BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring perkembangan zaman, kini teknologi banyak diterapkan di berbagai aspek untuk mempermudah aktivitas manusia. Salah satu aspek yang terdampak oleh teknologi adalah komunikasi. Dengan berkembangnya teknologi, komunikasi dapat dilakukan dengan jarak jauh. Salah satu aktivitas ini disebut dengan *Video Conference*. Meningkatnya trafik yang di *generate* oleh pengguna layanan ini terbilang cukup tinggi dan membuat *resource* dari *server* perlahan-lahan akan habis. Dengan habisnya *resource*, maka layanan *Video Conference* akan mengalami *downtime* dan pengguna tidak dapat mengakses aplikasi tersebut. Untuk mencegah terjadinya kasus ini, seorang *Administrator System*/ DevOps dapat menggunakan *Docker* sebagai salah satu pilihan *containerized platform* sehingga *High Availability* layanan tetap terjaga dan jumlah *resource* yang akan diterima oleh *server* dapat diatur.

Salah satu fungsi dari *Docker* adalah membuat replika dengan *orchestration tool* yang digunakan adalah Kubernetes. Kubernetes merupakan sebuah *open source container platform* yang dapat mengatur *workloads* sebuah aplikasi yang telah terkontainerisasi[1]. Kubernetes sudah mengadopsi sebuah infrastruktur baru bernama *microservices*. *Microservices* bekerja dengan cara membagi aplikasi menjadi bagian-bagian kecil dimana setiap bagian tersebut menjalankan tugas yang berbeda beda pada waktu yang bersamaan. Dengan adanya pembagian layanan tersebut, maka *downtime* dapat dihindari.

Kubernetes memiliki berbagai keunggulan. Kelebihan yang pertama adalah Kubernetes lebih efisien dibanding *monolith*. *Container* baru akan ditambahkan pada *node* yang memiliki ruang lebih. Kelebihan kedua adalah *self healing*. Ketika *Cluster* Kubernetes mengalami *error*, dia dapat melakukan *auto-restore*. Kelebihan ketiga adalah *simple maintenance*. *Container* dapat dipindahkan ke *node* lain ketika *maintenance* dilakukan. Kelebihan keempat adalah *automatic scaling*. Jumlah *pod* akan bertambah dan berkurang secara otomatis sesuai dengan permintaan *user* dan penggunaan CPU agar sesuai dengan *demand* yang dibutuhkan.

Kubernetes memiliki berbagai fitur penting dan salah satu fitur penting tersebut adalah *Autoscaler*. Fitur ini menjalankan *Containerized Applications* secara mandiri tanpa ada campur tangan manusia. Ada tiga tipe *Autoscaler pad* pada Kubernetes dimana dua diantaranya adalah *Horizontal Pod Autoscale* dan *Vertical Pod Autoscale*. *Horizontal Pod Autoscale* (HPA) berfungsi untuk menyesuaikan jumlah *pod* dan jumlah *resource* pada *service*. Ketika jumlah beban melebihi batas *resource* yang ada, maka HPA akan membuat suatu *pod* baru dan beban tersebut akan dibagikan ke *pod-pod* yang ada. *Vertical Pod Autoscale* (VPA) bekerja dengan cara menambah dan mengurangi jumlah CPU dan *memori reservation* pada *pod* sehingga dapat meningkatkan pemanfaatan *resource* pada *cluster*. Kedua fitur pada Kubernetes inilah yang nantinya akan menjaga *resource* bekerja secara optimal[2].

Kubernetes *Cluster* akan dijalankan diatas *Cloud Provider* yaitu DigitalOcena dan Li*node.* Kedua *Cloud provider* tersebut memiliki harga yang relatif terjangkau jika dibandingkan dengan *Google Cloud Platform* (GCP).

Dalam Tugas Akhir ini, penulis akan membangun sebuah infrastruktur *high availability* dimana infrastruktur tersebut berjalan pada *cluster* Kubernetes. Infrastruktur ini dibangun menggunakan aplikasi *Video Conference* yang berbasis *open source*. Setelah itu, hasil percobaan infrastruktur akan diuji menggunakan Parameter QoS (*Quality of Service*) yang terdiri dari *throughput, delay, jitter, dan packet loss.*

1.2 Rumusan Masalah

Rumusan masalah pada Tugas Akhir ini adalah:

1. Bagaimana merancang *Video Conference services* berbasis Kubernetes pada infrastruktur *public cloud*?
2. Bagaimana performansi infrastruktur *public cloud* yang menjalankan *Video Conference services* ketika dibebani *traffic generator*?
3. Bagaimana melakukan pengukuran kinerja Kubernetes menggunakan perbandingan beberapa parameter?

1.3 Tujuan dan Manfaat

Tujuan pembuatan Tugas Akhir ini adalah

1. Merancang suatu *Video Conference service* yang akan dibangun diatas *cluster* Kubernetes pada infrastruktur *public cloud*.
2. Menguji performansi WebRTC s*ervice* pada infrastruktur yang telah dibuat.
3. Mengamati dan membandingkan hasil parameter yang didapat pada dua *application server* yang berbeda. Parameter tersebut adalah *QoS* dan *error rate.*

Manfaat dari Tugas Akhir ini adalah:

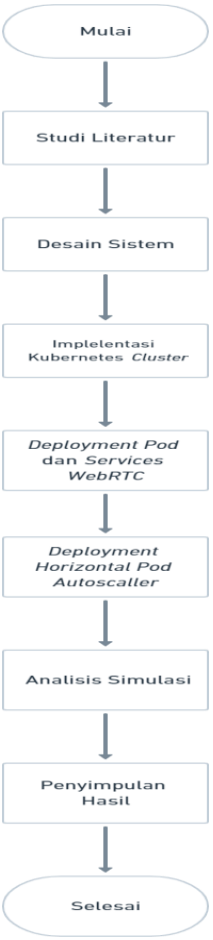
1. Memudahkan *DevOps / Administration System* mengatur jumlah *resource* dan menjaga *High Availability* layanan
2. Pengguna layanan tetap dapat mengakses layanan tersebut setiap saat tanpa khawatir adanya *downtime*

1.4 Batasan Masalah

1. Sistem Operasi yang digunakan adalah Linux Ubuntu 20.04.
2. Website *Video Conference* akan dibebani menggunakan sebuah *traffic generator* yaitu Jmeter.
3. Layanan yang digunakan adalah *service open source* WebRTC.
4. *Service* dibangun menggunakan *container Docker* dengan *Orchestration Tool* nya adalah Kubernetes.
5. Menggunakan Apache Jmeter sebagai *traffic generator* dan Wireshark sebagai *network analyzer*.
6. Infrastruktur yang digunakan adalah DigitalOcena dan Li*node*.
7. Sistem tidak menggunakan legal certificate sehingga mengabaikan jenis error *SSL handshake exception.*
4. Parameter yang akan diujikan adalah QoS (*Quality of Service*) dan Jumlah error.
5. Pengujian *video conference* hanya bisa dilakukan di satu jaringan local yang sama dikarenakan adanya masalah NAT pada WebRTC sehingga perlu setting stun atau turn server yang sudah di luar ranah *High Availability*.

6. Jenis *error* yang ditampilkan adalah *response exception* dimana jenis *error* ini diakibatkan karena *server* tidak mampu menampung lagi request yang datang.

1.5 Metode Penelitian



Gambar 1. 1. Metode Penelitian

Gambar 1.1 menjelaskan beberapa tahap metode penelitian yang dilakukan, yaitu:

1. Studi Literatur

   Mengumpulkan jurnal dan bahan bacaan sebagai sumber pendukung untuk menyelesaikan simulasi Tugas Akhir ini.
2. Desain Sistem

Pada simulasi kali ini, sistem akan didesain menjadi satu kesatuan Kubernetes *cluster* yang terdiri atas satu master *node* dan dua *worker node*. Master *node* dan *worker node* pertama dibuat menggunakan *server* DigitalOcena. Sedangkan Worker *node* kedua dibuat dengan *server* Li*node*.

3. Implementasi Kubernetes *cluster*

Pada tahap ini, Kubernetes cluster dibangun dengan cara instalasi kubeadm, kubelet, dan kubectl terlebih dahulu. Setelah instalasi berhasil dilakukan, maka dilakukan konfigurasi terhadap *master node* dan *worker node*.

4. *Deployment* WebRTC

Setelah Kubernetes *cluster* selesai dibangun, maka langkah selanjutnya adalah deployment *pod* dan service WebRTC. *Deployment pod* WebRTC dilakukan dengan membuat sebuah file yaml terlebih dahulu, sedangkan untuk melakukan deployment service, dilakukan dengan menggunakan perintah kubectl.

5. *Deployment Horizontal Pod Autoscaller*

Agar *pod* dapat bertambah/berkurang sesuai dengan kebutuhan cpu dan *memory*, maka perlu ditambahkan *Horizontal Pod Autoscaller* (HPA). Fitur ini dapat dijalankan dengan melakukan *setting* menggunakan *yaml file*.

6. Analisis Simulasi

Setelah semua tahap berhasil dilakukan, maka selanjutnya adalah melakukan pengujian pada sistem Kubernetes cluster dan pencatatan hasil simulasi yang telah dilakukan.

7. Penyimpulan Hasil

Hasil dari pengujian sistem akan disimpulkan untuk menjawab rumusan masalah.