BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada beberapa tahun terakhir, banyak perusahaan di seluruh dunia beralih dari sistem *monolithic* ke *microservice* (Capuano & Muccini, 2022). Ini didukung oleh data dari jrebel.com pada 2021 yang menunjukkan bahwa 49% responden menggunakan arsitektur *microservice* dalam aplikasi utama mereka. Dari data tersebut, 36% responden baru-baru ini bermigrasi ke arsitektur *microservice*, 30% sudah menggunakan aplikasi berbasis *microservice*, 21% masih dalam tahap pembicaraan, dan 13% tidak berencana untuk beralih ke arsitektur berbasis *microservice* (Johnson, 2021). Fenomena ini muncul karena percepatan digitalisasi yang mengharuskan pemenuhan kebutuhan lebih cepat. Keuntungan arsitektur *microservice* meliputi kemampuan peningkatan skala dan ketahanan sistem melalui proses *deployment* serta dapat memadupadankan teknologi.

Arsitektur *microservice* memungkinkan pengerjaan secara paralel, serta memungkinkan perekrutan lebih banyak *developers* tanpa hambatan. Ini memudahkan pemahaman tugas masing-masing *developer* karena fokus pada satu bagian. Isolasi juga mendukung variasi teknologi seperti penggabungan bahasa pemrograman, gaya pemrograman, *platform deployment*, dan basis data (Newman, 2019). Meski memiliki keunggulan, arsitektur *microservice* juga menunjukkan kelemahan seperti kompleksitas jaringan yang lebih tinggi dan potensi *overhead* pada basis data dan *server*. Sehingga membutuhkan pemahaman yang mendalam tentang *containerisasi*, pengaturan *container*, dan *deployment*.

Pada 2017, sebanyak 78,66% responden memilih Docker sebagai teknologi *container* yang paling banyak digunakan, diikuti oleh rkt (1,67%) dan Mesos Containerizer (2,93%) (Vailshery, 2017). Teknologi *container* memfasilitasi pembuatan dan *deployment* cepat berbagai layanan aplikasi. Namun dibeberapa penelitian, Docker *container* tidak begitu efektif untuk mengelola *container* dalam jumlah besar dan memiliki kekurangan pada sistem keamanannya (Preeth E N dkk., 2015) (Arango dkk., 2017). Dikarenakan keterbatasan dan untuk mengatasi

beberapa kompleksitas operasional, Kubernetes muncul sebagai solusi dalam konteks *containerisasi*. Kubernetes mampu mengelola Docker *Container* dan *workload*, serta mempermudah operasi saat melibatkan banyak *container* di berbagai *server*.

Dikembangkan oleh Google pada tahun 2014, Kubernetes kini tersedia pada layanan terkelola di berbagai Cloud Service Provider (CSPs) seperti Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), dan Microsoft Azure Kubernetes Service (AKS). Survey dari Cloud Native Computing Foundation (CNCF) pada 2022 juga mencatat bahwa 96% responden mengadopsi atau mengevaluasi Kubernetes, dengan 79% nya menggunakan layanan terkelola termasuk GKE, EKS, dan AKS (Doerrfeld, 2023). Untuk memilih *platform* yang akan diterapkan, dibutuhkanlah sebuah perbandingan dari uji performa. Seperti penelitian yang dilakukan oleh (Nugroho, 2018) dan (Pereira Ferreira & Sinnott, 2019) yang mengevaluasi karakteristik *container* untuk menganalisis *overhead*, menguji *availability* dan *request* dari *container*.

Berdasarkan informasi sebelumnya, dibutuhkanlah suatu pengujian yang berbeda dari penelitian sebelumnya sebagai informasi lebih dan penguat argumen. Pengujian dilakukan secara *real time* dengan parameter CPU, *memory* dan *network* yang diuji pada suatu *container* Kubernetes di *platform* Google Cloud Platform (GCP), Amazon Web Service (AWS), dan Microsoft Azure.

1.2 Rumusan Dan Batasan Masalah

1.2.1 Rumusan Masalah

Berdasarkan uraian tersebut maka dapat disimpulkan beberapa permasalahan sebagai berikut:

1. Bagaimana analisis penggunaan Kubernetes *service* pada *platform* GCP, AWS dan Azure dalam membangun suatu *container*?
2. Bagaimana perbandingan performa Kubernetes *service* pada *platform* GCP, AWS dan Azure berdasarkan beberapa parameter pengujian seperti CPU, *memory* dan *network* yang diuji didalam *container*?

1.2.2 Batasan Masalah

Dalam penelitian ini terbatas pada pembahasan yaitu sebagai berikut:

1. *Container* berisi aplikasi nginx versi 1.23.
2. Penelitian ini berfokus pada perbandingan performa *container* antar CSPs termasuk CPU, *Memory*, dan *Network.*
3. Ketiga CSPs menggunakan versi Kubernetes 1.27.
4. Ketiga CSPs memakai 2 *nodes.*
5. Ketiga CSPs memakai *instance machine* dengan 4 vcpu dan 16 GiB *memory.*
6. Analisa performa pada saat *deployment.*
7. Tidak membahas arsitektur *monolithic* ataupun *microservice.*

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah disusun maka tujuan penelitian ini adalah:

1. Melakukan analisis penggunaan Kubernetes *service* pada *platform* GCP, AWS dan Azure dalam membangun suatu *container.*
2. Membandingkan performa Kubernetes *service* pada *platform* GCP, AWS dan Azure berdasarkan beberapa parameter pengujian seperti CPU, *memory* dan *network* yang diuji didalam *container.*

1.4 Manfaat Penelitian

1.4.1 Manfaat Teoritis

Penelitian ini dapat menambah pengetahuan teori mengenai, *Cloud Computing*, teknologi *container*, dan Kubernetes *Service* di *Platform* GCP, AWS, dan Azure.

1.4.2 Manfaat Praktis

1.4.2.1 Bagi Penulis

1) Dapat menambah pengetahuan di bidang *Cloud Computing*, *container*, dan Kubernetes.
2) Melatih penulis secara *technical* dalam mengoperasikan, me-*monitoring* dan menganalisis beberapa *tools* yang digunakan dalam penelitian ini.

1.4.2.2 Bagi Pengembangan Ilmu

1) Mengetahui metode serta *tools* yang digunakan dalam uji performa *container* pada Kubernetes *Service* di *platform* GCP, AWS, dan Azure.
2) Menjadi referensi mahasiswa/i yang ingin melanjutkan penelitian serupa.

1.4.2.3 Bagi Perusahaan

1) Bahan pertimbangan untuk memilih *platform* terbaik untuk menjalankan *container* diatas Kubernetes *Service.*

1.5 Sistematika Penulisan Skripsi

Untuk mempermudah melihat dan mengetahui pembahasan yang ada pada skripsi ini secara menyeluruh, maka diperlukan mengemukakan sistematika yang ada pada skripsi ini secara menyeluruh, maka diperlukan mengemukakan sistematika yang merupakan kerangka dan pedoman penulisan skripsi. Adapun sistematika penulisannya sebagai berikut:

1.5.1 Bagian Awal Skripsi

Bagian awal memuat halaman sampul depan, lembar pengesahan, lembar pernyataan, kata pengantar, ucapan terimakasih, abstrak, *abstract*, daftar isi, daftar gambar, daftar tabel, daftar lampiran dan daftar istilah.

1.5.2 Bagian Utama Skripsi

Bagian utama terbagi atas bab dan sub bab yaitu sebagai berikut:

BAB I PENDAHULUAN

Bab ini terdiri dari latar belakang, rumusan masalah, tujuan dan manfaat.

BAB II TINJAUAN PUSTAKA

Bab ini terdiri dari landasan teori yang berisi tentang pembahasan *Cloud Computing, Container*, Docker, Kubernetes, dan Cloud Service Provider (CSPs).

BAB III METODE PENELITIAN

Bab ini mengemukakan metode penelitian yang dilakukan penulis.

BAB IV HASIL DAN PEMBAHASAN

Bab ini terdiri dari gambaran hasil penelitian dan analisa serta pembahasan hasil penelitian.

BAB V PENUTUP

Bab ini berisi kesimpulan, implikasi dan rekomendasi dari seluruh penelitian yang telah dilakukan.

1.5.3 Bagian Akhir Skripsi

Bagian akhir skripsi berisi daftar pustaka, lampiran, dan daftar riwayat hidup penulis.