkan hasil simulasi. Kesimpulan dan saran pengembangan penelitian untuk kedepannya dideskripsikan pada bab 5.