BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisis perbandingan performa antara NGINX *Ingress Controller* dan Traefik *Ingress Controller* pada Kubernetes *cluster* AWS EKS yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Pada kondisi beban tinggi, NGINX *Ingress Controller* unggul pada mayoritas latensi, dan kapasitas *throughput* rata-rata di 9.184 req/s yang jauh lebih besar dibandingkan Traefik *Ingress Controller* meski mengalami *error rate* sebesar 0,40%. Di sisi lain, Traefik *Ingress Controller* unggul pada latensi max dan p99 dengan jumlah *throughput* yang lebih kecil di angka 5.562 req/s, namun tidak mengalami *error rate*.
2. NGINX *Ingress Controller* unggul dalam efisiensi penggunaan sumber daya CPU dan memori dalam menghadapi beban yang jauh lebih besar dibanding Traefik *Ingress Controller*. Hal ini ditunjukkan oleh penggunaan sumber daya Traefik *Ingress Controller* yang sangat intensif untuk menghadapi beban yang jauh lebih kecil dibanding NGINX *Ingress Controller*.
3. Traefik *Ingress Controller* terbukti menjadi pilihan yang lebih stabil secara sistem, dibuktikan dengan tidak adanya *error rate*, sedangkan NGINX *Ingress Controller* menghasilkan *error rate* yaitu HTTP 502 (Bad Gateway) yang menunjukkan performa agresif tersebut membuat *backend* mengalami *overload*.

Berdasarkan kesimpulan di atas, tidak ada satu *Ingress Controller* yang unggul di semua metrik. NGINX *Ingress Controller* adalah pilihan yang superior untuk skenario yang membutuhkan efisiensi sumber daya tinggi dan volume *throughput* yang masif. Traefik *Ingress Controller* adalah pilihan yang lebih tepat untuk skenario yang memprioritaskan keandalan sistem dan performa yang dapat diprediksi pada kasus-kasus terburuk khususnya pada sistem yang kritis dengan pertimbangan sumber daya.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa hal yang dapat menjadi saran untuk pengembangan penelitian selanjutnya:

1. Penelitian selanjutnya disarankan untuk menambah jumlah *Ingress Controller* yang dibandingkan, seperti HAProxy, dan Envoy, ataupun *Service Mesh Gateway* seperti Istio dan Consul.
2. Penelitian ini hanya berfokus pada performa. Aspek efisiensi biaya dapat diteliti lebih lanjut. Mengingat NGINX menggunakan sumber daya yang jauh lebih sedikit, penelitian selanjutnya dapat mengkalkulasi perbandingan biaya operasional secara langsung antara kedua *Ingress Controller* tersebut pada lingkungan *cloud*.
3. Penelitian selanjutnya dapat menguji performa *Ingress Controller* dalam hubungannya dengan mekanisme *auto-scaling* pada Kubernetes, seperti *Horizontal Pod Autoscaler* (HPA), untuk menganalisis bagaimana latensi, *throughput*, dan *utilization* CPU serta memori yang paling efektif digunakan sebagai HPA untuk menjaga keandalan sistem dan ketersediaan selama beban fluktuatif.