BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Seiring dengan perkembangan teknologi pada masa kini, baik dari segi jenis layanan yang tersedia semakin variatif, juga dari segi pengguna layanan juga semakin banyak. Meningkatnya pengguna layanan mengakibatkan banyak sekali data yang perlu diolah oleh pihak penyedia layanan, sehingga perlu memiliki kemampuan pengolahan *database* yang baik, agar data yang dimiliki pelanggan tersimpan dengan baik, dan tidak terjadi kesalahan penyusunan informasi. Penggunaan *database* jenis NoSQL bisa menjadi solusi untuk masalah ini. NoSQL didesain untuk mengolah data lalu menyimpannya dengan konsisten, dengan tujuan bisa dilihat lagi dikemudian hari oleh pelanggan juga penyedia layanan. NoSQL dikembangkan untuk menyelesaikan masalah *scaling* dan *reliability*, dan sangat cocok untuk aplikasi yang berkembang dengan cepat, karena bersifat *dynamic schema.*

Semakin banyak pengguna layanan, maka *resource* dan *server* perlahan akan habis. Penyedia layanan akan kewalahan mengerjakan permintaan pelanggan, akibatnya terjadi *downtime* pada layanan. Salah satu cara untuk menghindari kasus ini, dapat digunakan Docker sebagai *container application*, yang membuat *High Availability* sehingga layanan *database* tetap terjaga. *High Availability* adalah kemampuan sistem atau *cluster* untuk menjaga layanannya tetap bekerja dengan baik. Berfungsi untuk mengurangi *error* pada layanan, ketika layanan sedang berjalan.

Penelitian sebelumnya pernah membahas tentang penggunaan Algoritma WRR pada LVS/TUN Oleh Bagus Aditya pada tahun 2015 dengan hasil bahwa kapasitas transaksi setiap detik dari *single database server* dengan dua buah *database server* yang terkluster dapat meningkat sebanyak 50%[1]. Penelitian lainnya adalah Implementasi Haproxy sebagai *Load Balancer Web Server* oleh Alam Rahmatulloh dan Firmansyah MSN yang dilakukan pada tahun 2017. Penelitian ini membuktikan jika sistem *load balancing* dapat bekerja dengan baik

ketika request datang dari *client* telah berhasil didistribusikan oleh *balancer* kepada setiap *node cluster*, sehingga *server* tidak mengalami overload[2].

Pada tugas akhir ini, penulis akan membangun sebuah *server database* MongoDB menggunakan micro Kubernetes *cluster* yang memiliki fitur *Horizonal Pod Autoscaler* yang mampu menduplikasi *pod*, sehingga dapat menjamin seluruh *request* yang masuk dapat ditanggapi oleh *server*. *Micro* Kubernetes *Cluster* juga memiliki fitur *Load Balancer* yang akan membagi *request* kepada *node* yang tergabung di *cluster*, untuk menjaga *server* tidak mengalami *downtime*[3].

1.2 Rumusan Masalah

Seperti yang sudah dijelaskan pada latar belakang, masalah pada tugas akhir ini dapat dirumuskan sebagai:

1. Bagaimana pengaruh *microcluster* Kubernetes dengan *high availability* yang tinggi untuk suatu infrastruktur database NoSQL?
2. Bagaimana perbandingan *CPU Usage* dan *Response Time* dari *service* yang dibangun dengan *microcluster* Kubernetes dengan *server* yang dibangun secara monolitik?

1.3 Tujuan Dan Manfaat

Tujuan pembuatan Tugas Akhir ini adalah

1. Membangun suatu *database* NoSQL diatas *microcluster* Kubernetes.
2. Mengamati kinerja *service* ketika dibebani oleh sebuah *traffic generator*.
3. Menguji infrastruktur yang dibagun menggunakan parameter *response time dan CPU Usage*.
4. Menganalisis perbandingan hasil pengujian antara *server* yang dibangun secara monolitik dengan dibangun dengan *microcluster* Kubernetes.

Manfaat dari Tugas Akhir ini adalah :

1. Memudahkah DevOps / *Administration System* dalam mengatur jumlah *resource* dan mejaga *High Availability* dari layanan yang digunakan.
2. Pengguna layanan bisa mengakses *service* tanpa khawatir *service* mengalami *downtime.*

1.4 Batasan Masalah

1. Sistem Operasi yang digunakan adalah Linux Ubuntu 20.04.
2. Layanan NoSQL akan dibenani menggunakan *traffic generator*.
3. Layanan DBMS yang digunakan adalah MongoDB.
4. Service yang dibangun *pad* arsitektur *microcluster* Kubernetes menggunkan *container docker* dengan *orchestration tool microcluster* Kubernetes.
5. *Traffic generator* yang digunakan adalah Apache JMeter.
6. Parameter yang akan dibandingkan adalah *Response Time dan CPU Usage*.

1.5 Metode Penelitian

Pendekatan dan metode yang digunakan dalam penyelesaian tugas akhir ini adalah :

1. Studi Literatur

   Mencari informasi terkai dengan judul penelitian dengan media jurnal dan diskusi.
2. Perancangan Sistem

   Membuat rancangan sistem dan scenario yang akan diterapakan pada penelitian.
3. Implementasi dan Pengukuran

   Mengimplementasikan rangcangan sistem yang dibuat, selanjutkan melakukan pengukuran parameter yang akan diteliti.
4. Analisa Sistem

   Menganalisis sistem yang dibuat berdasarkan pengukuran parameter yang sudah didapatkan.