BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian M. Agung Nugroho & Cuk Subiyantoro tahun 2018, terjadi peningkatan *availability* layanan dengan rancangan Kubernetes tidak ditemukan persentase failure request, dari tes simulasi pengiriman request dengan jumlah beban akses 100 *user*, 400 *user*, 800 *user*, 1600 *user*, 3200 *user*, 6400 *user*, 12800 *user*, dan 25600 *user*. Dan rata-rata diselesaikan 100% selama *range* waktu 1 – 4 detik. *Cluster* kubernetes dengan optimalisasi dapat mengurangi persentase *request failure* dan meningkatkan availability layanan. Dimana saran penyedia layanan dapat melakukan analisis performa berdasarkan resiko request yang dapat mereka tangani, sebagai contoh, jika penyedia layanan ingin dapat memberikan layanan dengan jumlah maksimal *user* 25600 dengan waktu dibawah 5 s, maka target tersebut telah tercapai. Namun jika penyedia layanan perlu memberikan layanan ketercapaian adalah 1 s, sekalipun jumlah akses *user* maksimal, maka diperlukan beberapa optimalisasi dari sisi layanan.

Kemudian penelitian Ali Akbar Khatami tahun 2020, sistem high availability dengan menggunakan Kubernetes cluster berhasil dengan *uptime* maupun *downtime instance* ketika menjalankan Kubernetes, yaitu dari 2160 menit Kubernetes cluster menyala, uptime yang didapatkan sebesar 2160 menit, atau

100%. Terpenuhinya *high availability* Kubernetes karena meskipun ada *instance* yang *down*, layanan yang disimpan di cluster masih dapat diakses melalui *instance* lain. Penggunaan hardware harus menggunakan hardware dengan kecepatan clock yang mumpuni agar Kubernetes cluster dapat berjalan dengan optimal, Penggunaan *instance* dengan RAM minimal 4GB untuk membuat Kubernetes *cluster* agar terdapat banyak space untuk *processing*, Menggunakan IP *public* agar Kubernetes cluster dapat diakses secara global dan menggunakan sertifikat SSL resmi untuk etcd cluster maupun Kubernetes cluster.

Lalu penelitian Zulfikar & Adhitya Bhawiyuga & Achmad Basuki tahun 2022 berhasil menerapkan sistem lab virtual berbasis multi klaster. Hal tersebut dilakukan dengan membuat satu klaster orkestrator dan tiga buah klaster universitas. Perlu dilakukan penelitian lebih dalam terkait metode alternatif untuk mengakses aplikasi dalam kontainer seperti menggunakan ingress atau melalui VPN Gateway untuk memungkinkan mengakses Service Cluster IP secara langsung. Selain itu, dapat dilakukan penelitian terkait integrasi komputasi tepi dengan aplikasi dalam klaster yang telah dibuat.

Kemudian penelitian Azwar Riza Habibi & Galih Laksono tahun 2020 implementasi Ngorder API pada sistem microservice dapat dikatakan bahwa dengan menggunakan Katalon Studio dapat dikatakan bahwa layanan API Gateway Amazon Web Service dapat mendistribusikan trafik ke dalam cluster Kubernetes dengan baik sesuai dengan kategori jalur yang ditentukan.

Lalu penelitian Lilik Widyawati & Heroe Santoso & Hamdika Budiman tahun 2021 Kubernetes dapat digunakan untuk melakukan container pada web berbasis Nginx dan FTP server sehingga dapat terjaga ketersediaannya berdasarkan skenario uji coba mengaktifkan seluruh node, menonaktifkan salah satu node dan menonaktifkan seluruh node pembentuk deploy. Dimana saran berupa dilakukan perbandingan dengan penelitian sejenis.

Penelitian kali ini memiliki kesamaan dengan pustaka diatas yaitu sama-sama menggunakan kubernetes dan docker dalam membangun infrastruktur, serta penerapan beberapa saran yang diperoleh dari pustaka-pustaka diatas, seperti penggunaan instance dengan RAM minimal 4GB untuk membuat Kubernetes *cluster* agar terdapat banyak space untuk *processing*, menggunakan IP *public* agar Kubernetes cluster dapat diakses secara global serta implementasi Ingress pada Kubernetes.

Berikut merupakan daftar pustaka yang digunakan sebagai acuan serta referensi dari penelitian ini yang terdapat pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

Penulis (Tahun)	Topik Penelitian	Teknologi	Hasil
M. Agung Nugroho & Cuk Subiyantoro (2018)	Analisis Cluster Container	Docker, Kubernetes, Locust Load Testing	Cluster kubernetes dengan optimalisasi dapat mengurangi persentase request failure dan meningkatkan availability layanan
Ali Akbar Khatami (2020)	Implementasi High Availability Storage Server	Docker, Kubernetes, Rook Cloud Native, Cepth	Sistem high availability dengan menggunakan Kubernetes cluster berhasil memenuhi kriteria high availability dengan tingkat availability sebesar 100%
Mohammad Zulfikar & Adhitya Bhawiyuga & Achmad Basuki (2022)	Implementasi Lab Virtual	Kubernetes, Docker	Berhasil menerapkan sistem lab virtual berbasis multi klaster
Azwar Riza Habibi & Galih Laksono (2020)	API Service	Kubernetes, Docker, Terraform	Dapat mendistribusikan trafik ke dalam cluster Kubernetes dengan baik sesuai dengan kategori jalur yang ditentukan dan fungsi logika pada script Ngorder API tidak mengalami kendala saat ditransfer ke infrastruktur dengan sistem microservice
Lilik Widyawati & Heroe Santoso & Hamdika Budiman (2021)	Analisa Penerapan Server Deployment	Kubernetes, Docker, FTP, Web Server NGINX	Kubernetes dapat digunakan untuk melakukan container pada web berbasis Nginx dan Ftp server sehingga dapat terjaga ketersediaannya

			berdasarkan skenario uji coba
Ragil Murdiantoro Aji (2023)	Infrastruktur Aplikasi Digital Signature Documenso	Kubernetes, Cloud Computing, Docker, Digital Signature	Aplikasi Documenso di dalam cluster Kubernetes berhasil diimplementasikan dan proses auto scaling pada pod dapat berjalan secara normal, kemudian hasil pengujian tertinggi sebanyak 35000 request di halaman login mendapatkan rata-rata waktu respon selama 399.64 milisecond, sedangkan di fitur ubah nama pengguna mendapatkan 50.42 milisecond

2.2 Dasar Teori

2.2.1 Container

Container memiliki konsep yang hampir sama dengan konsep virtualisasi hanya saja container lebih ringan jika dibandingkan dengan virtualisasi seperti sedikit memakan resource dan waktu. Ada 3 ide yang paling mendasar dari sebuah container. Ide tersebut menjelaskan bahwa container merupakan sebuah runtime portable yang ringan, kontainer memiliki kemampuan untuk mengembangkan, menguji dan menyebarkan aplikasi ke sejumlah beser server dan container berkemampuan untuk menghubungkan antar kontainer (Calus Pahl, 2015).

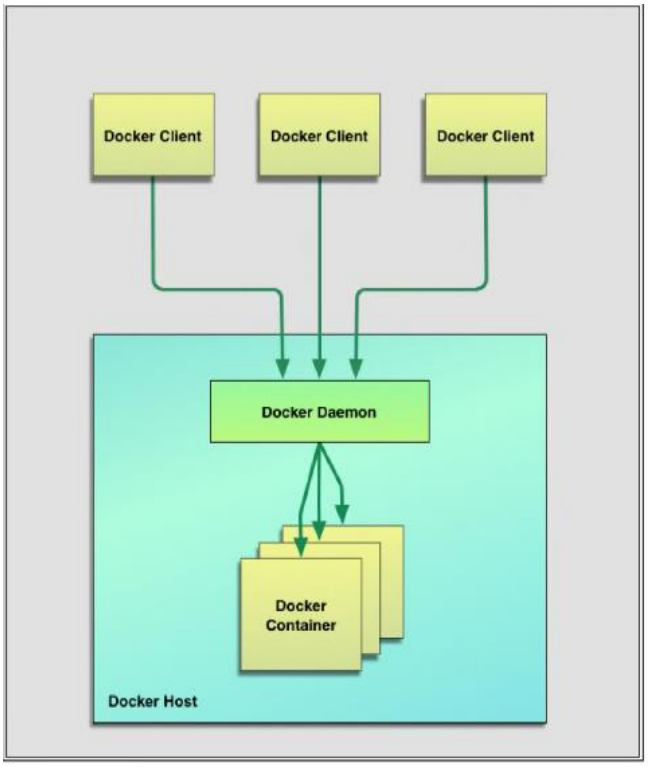
Tidak seperti hypervisor pada virtualisasi, dimana satu atau lebih mesin independen berjalan secara virtual pada hardware fisik melalui lapisan perantara, container berjalan di ruang pengguna diatas kernel sistem operasi. Hasilnya, kontainer sering disebut virtualisasi tingkat sistem operasi. Teknologi container memungkinkan beberapa instance di ruang pengguna yang terisolasi dijalankan pada satu host. Karena statusnya sebagai guest dari sistem operasi, container terkadang dianggap kurang fleksibel; mereka umumnya hanya dapat menjalankan guest sistem operasi yang sama atau serupa dengan host yang mendasarinya (James Turnbull, 2019).

2.2.2 Docker

Menurut James Turnbull, Docker adalah sebuah perangkat lunak open source yang mengotomasi deployment dari aplikasi ke dalam kontainer. Docker ditulis oleh tim Docker, Inc, dan telah dirilis dibawah lisensi Apache 2.0. Docker menambahkan engine deployment aplikasi diatas virtual lingkungan eksekusi kontainer ter virtualisasi. Hal ini dirancang untuk menyediakan lingkungan yang ringan dan cepat untuk menjalankan kode aplikasi serta alur kerja yang efisien untuk memasukkan kode tersebut dari perangkat pribadi seperti laptop ke lingkungan pengujian dan kemudian ke dalam production. Ada beberapa komponen didalam docker diantaranya:

1. Docker Engine

Docker merupakan aplikasi berbasis client-server. Docker client berbicara dengan docker server atau daemon yang pada gilirannya melakukan semua pekerjaan. Docker dikirimkan dengan paket biner yang dijalankan secara command line, serta docker menyediakan RESTfull API untuk berinteraksi dengan daemon docker. Docker juga bisa menjalankan daemon dan client secara bersamaan di satu host yang sama atau dengan host yang berbeda.



Gambar 2.1 Arsitektur Client-Server Pada Docker

2. Docker Imange

Docker image merupakan building blocks dari container docker itu sendiri. Container dijalankan dari docker images. Docker image merupakan “build” bagian dari life cycle Docker.

3. Docker Registry

   Docker registries digunakan untuk menyimpan images yang telah dibuild. Ada dua tipe registries: public dan private. Docker menyediakan registies secara terbuka yang dikenal sebagai Docker Hub. Docker Hub juga menyediakan penyimpanan image bersifat publik atau private.

4. Docker Container

   Container Docker dapat mengemas aplikasi atau service yang akan dijalankan ke dalam sebuah container. Container diluncurkan dari images dan dapat berisi satu atau lebih proses yang sedang berjalan. Analoginya seperti images merupakan sebuah aspek pembuatan atau pengemasan dan container sebagai aspek pengoprasian atau eksekusi.

   Docker container adalah:

   - Sebuah format image
   - Sebuah set dari standar operasi
   - Sebuah environment execution

Docker menggunakan konsep standar shipping container, yang digunakan sebagai transportasi untuk mengangkut barang secara global, sebagai model kontainernya. Setiap kontainer berisikan images software yang memungkinkan serangkaian operasi dilakukan, seperti dapat dibuat, dimulai, dihentikan, di restart, dan di destroy. Docker container tidak peduli terhadap konten yang ada di dalamnya, setiap kontainer memiliki sifat yang sama.

2.2.3 Google Cloud Platform (GCP)

Menurut JJ Geewax, cloud adalah lapisan abstraksi dalam infrastruktur komputer, tempat komputasi, penyimpanan, analitik, jaringan, dan banyak lagi semuanya didorong ke tingkat yang lebih tinggi dalam stack komputasi. Struktur ini mengalihkan fokus pengembang dari CPU dan RAM dan menuju API untuk operasi tingkat yang lebih tinggi seperti menyimpan atau menanyakan data. Layanan cloud sangat fleksibel dan sebagian besar tidak memerlukan penyediaan atau kontrak jangka panjang. mengandalkan layanan ini memungkinkan meningkatkan dan menurunkan skala tanpa pemberitahuan atau penyediaan terlebih dahulu, sambil membayar sumber daya yang digunakan pada bulan tertentu.

Google Cloud Platform (GCP) adalah kumpulan produk yang memungkinkan pengguna menggunakan beberapa infrastruktur internal Google. Koleksi ini mencakup banyak hal yang umum di semua penyedia cloud cloud, seperti mesin virtual on-demand melalui Google Compute Engine atau penyimpanan objek untuk menyimpan file melalui Google Cloud Storage. Ini juga mencakup API untuk beberapa teknologi canggih buatan Google, seperti Bigtable, Cloud Datastore, atau kubernetes. Google beroperasi pada skala sedemikian rupa sehingga memiliki banyak keuntungan ekonomi, yang didapatkan dalam bentuk harga yang lebih rendah.

Google Cloud terdiri dari sekumpulan aset fisik, seperti komputer dan hard disk drive, serta sumber daya virtual, seperti mesin virtual (VM), yang terdapat di data center Google di seluruh dunia. Setiap lokasi pusat data berada di suatu wilayah. Wilayah tersedia di Asia, Australia, Eropa, Amerika Utara, dan Amerika Selatan. Setiap wilayah merupakan kumpulan zona-zona yang terisolasi satu sama lain di dalam wilayah tersebut. Setiap zona diidentifikasi dengan nama yang menggabungkan huruf pengenal dengan nama wilayah. Misalnya zona a di kawasan Asia Timur diberi nama asia-east1-a. Distribusi resource ini memberikan beberapa manfaat, termasuk redundansi jika terjadi kegagalan dan mengurangi latensi dengan menempatkan sumber daya lebih dekat dengan klien. Distribusi ini juga memperkenalkan beberapa aturan tentang bagaimana sumber daya dapat digunakan secara bersama-sama.

2.2.4 Kubernetes

Kubernetes adalah sistem orkestrasi container open source. Kubernetes mengelola aplikasi dalam container di beberapa host untuk penerapan, pemantauan, dan penskalaan container. Awalnya dibuat oleh Google, pada bulan Maret 2016 disumbangkan ke Cloud Native Computing Foundation (CNCF).

Kubernetes, atau disingkat “k8s” atau “kube”, memungkinkan pengguna untuk mendeklarasikan aplikasi yang diinginkan menggunakan konsep seperti “deployment” dan “services.”. Kubernetes memulai dan kemudian terus memantau container, menyediakan restart otomatis, penjadwalan ulang, dan

replikasi untuk memastikan aplikasi tetap dalam kondisi yang diinginkan. Kubernetes tersedia sebagai instalasi mandiri atau dalam berbagai distribusi, seperti Red Hat OpenShift, Pivotal Container Service, CoreOS Tectonic, dan Canonical Kubernetes.

Resource Kubernetes dapat dibuat dengan baris perintah tetapi biasanya ditentukan menggunakan file Yet Another Markup Language (YAML). Resource yang tersedia dan kolom untuk setiap resource dapat berubah dengan versi Kubernetes yang baru. Di kubernetes ada beberapa istilah seperti:

a. Pod

   Pod merupakan sekelompok kontainer yang terdiri dari satu atau lebih. Kubernetes akan menjadwalkan semua container untuk sebuah pod ke dalam host yang sama, dengan namespace jaringan yang sama, sehingga semuanya memiliki alamat IP yang sama dan dapat saling mengakses menggunakan localhost.

b. Deployment

   Deployment merupakan penskalaan pod dan pembaharuan secara rolling. Kubernetes akan memastikan bahwa sejumlah pod yang ditentukan sedang berjalan, dan pada pembaruan berkelanjutan, Kubernetes akan mengganti instance pod satu per satu, sehingga memungkinkan pembaruan aplikasi tanpa downtime. Deployment menggantikan konsep Replica Set yang lama.

Deployment membuat kumpulan replika, namun tidak perlu berinteraksi dengan kumpulan replika secara langsung.

c. Service

Service merupakan Load Balancing pada Deployment. Dalam penerapan, setiap pod diberi alamat IP yang unik, dan ketika sebuah pod diganti, pod baru biasanya menerima alamat IP baru. Dengan mendeklarasikan service, dapat menyediakan satu titik masuk untuk semua pod dalam penerapannya.

d. Persistent Volume Claim

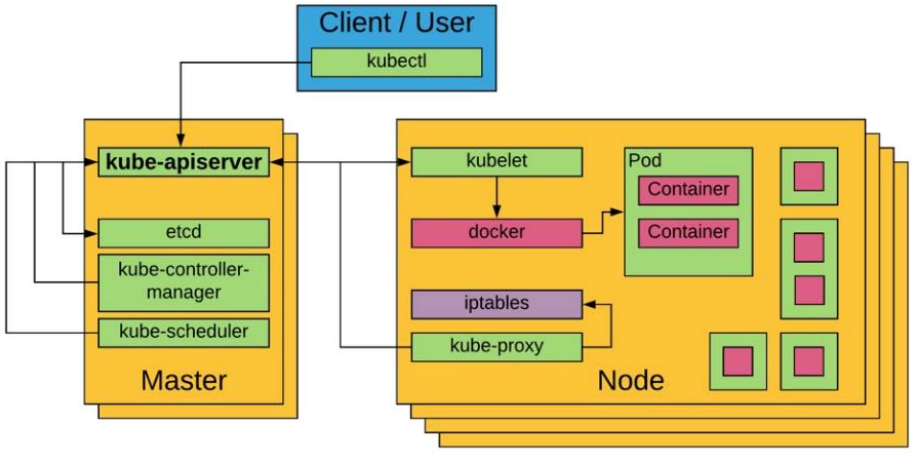
Kubernetes memiliki beberapa jenis resource penyimpanan. Persistent Volume Claim meminta Kubernetes untuk mengalokasikan penyimpanan secara dinamis dari Kelas Penyimpanan. Kelas Penyimpanan biasanya dibuat oleh administrator cluster Kubernetes dan harus sudah ada. Setelah Klaim Volume Persisten dibuat, Klaim Volume Persisten dapat diberikan ke sebuah Pod. Kubernetes akan menyimpan penyimpanan tersebut selama Persistent Volume Claim masih ada, meskipun Pod yang terpasang telah dihapus.

e. Ingress

Ingress mengexpose rute HTTP dan HTTPS dari luar cluster ke layanan di dalam cluster. Perutean lalu lintas dikontrol oleh aturan yang ditentukan pada sumber daya Ingress. Ingress dapat dikonfigurasi untuk memberikan externally-reachable URLs, load balance traffic, terminate SSL / TLS, dan offer name-based virtual hosting. Ingress controler bertanggung jawab untuk memenuhi Ingress, biasanya dengan load balancer, meskipun pengontrol

tersebut juga dapat mengonfigurasi router edge atau frontend tambahan untuk membantu menangani lalu lintas. Ingress tidak mengekspos port atau protokol yang sewenang-wenang.

Cluster Kubernetes adalah sekumpulan mesin fisik atau virtual dan sumber daya infrastruktur yang digunakan untuk menjalankan aplikasi. Mesin yang mengelola cluster disebut Master, dan mesin yang menjalankan container disebut Node. Kemudian Kubernetes menggunakan arsitektur client-server, seperti yang terlihat dibawah ini:



Gambar 2.2 Arsitektur Kubernetes

Master menjalankan service yang mengelola cluster. Yang paling penting adalah kube-apiserver, yang merupakan layanan utama yang digunakan klien dan node untuk menanyakan dan memodifikasi resource yang berjalan di cluster. Server API dibantu oleh: etcd, penyimpanan nilai kunci terdistribusi yang digunakan untuk mencatat status cluster; kube-controller-manager, sebuah

program pemantauan yang memutuskan perubahan apa yang harus dilakukan ketika sumber daya ditambahkan, diubah, atau dihapus; dan kube-scheduler, sebuah program yang memutuskan di mana pod akan dijalankan berdasarkan node yang tersedia dan konfigurasinya dalam instalasi Kubernetes dengan ketersediaan tinggi, akan terdapat beberapa master, dengan satu master bertindak sebagai master utama dan master lainnya sebagai replika.

Node adalah mesin fisik atau virtual dengan layanan yang diperlukan untuk menjalankan container. Sebuah cluster Kubernetes harus memiliki node sebanyak yang diperlukan untuk semua pod yang diperlukan. Setiap node memiliki dua layanan Kubernetes: kubelet, yang menerima perintah untuk menjalankan container dan menggunakan mesin container (misalnya Docker) untuk menjalankannya; dan kube-proxy, yang mengelola aturan jaringan sehingga koneksi ke alamat IP layanan di rutekan dengan benar ke pod. Seperti yang ditunjukkan pada Gambar 2.2, setiap node dapat menjalankan beberapa pod, dan setiap pod dapat menyertakan satu atau lebih container. Pod ini murni merupakan konsep Kubernetes; kubelet mengonfigurasi mesin container untuk menempatkan beberapa container di namespace jaringan yang sama sehingga container tersebut berbagi alamat IP.

Kubernetes menyediakan fitur-fitur yang dapat digunakan sebagai pendukung lifecycle pada deployment, diantaranya:

a. Automated rollouts and rollbacks

   Kubernetes secara bertahap meluncurkan perubahan pada aplikasi atau konfigurasinya, sambil memantau kesehatan aplikasi untuk memastikan perubahan tersebut tidak mematikan semua instance pada saat yang bersamaan. Jika terjadi kesalahan, Kubernetes akan mengembalikan perubahan tersebut.

b. Service discovery and load balancing

   Tidak perlu memodifikasi aplikasi untuk menggunakan mekanisme service discovery yang tidak dikenal. Kubernetes memberikan Pod alamat IP mereka sendiri dan satu nama DNS untuk sekumpulan Pod, dan dapat melakukan penyeimbangan muatan di seluruh Pod tersebut.

c. Storage orchestration

   Pemasangan sistem penyimpanan pilihan secara otomatis, baik dari penyimpanan lokal, penyedia cloud publik, atau sistem penyimpanan jaringan seperti iSCSI atau NFS.

d. Self-healing

   Memulai ulang kontainer yang gagal, mengganti dan menjadwal ulang kontainer ketika node mati, mematikan kontainer yang tidak merespons pemeriksaan kesehatan yang ditentukan pengguna, dan tidak mengiklankannya ke client hingga siap dirilis.

e. Secret and configuration management

Penerapan dan pembaharuan konfigurasi aplikasi yang tersentralisasi, tanpa membangun kembali image dan tanpa mengekspos konfigurasi.

f. Automatic bin packing

Secara otomatis menempatkan kontainer berdasarkan kebutuhan sumber daya dan batasan lainnya, tanpa mengorbankan ketersediaan.

g. Batch execution

Selain service, Kubernetes dapat mengelola beban kerja batch dan CI, juga dapat mengganti container yang gagal, jika diinginkan.

h. Horizontal scaling

Skalakan aplikasi ke atas atau ke bawah dengan perintah sederhana, dengan UI, atau secara otomatis berdasarkan penggunaan CPU.

i. IPv4/IPv6 dual-stack

Alokasi alamat IPv4 dan IPv6 ke Pod dan Layanan.

j. Designed for extensibility

Menambahkan fitur ke cluster Kubernetes tanpa mengubah source code upstream.

2.2.5 Google Kubernetes Engine (GKE)

Menurut dokumentasi Google Cloud Platform, Google Kubernetes Engine (GKE) merupakan layanan Kubernetes terkelola yang dapat digunakan untuk men-deploy dan mengoperasikan aplikasi dalam container dalam skala besar menggunakan infrastruktur Google. GKE adalah implementasi platform orkestrasi

container open source Kubernetes yang dikelola Google. GKE sangat ideal jika memerlukan platform yang memungkinkan konfigurasi infrastruktur yang menjalankan aplikasi dalam container, seperti jaringan, penskalaan, hardware, dan keamanan. GKE memberikan kekuatan operasional Kubernetes sekaligus mengelola banyak komponen dasar, seperti control plan dan node.

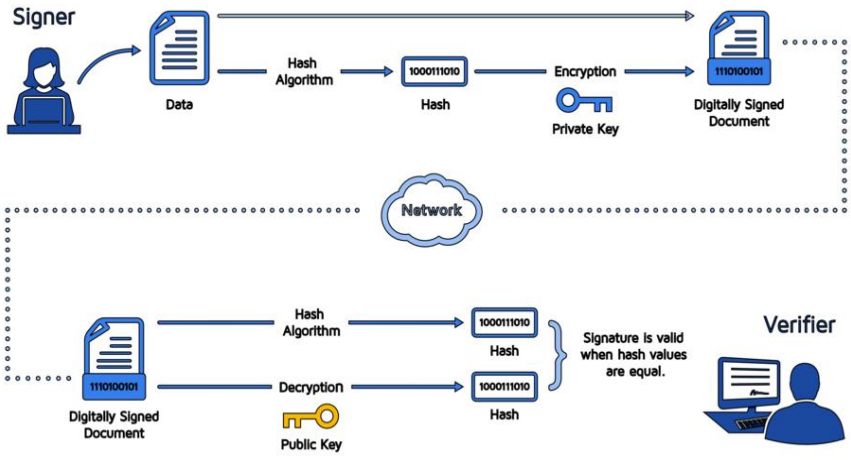
2.2.6 Digital Signature

Menurut Julius Indra Dwiparyo, tanda digital signature adalah sebuah identitas elektronik yang berfungsi sebagai tanda persetujuan terhadap kewajiban-kewajiban yang melekat pada sebuah akta elektronik. Sedangkan menurut UU ITE Pasal 1 angka 2, digital signature adalah sebuah pengaman pada data digital yang dibuat dengan kunci tanda tangan pribadi (private signature key), yang penggunaannya tergantung pada kunci publik (public key) yang menjadi pasangannya. Eksistensi digital signature ini ditandai oleh keluarnya sebuah sertifikat kunci tanda tangan (signature key certificate) dari suatu badan pembuat sertifikat (certifier). Dalam sertifikat ini ditentukan nama pemilik kunci tanda tangan dan karakter dari data yang sudah ditandatangani, untuk kekuatan pembuktian (German Draft Digital signature Law, 1996).

Ketika penandatangan secara elektronik menandatangani dokumen, tanda tangan dibuat menggunakan kunci pribadi penandatangan, yang selalu disimpan dengan aman oleh penandatangan. Algoritma matematika bertindak seperti sandi, membuat data yang cocok dengan dokumen yang ditandatangani, disebut hash,

dan mengenkripsi data. Data terenkripsi yang dihasilkan adalah tanda tangan digital. Tanda tangan juga ditandai dengan waktu dokumen ditandatangani. Jika dokumen berubah setelah penandatanganan, tanda tangan digital dianggap tidak valid.

Sebagai contoh, Jane menandatangani perjanjian menggunakan kunci pribadinya. Penerima menerima dokumen. Penerima yang lain menerima dokumen dan juga menerima salinan kunci publik Jane. Jika kunci publik tidak dapat mendekripsi tanda tangan (melalui sandi dari mana kunci dibuat), itu berarti tanda tangan itu bukan milik Jane, atau telah diubah sejak ditandatangani. Tanda tangan kemudian dianggap tidak valid.



Gambar 2.3 Alur Kerja Digital Signature

2.2.7 Documenso

Salah satu tool open source digital signature yang dapat di hosting secara mandiri yaitu Documenso. Documenso sendiri merupakan tool digital signature alternatif dari Docusign yang di bangun oleh Timur Ercan beserta komunitas, source code dapat diakses di publik repository Github. Documenso dirilis di awal tahun 2023 dengan versi 0.1 dengan stack yang dipakai seperti Typescript, Next.js, Prisma, Tailwind, shadcn/ui, NextAuth.js, react-email, tRPC, Node SignPDF, React-PDF, React-PDF, PDF-Lib, Stripe, dan Vercel. Documenso memiliki fitur yang terbilang cukup sederhana dalam penggunaanya seperti sign doc, send & receive document with email, sign email notification, dan documents summary.

2.2.8 PostgreSQL

Menurut dokumentasi PostgreSQL, PostgreSQL adalah Object-Relational Database Management System (ORDBMS) yang telah di develop dalam berbagai bentuk semenjak tahun 1977. PostgreSQL adalah sistem database relasional yang open source yang kuat yang menggunakan dan memperluas bahasa SQL yang dikombinasikan dengan banyak fitur yang menyimpan dan menskalakan beban kerja data paling rumit dengan aman. Asal usul PostgreSQL dimulai pada tahun 1986 sebagai bagian dari proyek POSTGRES di Universitas California di Berkeley dan memiliki lebih dari 35 tahun pengembangan aktif pada platform inti.

2.2.9 Sendgrid Mail Service

Menurut dokumentasi Sendgrid, Sendgrid adalah layanan email yang dapat digunakan dengan API. Jenis layanan email ini adalah penyedia email "transaksional". Sebagian besar email yang masuk dari suatu web akan terkait dengan layanan yang disediakan oleh web. Statistik berapa banyak email yang dikirim, apa yang dibuka (jika didukung), dan bahkan log pengiriman tersedia dengan panelnya. Beberapa filter anti-spam, memisahkan email biasa dari email transaksional memastikan email masih dapat dikirim dan masih dapat berkomunikasi dengan klien. Sendgrid memiliki beberapa fitur yang tersedia yaitu: SMTP Service, Custom API Integration, Open & Click Tracking, Email Template Engine, File attachment, SMTP API.

2.2.10 Wireshark

Wireshark adalah penganalisis paket jaringan. Penganalisis paket jaringan akan mencoba menangkap paket jaringan dan mencoba menampilkan data paket tersebut sedetail mungkin. Beberapa tujuan yang dimaksudkan adalah administrator jaringan menggunakannya untuk memecahkan masalah jaringan, network security engineer menggunakannya untuk memeriksa masalah keamanan, developer menggunakannya untuk men-debug implementasi protokol, orang menggunakannya untuk mempelajari internal protokol jaringan (Ulf Lamping, 2004).

2.2.11 Throughput

Througput merupakan parameter QoS yang merupakan kecepatan transfer data efektif, yang dapat diukur dalam satuan bps, throughput merupakan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. Rumus untuk mehitung throughput didapatkan sebagai berikut :

$$Throughput = \frac{Jumlah\ packet\ data\ yang\ dikirim\ (bit)}{Waktu\ pengiriman\ paket\ (second)}$$

Gambar 2.4 Rumus Untuk Mencari Throughput

2.2.12 Delay

Delay merupaka parameter QoS yang merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. Delay juga dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama. Delay dapat diukur sebagai berikut :

$$\text{Rata - Rata Delay} = \frac{\text{Total delay}}{\text{Total paket yang diterima}}$$

Gambar 2.5 Rumus Untuk Mencari Delay

Menurut TIPHON QoS Delay dapat dikategorikan pada tabel dibawah

Tabel 2.2 Tabel Kategori QoS Delay

No	Kategori	Besar Delay (ms)
1	Sangat Baik	< 150
2	Baik	150 – 249
3	Cukup Baik	250 – 349
4	Buruk	350 – 449
5	Tidak Direkomendasikan	> 449

2.2.13 Packet Loss

Packet loss merupakan parameter QoS yang menggambarkan Suatu kodisi jumlah total paket yang hilang, hal tersebut dapat terjadi karena collision dan congestion pada jaringan dan hal tersebut berpengaruh pada semua aplikasi karena retransmisi akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah bandwith cukup tersedia untuk aplikasi tersebut. Packet loss dapat diukur sebagai berikut :

$$\text{Packet Loss} = \frac{(\text{Paket dikirim - paket diterima}) \text{ X } 100\%}{\text{Paket dikirim}}$$

Gambar 2.6 Rumus Untuk Mencari Packet Loss

Menurut TIPHON QoS packet loss dapat dikategorikan pada tabel dibawah

Tabel 2.3 Tabel Kategori QoS Packet Loss

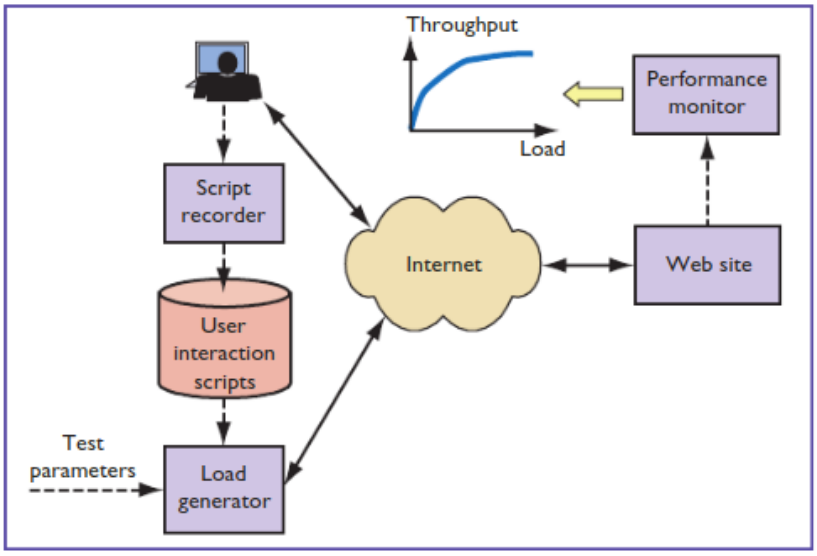
No	Kategori	Besar (%)
1	Sangat Baik	0 – 2
2	Baik	3 – 14
3	Cukup Baik	15 – 25
4	Buruk	> 25

2.2.14 Load Testing

Menurut Daniel A. Menascé, untuk mengukur performa sebuah infrastruktur IT dapat diukur dengan menggunakan metode load testing. Load testing memungkinkan untuk menilai bagaimana website support dengan beban kerja yang diharapkan dengan menjalankan serangkaian skrip tertentu yang meniru perilaku user pada tingkat beban yang berbeda. Load testing merupakan metode untuk mengukur QoS (Quality of Service) dari sebuah website. Kunci untuk menilai seberapa baik aplikasi berbasis web memenuhi harapan berdasarkan dua ukuran utama yaitu availability dan response time. Availability mengukur persentase waktu user dapat mengakses aplikasi. Response time mengukur berapa lama aplikasi memberikan respon.

Load generator meniru perilaku browser; ia terus-menerus mengirimkan permintaan ke situs web, menunggu beberapa saat setelah situs mengirimkan balasan atas permintaan tersebut, dan kemudian mengirimkan permintaan baru. Load generator dapat meniru ribuan pengguna secara bersamaan untuk menguji

skalabilitas situs web. Setiap browser yang ditiru disebut pengguna virtual, yang merupakan konsep pengujian beban utama. Uji beban hanya valid jika pengguna virtual; perilaku memiliki karakteristik yang mirip dengan pengguna sebenarnya. Oleh karena itu dalam menjalankan load testing harus memastikan bahwa pengguna virtual mencakupi : mengikuti pola yang mirip dengan pengguna sebenarnya, menggunakan waktu yang realistis, dan bereaksi seperti pengguna yang frustasi, meninggalkan sesi Web jika waktu respons berlebihan.



Gambar 2.7 Proses Load Testing