BAB I
PENDAHULUAN

1.1 Latar Belakang

Kubernetes saat ini merupakan platform orkestrasi kontainer yang paling luas digunakan, memungkinkan pengguna untuk mengelola aplikasi yang dikemas dalam kontainer di berbagai lingkungan seperti *cloud*, pusat data, maupun infrastruktur *hybrid* [1]. Kubernetes, yang dikenal sebagai platform orkestrasi terdepan, menyediakan beragam kemampuan seperti *auto-scaling*, *load balancing*, *self-healing*, serta *rolling update*. Keunggulan-keunggulan ini membuatnya menjadi solusi yang banyak dipilih untuk penerapan sistem kontainerisasi yang bersifat kompleks dan menangani beban kerja besar [2].

Ingress merupakan sekumpulan aturan yang digunakan untuk mengelola dan mengarahkan koneksi masuk ke dalam Kubernetes *cluster*. Aturan ini memungkinkan pengalihan permintaan dari pengguna eksternal ke *service* internal tertentu yang telah ditentukan sebagai *backend* [3]. Sementara itu, *Ingress Controller* bertugas mengelola lalu lintas dari luar *cluster* dan meneruskan permintaan ke *service* yang tepat, sesuai dengan aturan yang ditetapkan dalam konfigurasi *Ingress* [4].

Ingress menawarkan beberapa keunggulan, seperti akses melalui nama domain, kemampuan *LoadBalancer*, serta fitur *Sticky Session*. Agar *Ingress* dapat berfungsi pada *cluster* Kubernetes, diperlukan sebuah *Ingress Controller*. Contoh *Ingress Controller* yang dapat digunakan dalam Kubernetes antara lain NGINX dan Traefik [5].

Penelitian terdahulu, menunjukkan bahwa Traefik memiliki *latensi* yang lebih rendah dibandingkan NGINX pada persentil ke-75, ke-95, dan ke-99, serta mampu menangani jumlah rata-rata permintaan per detik (*throughput*) yang lebih tinggi pada skenario beban ringan [6]. Sementara penelitian lain menemukan bahwa meskipun *cluster* Traefik lebih unggul secara performa, penggunaan memorinya tercatat lebih besar dibandingkan dengan *cluster* NGINX pada skenario beban 500

client [5]. Sehingga, perbandingan kinerja pada skenario beban tinggi (*high load)* menjadi esensial. Pengujian pada level ini krusial karena perbedaan dalam efisiensi sumber daya (CPU dan memori) cenderung menunjukkan perbedaan signifikan secara statistik di semua tingkat beban yang seringkali diikuti oleh peningkatan *error rate* yang belum banyak dibahas secara mendalam dalam konteks perbandingan kedua *Ingress Controller* tersebut. Selain itu, metrik performa utama seperti latensi, dan *throughput* juga menunjukkan perbedaan signifikan yang baru muncul atau berubah seiring meningkatnya beban [4].

Berdasarkan permasalahan dan pembahasan yang telah dipaparkan, masih terdapat kesenjangan dalam pengujian dan perbandingan performa *Ingress Controller,* khususnya NGINX dan Traefik, dalam konteks beban tinggi dan lonjakan permintaan ekstrem pada Kubernetes *cluster*. Penelitian ini bertujuan untuk mengisi kesenjangan tersebut dengan melakukan analisis komparatif yang menyeluruh terhadap performa kedua *Ingress Controller* tersebut.

Untuk menjamin validitas, konsistensi, dan reliabilitas hasil, penelitian ini mengusulkan pendekatan *Infrastructure as Code* (IaC) [7]. Pendekatan ini memastikan bahwa setiap *Ingress Controller* diuji pada fondasi infrastruktur yang identik dan bersih, sehingga menciptakan perbandingan yang adil, dan hasilnya dapat direproduksi secara ilmiah.

Melalui penelitian ini, perbandingan performa *Ingress Controller* NGINX dan Traefik diharapkan dapat memberikan gambaran yang lebih jelas mengenai keunggulan dan keterbatasan masing-masing dalam menghadapi skenario beban berat. Hasil penelitian ini nantinya diharapkan dapat memberikan kontribusi dalam pemilihan *Ingress Controller* yang tepat untuk lingkungan produksi, sehingga membantu meningkatkan performa, efisiensi, dan kualitas aplikasi berbasis Kubernetes.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, rumusan masalah yang dapat disimpulkan yaitu:

1. Bagaimana performa NGINX dan Traefik sebagai *Ingress Controller* pada Kubernetes *cluster* berdasarkan parameter seperti latensi, *throughput*, dan *error rate* saat menghadapi beban tinggi?
2. Bagaimana efisiensi penggunaan sumber daya CPU dan Memori pada masing-masing *Ingress Controller* saat menghadapi beban?
3. *Ingress Controller* manakah yang mampu menghasilkan keandalan sistem yang lebih baik dalam menangani skenario beban puncak serta lonjakan permintaan yang fluktuatif?

1.3 Batasan Masalah

Agar penelitian ini tetap fokus dan terarah, terdapat beberapa batasan masalah yang diterapkan dalam pelaksanaannya. Batasan-batasan masalah ini bertujuan untuk memperjelas ruang lingkup studi serta menghindari perluasan topik di luar tujuan utama penelitian. Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini hanya membandingkan dua jenis *Ingress Controller*, yaitu NGINX dan Traefik, yang digunakan dalam Kubernetes *cluster*.
2. Penelitian dilakukan pada lingkungan AWS Elastic Kubernetes Service (EKS).
3. Penelitian ini tidak menerapkan SSL/TLS dan mTLS sebagai faktor keamanan.
4. Aplikasi *backend* yang digunakan pada penelitian adalah aplikasi *dummy* berbasis Go yang berfungsi sebagai *endpoint* bagi *Ingress Resource*.
5. Evaluasi performa difokuskan pada lima metrik utama yaitu latensi, *throughput*, *error rate*, serta penggunaan CPU dan memori.

6. Pengujian dilakukan menggunakan pendekatan *Infrastructure as Code (IaC)* dengan Terraform.
7. Pengumpulan dan analisis data dilakukan menggunakan Prometheus, Grafana, dan k6.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah membandingkan performa NGINX dan Traefik sebagai *Ingress Controller* di lingkungan AWS EKS, serta efisiensi sumber daya CPU dan memori pada skenario beban tinggi untuk menentukan pilihan yang optimal bagi kebutuhan Kubernetes *cluster*.

1.5 Manfaat Penelitian

1. Memberikan informasi empiris mengenai performa NGINX dan Traefik sebagai *Ingress Controller* dalam kondisi beban tinggi dan lonjakan permintaan ekstrem.
2. Membantu pengembang dan DevOps dalam menentukan pilihan *Ingress Controller* yang sesuai dengan kebutuhan aplikasi.
3. Mendukung perancangan arsitektur sistem Kubernetes yang lebih efisien, andal, dan skalabel.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Bab I menjelaskan latar belakang masalah, rumusan masalah, tujuan, serta manfaat penelitian. Selain itu, bab ini juga memaparkan batasan masalah untuk memperjelas ruang lingkup penelitian. Pada bagian akhir, disajikan sistematika penulisan yang memberikan gambaran umum mengenai alur penelitian yang dilakukan.

BAB II TINJAUAN PUSTAKA

Bab II membahas hasil-hasil penelitian terdahulu yang dilakukan oleh peneliti lain sebagai acuan bagi pelaksanaan penelitian ini, serta memuat landasan teori yang menjadi dasar dalam mendukung dan menjelaskan penelitian yang dilakukan.

BAB III METODE PENELITIAN

Bab III menguraikan desain penelitian, langkah-langkah pelaksanaan, serta teknik analisis yang digunakan. Fokus utama adalah implementasi Ingress Controller pada Kubernetes *Cluster* yang akan diuji melalui *load testing*.

BAB IV HASIL DAN PEMBAHASAN

Bab IV membahas tahapan pelaksanaan berdasarkan alur yang telah dirancang, mencakup proses implementasi rancangan ke dalam bentuk kode untuk memperoleh hasil penelitian.

BAB V KESIMPULAN

Bab V menyimpulkan hasil penelitian, memberikan rekomendasi tentang *Ingress Controller* yang paling efektif terhadap skenario beban tinggi dan lonjakan permintaan ekstrem, serta saran untuk penelitian lanjutan.