BAB 1. PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan dan pengelolaan aplikasi modern berbasis *cloud*, *application scaling* dan pendekatan *DevOps* menjadi dua konsep penting yang saling mendukung untuk memastikan kinerja, keandalan, dan efisiensi infrastruktur. *Application scaling* memungkinkan sistem menyesuaikan kapasitas sumber daya secara dinamis sesuai perubahan beban kerja melalui *horizontal scaling* (menambah atau mengurangi jumlah *pod*) dan *vertical scaling* (menambah kapasitas sumber daya pada *pod* yang sama). Sementara itu, *DevOps* membantu mempercepat siklus pengembangan dengan mengoptimalkan proses *deployment*, *monitoring*, dan *scaling*, sehingga integrasi lebih cepat dan operasional lebih efisien.

Penerapan ini bertujuan untuk mengimplementasikan *Horizontal Pod Autoscaler* (HPA) pada aplikasi *website* berbasis Kubernetes, sehingga sistem dapat melakukan *autoscaling* secara otomatis. Tujuan akhirnya adalah memastikan aplikasi mampu menangani fluktuasi beban kerja tanpa terjadi *over-provisioning* yang boros sumber daya, maupun *under-provisioning* yang menurunkan performa layanan, sehingga aplikasi tetap responsif dan andal di kondisi trafik yang dinamis.

Metode yang digunakan mencakup penerapan pipeline *Continuous Integration/Continuous Deployment* (CI/CD) berbasis *DevOps* untuk otomatisasi deployment aplikasi di lingkungan Kubernetes yang menggunakan *docker* sebagai *container runtime*. HPA diatur untuk memonitor metrik sumber daya seperti penggunaan CPU dan memori pada *pod*, dengan *target average utilization* yang sudah ditentukan. Serta melakukan analisis terhadap *latency*, *throughput*, dan *error rate* selama pengujian.

Penerapan HPA pada aplikasi *website* berbasis Kubernetes ini diharapkan efektif dalam menyesuaikan jumlah *pod* sesuai beban kerja yang berubah-ubah, sehingga menjaga stabilitas layanan. Dalam skenario trafik tidak stabil seperti *event* promosi pada situs *e-commerce* atau lonjakan akses akibat berita viral pada portal berita, HPA secara otomatis menambah *pod* saat beban meningkat dan mengurangi

pod saat beban turun, menjaga *latency* tetap rendah dan meminimalkan risiko *downtime*.

Menurut penelitian yang dilakukan oleh Yusuf H. Manik dan tim dari Universitas Telkom mengenai implementasi *Horizontal Pod Autoscaler* (HPA) pada aplikasi web berbasis *Google Kubernetes Engine* (GKE), penggunaan HPA terbukti meningkatkan performa dan efisiensi sistem secara signifikan. Mereka menyatakan bahwa Cluster (Kubernetes) dengan HPA menunjukkan peningkatan jumlah transaksi hingga lima kali lipat dan waktu respons yang dua kali lebih cepat dibandingkan dengan *cluster* tanpa *autoscaling* (Manik et al., 2021). Studi ini mengimplementasikan HPA dalam arsitektur *website* untuk dapat menyesuaikan jumlah *pod* secara otomatis terhadap variasi beban kerja, sehingga menjaga stabilitas layanan serta mengoptimalkan penggunaan sumber daya yang dimiliki.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijabarkan, maka penulis mendapatkan pokok permasalahan sebagai berikut:

a. Bagaimana menerapkan *autoscaling* pada aplikasi *website* menggunakan HPA?
b. Bagaimana efektivitas penerapan HPA dalam menjaga stabilitas dan performa aplikasi saat terjadi fluktuasi beban kerja?
c. Bagaimana topologi *autoscaling* HPA dapat bekerja pada *website*?

1.3 Tujuan

Adapun tujuan dari dibentuknya laporan ini adalah:

a. Mengimplementasikan *Autoscaling* pada suatu aplikasi berbasis *cloud server* dengan Kubernetes dan Containerd
b. Memberikan solusi yang dapat meningkatkan kinerja aplikasi dengan memastikan bahwa aplikasi selalu memiliki kapasitas yang cukup untuk menangani permintaan *user*.

c. Mengevaluasi efektivitas HPA dalam menangani perubahan *traffic* serta mengintegrasikannya dengan alat pemantauan seperti Prometheus dan Grafana untuk meningkatkan observabilitas sistem.

1.4 Manfaat

Dengan terbentuknya proposal tugas akhir ini, diharapkan dapat memberi manfaat yakni sebagai berikut:

a. Skalabilitas otomatis dan efisiensi sumber daya

HPA memungkinkan sistem menyesuaikan jumlah *pod* secara dinamis berdasarkan metrik beban kerja seperti penggunaan CPU atau memori, sehingga aplikasi dapat merespons lonjakan *traffic* tanpa perlu intervensi manual.

b. Meningkatkan performa dan responsivitas saat menjalankan aplikasi

HPA bekerja dengan cara menyesuaikan jumlah replika *pod* secara otomatis berdasarkan beban kerja yang diterima, seperti penggunaan CPU atau memori. Ketika beban meningkat, HPA akan menambah jumlah *pod* agar permintaan dari pengguna dapat dilayani tanpa penurunan kinerja, sehingga aplikasi tetap responsif.

c. Mendukung penerapan DevOps dan CI/CD

Aplikasi dapat diskalakan secara otomatis berdasarkan beban kerja tanpa intervensi manual, sehingga mendukung siklus pengembangan yang cepat, rilis berkelanjutan, dan responsif terhadap perubahan kebutuhan pengguna. Hal ini memungkinkan tim DevOps untuk lebih fokus pada peningkatan kualitas kode dan pengiriman fitur baru, karena proses penskalaan infrastruktur menjadi lebih andal, efisien, dan terintegrasi dalam *workflow* CI/CD yang *modern*.

d. Pengelolaan infrastruktur yang lebih efisien

Penerapan HPA ini berguna agar sistem menambah atau mengurangi *pod* hanya ketika benar-benar diperlukan untuk menghindari pemborosan sumber daya pada saat beban rendah dan mencegah kekurangan kapasitas saat beban tinggi.

e. Meningkatkan *observability* dan *monitoring*

Membantu tim pengelola sistem untuk lebih mudah memahami, memantau, dan menganalisis kondisi aplikasi secara real time.