BAB 2
TINJAUAN PUSTAKA

2.1 Cloud Computing

Cloud computing adalah model untuk memungkinkan akses jaringan yang mudah dan on-demand terhadap sebuah pool sumber daya komputasi yang dapat disesuaikan (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) yang dapat diperlukan dan dilepas dengan upaya manajemen yang minimal atau interaksi dengan penyedia layanan. Model ini mempromosikan ketersediaan dan terdiri dari lima karakteristik esensial, tiga model layanan, dan empat model deployment[12].

Cloud Computing dalam bahasa Hacker Server konvensional akan di batasi oleh jumlah core processor, harddisk dan memory. Dengan keterbatasan fisik yang ada maka tidak mungkin membebani sebuar server konvensional dengan beban maksimal. Jika sumber daya habis, maka biasanya kita harus menginstall ulang seluruh aplikasi dan data di server yang kapasitasnya lebih besar memigrasi semua aplikasi yang ada ke server yang baru. Ini akan membutuhkan waktu 1-2 hari untuk menyiapkan sebuah server baru, itu pun kalau tidak ada masalah[13].

Cara kerja sistem Cloud Computing adalah server cloud dan sistem penyimpanan data terletak di tempat yang nyata tetapi lebih virtual karena dapat diakses dari komputer client. Pusat-pusat data dapat menyimpan informasi yang dibutuhkan, semacam video, audio, file, atau gambar untuk diakses[14].

Ada pun manfaat Cloud Computing adalah sebagai berikut :

1. Mengurangi biaya teknologi

   Cloud Computing menghemat biaya dalam hal biaya infrastruktur kebutuhan hardware, menghemat biaya listrik dan mengurangi tenaga IT professional sehingga menghemat biaya perawatan.

2. Meningkatkan kapasitas

   Kapasitas Cloud Computing memang tergantung pada biaya sewa, namun teknologi Cloud Computing bisa menyimpanan data pada Cloud Computing lebih besar daripada komputer pribadi.

3. Update otomatis

   Di dalam Cloud Computing kita tidak perlu khawatir dengan Update server dan software, karena semua itu telah dilakukan secara otomatis.

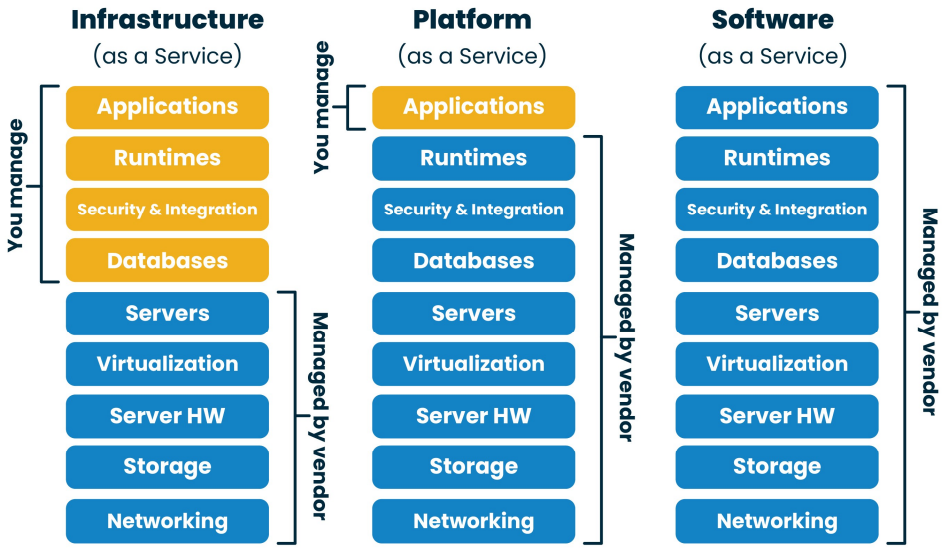
4. Availability

   Availability berarti Cloud Computing mempunyai jaminan dapat diakses 7 x 24 jam dari mana saja dan kapan saja.

5. Mobilitas

   User dapat mengakses informasi dimana pun mereka berada, mereka tidak perlu membuka komputer untuk mendapatkan informasi yang mereka butuhkan.

Ditinjau dari model layanan, Cloud Computing dibagi menjadi tiga jenis yang saling berhubungan, yakni Software as a Service (SaaS), Platfrom as a Service (PaaS), dan Infrastructure as a service (IaaS). Pada dasarnya, perbedaan utama pada ketiga jenis layanan ini terletak pada tanggung jawab pengguna dan penyedia layanan terhadap layanan yang diberikan. Ada pun model layanan dari Cloud Computing seperti pada Gambar2.1 [15]:



Gambar 2. 1 Model Layanan Cloud Computing

1. Platform as Service (PaaS)

   Hal ini memfokuskan pada aplikasi dimana dalam hal ini memungkinkan developer untuk tidak memikirkan hardware dan tetap fokus pada Application developmentnya tanpa harus mengkhawatirkan operating sistem, infrastructure scaling, load balancing dan lainya[15].

2. Software as a Service (Saas)

   Layanan dari Cloud Computing dimana kita tinggal memakai software (perangkat lunak) yang telah disediakan. Kita cukup tahu bahwa perangkat lunak bisa berjalan dan bisa digunakan dengan baik. Contoh: layanan email publik (Gmail, YahooMail, Hotmail,dsb), social network (Facebook, Twitter, dsb) instant messaging (YahooMessenger, Skype, GTalk, dsb) dan masih banyak lagi yang lain. Dalam perkembangan-nya, banyak perangkat lunak yang dulu hanya kita bisa nikmati dengan menginstall aplikasi tersebut di komputer kita (on-premise) mulai bisa kita nikmati lewat Cloud Computing. Keuntungannya, kita tidak perlu membeli lisensi dan tinggal terkoneksi ke internet untuk memakai-nya. Contoh: Microsoft Office yang

sekarang kita bisa nikmati lewat Office 365, Adobe Suite yang bisa kita nikmati lewat Adobe Creative Cloud, dan sebagainya[15].

3. Infrastructure as a service (Iaas)

   Layanan dari Cloud Computing dimana kita bisa "menyewa" infrastruktur IT (komputasi, storage, memory, network dsb). Kita bisa definisikan berapa besar-nya unit komputasi (CPU), penyimpanan data (storage) , memory (RAM), bandwith, dan konfigurasi lain-nya yang akan kita sewa. Mudah-nya, Iaas ini adalah menyewa komputer virtual yang masih kosong, dimana setelah komputer ini disewa kita bisa menggunakan-nya terserah dari kebutuhan kita. Kita bisa install sistem operasi dan aplikasi apa pun diatas-nya.Jaringan clientserver[15].

2.2 Virtualisasi

Virtualisasi adalah suatu metode menggabungkan dan saling berbagi sumber daya teknologi informasi yang bertujuan untuk mempermudah pengelolaan dan meningkatkan penggunaan aset sehingga sumberdaya teknologi informasi lebih maksimal digunakan serta dapat memenuhi permintaan bisnis, dengan virtualisasi suatu perangkat komputer server dapat digunakan berbagai aplikasi dan sistem operasi sehingga seolah-olah beroperasi dalam beberapa aset. Hal ini dapat meningkatkan utilitas aset dan mengurangi biaya kebutuhan aset fisik. Inti dari virtualisasi adalah membuat suatu simulasi dari perangkat keras, sistem operasi, jaringan dan lainya. virtualisasi digunakan sebagai sarana untuk improvisasi skalibilitas dari perangkat keras yang ada[16].

Terdapat delapan istilah dalam penerapan virtualisasi. Diantaranya adalah *virtualisasi* sistem operasi, *virtualisasi* aplikasi server, *virtualisasi* aplikasi, manajemen *virtualisasi*, *virtualisasi* jaringan, *virtualisasi* perangkat keras, *virtualisasi* penyimpanan dan layanan *virtualisasi*[16].

2.2.1 Manfaat Virtualisasi

Virtualisasi memberikan banyak manfaat dibandingkan infrastruktur biasa. yaitu diantaranya dengan virtualisasi dapat mengurangi biaya fiskal dan kemudahan

dalam pengelolaan dan pemasangan tambahan. layanan virtualisasi dapat memungkinkan guest tertentu dapat diduplikasi dengan mudah sehingga dapat digunakan dalam melakukan testing aplikasi yang akan dipasang meski dalam lingkungan operasi sistem yang berbeda[17].

Berikut adalah beberapa manfaat menggunakan virtualisasi:

1. Meningkatkan Efektifitas Komputer server.

   Virtualalisasi memberikan manfaat meningkatkan keefektifan penggunaan sumberdaya komputer seperti RAM dan Prosesor. serta penghematan terhadap energi listrik dengan sedikit pembelian terhadap fisikal server. penerapan virtualisasi tersebut dengan menggunakan beberapa virtual machine dalam satu fisikal server /Host.

2. Memudahkan dalam Proses Backup dan Recovery.

   Setiap server yang dijalankan di dalam sebuah mesin virtual dapat disimpan dalam sebuah image yang berisi seluruh konfigurasi system. Jika suatu saat terjadi masalah pada server yang berjalan, maka tidak perlu memasang dan konfigurasi ulang, cukup dengan mengambil salinan dari image yang telah disimpan.

3. Memudahkan dalam Proses Pemasangan.

   Dengan virtualisasi server virtual dapat dikloning dan dapat dijalankan pada komputer lain dengan mengatur sedikit konfigurasi sehingga mengurangi beban kerja para staf teknologi informasi dan dapat mempercepat proses implementasi suatu system.

4. Memudahkan dalam Pengelolaan dan Pemeliharaan.

   Jumlah perangkat komputer yang lebih sedikit secara otomatis akan mengurangi waktu dan biaya dalam pengelolaan, serta memberikan kemudahan dalam proses pemeliharaan perangkat yang lebih sedikit.

5. Melakukan Standardisasi Perangkat Keras.

Virtualisasi akan melakukan dan enkapsulasi perangkat keras sehingga proses pengenalan dan pemindahan suatu spesifikasi perangkat keras tertentu tidak menjadi suatu masalah karena akan secara otomatis terbaca oleh sistem.

2.2.2 Tipe Virtualisasi

1. Virtualisasi Perangkat Keras

   Virtualisasi perangkat keras adalah upaya menciptakan mesin virtual yang bekerja layaknya sebuah komputer beserta dengan sistem operasi. dalam virtualisasi perangkat keras, mesin host adalah mesin tempat virtualisasi itu berada, dan mesin guest itu sendiri[18].

   Ada beberapa jenis virtualisasi perangkat keras yaitu diantaranya:

   1) Virtualisasi penuh yaitu virtualisasi yang hampir menyerupai mesin asli dan mampu menjalankan sistem operasi tanpa perlu diubah .
   2) Virtualisasi sebagian yaitu virtualisasi di mana tidak semua aspek lingkungan disimulasikan dan tidak semua perangkat lunak dapat langsung berjalan, beberapa perlu disesuaikan agar dapat berjalan dalam lingkungan virtual tersebut.
   3) Para virtualisasi yaitu virtualisasi perangkat tidak disimulasikan tetapi perangkat lunak guest berjalan dalam domain nya sendiri seolah-olah dalam sistem yang berbeda. Dalam hal ini perangkat lunak tamu perlu disesuaikan untuk dapat berjalan.

2. Virtualisasi Perangkat Lunak

   Virtualisasi perangkat lunak adalah virtualisasi yang memungkinkan satu komputer host untuk membuat dan menjalankan satu atau lebih lingkungan virtual. virtualisasi perangkat lunak banyak digunakan untuk mensimulasikan sebuah sistem komputer lengkap, dengan tujuan untuk memungkinkan sistem operasi guest berjalan[18].

2.3 Virtualisasi Kontainer

Di dalam konteks virtualisasi Kontainer dapat diartikan sebagai alat yang dapat dipergunakan untuk system yang terisolasi yang terletak pada level operasi sistem yang yang dijalankan pada satu induk kernel atau Host[19].

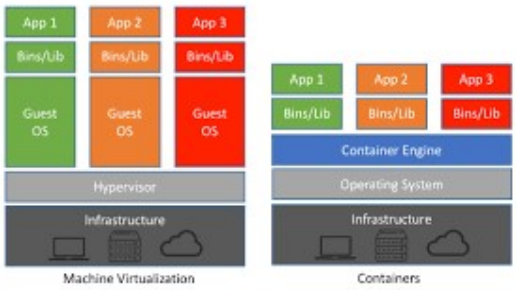
Terdapat 2 jenis Kontainer yang umum digunakan yaitu

1. Kontainer berbasis sistem operasi adalah suatu kontainer yang memberikan isolasi pada level sistem operasi dan memanfaatkan kernel yang sama dari suatu induk. contohnya: LXC, openVZ dan lainya.

2. Kontainer berbasis aplikasi adalah suatu kontainer yang memberikan isolasi pada level aplikasi dengan memanfaatkan beberapa komponen yang ada pada sistem operasi induk yang ditambah dengan beberapa komponen pada kontainer lain yang menjadi dasar dari berjalannya suatu aplikasi. Contohya: Docker,Rocket.

2.4 Virtual Machine dan Kontainer

Virtual machine adalah perangkat lunak yang dapat mengisolasi sebuah mesin komputer serta dapat menjalankan semua program seperti komputer aslinya atau duplikat dari komputer asli, sedangkan kontainer adalah suatu teknologi virtualisasi yang dapat mengisolasi sebuah proses dari proses yang lain nya yang akan mengisolasi library dan aplikasi yang digunakan saja tanpa mengisolasi seluruh komponen seperti perangkat keras kernel, serta sistem operasi.



Gambar 2. 2 Virtual Machine vs Container

Pada Gambar 2.2 dapat dilihat perbedaan pada layer virtualisai antara virtual machine dan juga container. Dalam teknologi virtualisasi virtual machine dan docker sama-sama memiliki konsep skema virtualisasi, akan tetapi ada yang membedakan antara virtual machine dan Kontainer, berikut adalah beberapa perbedaan mendasar antara virtual machine dan kontainer[20]:

1. Virtual machine menggunakan kernel tersendiri sehingga membuat beban pada Host menjadi lebih lebih berat sedangkan Kontainer membagi kernel nya kedalam kedalam kontainer yang sudah ada sehingga lebih efektif digunakan.
2. Virtual machine menggunakan keseluruhan sumberdaya perangkat keras yang ada pada host sehingga host tersebut menjalankan operasi sistem secara ganda bersamaan, sedangkan kontainer bersifat seperti aplikasi dan hanya sedikit menggunakan sumberdaya pada host.
3. Virtual machine tidak dapat mengalokasikan spesifikasi pada antar virtual machine sedangkan docker dapat mengalokasikan spesifikasi antar kontainer sehingga dapat melakukan efesiensi sumber daya dengan sebaik-baiknya pada sistem.

2.5 Devops

DevOps adalah serangkaian metode di mana tim pengembang dan tim operasional berkomunikasi dan berkolaborasi untuk memberikan perangkat lunak dan layanan dengan cepat, dapat diandalkan, dan dengan kualitas yang lebih tinggi. DevOps adalah pembagian tugas dan tanggung jawab di dalam sebuah tim yang diberdayakan dengan akuntabilitas penuh terhadap layanan mereka dan teknologi dasarnya, mulai dari pengembangan, penyebaran, hingga dukungan[21]

Konsep Devops ini memungkinkan peran kedua tim tersebut untuk saling berkoordinasi dan berkolaborasi untuk menghilangkan proses-proses manual yang syarat dengan kesalahan diganti menjadi proses yang terintegrasi. Tujuan utamanya adalah menghasilkan produk dengan cepat, lebih baik dan lebih

handal. Pada praktiknya, Devops menggunakan beberapa metode diantaranya [22]

1. Continuous Integration

   Continuous Integration merupakan salah satu implementasi dari DevOps yang menangani otomatisasi proses build dan test software setiap kali melakukan git push ke repositori.
2. Continuous Delivery

   Continuous Delivery merupakan salah satu implementasi dari DevOps yang menangani otomatisasi dari seluruh proses development dimulai dari build, test, 8 hingga deploy. Namun, pada Continuous Delivery proses approving masih dilakukan secara manual.
3. Continuous Deployment

   Pada Continuous Deployment, semua proses dari build, test, hingga deploy sudah sepenuhnya dilakukan secara otomatis. Adapun yang dilakukan secara manual pada Continuous Deployment adalah saat developer melakukan review terhadap code saat melakukan merging ke branch master.pada metode continous deployment ini terdapat beberapa cara untuk menerapkannya diantaranya adalah:

   1) Push Based Deployment
      a) Perubahan pada infrastruktur dibuat dan didorong ke repositori git
      b) Perubahan tersebut dalam repositori Git memicu serangkaian perintah dalam alur kerja.
      c) Serangkaian perintah ini kemudian mendorong perubahan yang diperbarui ke infrastruktur, mengaktualisasikan infrastruktur dengan versi terbaru.
   2) Pull Based Deployment
      a) Perubahan pada infrastruktur dibuat dan didorong ke repositori Git.

b) Infrastruktur secara otomatis menarik pembaruan terbaru dari repositori Git dan menerapkannya.

2.6 Cloud Native

Cloud native adalah istilah yang digunakan untuk mendeskripsikan perangkat lunak yang dibangun untuk berjalan di lingkungan komputasi awan. Aplikasi ini dirancang agar dapat diskalakan, sangat tersedia, dan mudah dikelola. Sebaliknya, solusi tradisional seringkali didesain untuk lingkungan lokal dan kemudian diadaptasi untuk lingkungan cloud. Hal ini dapat menyebabkan kinerja yang kurang optimal dan peningkatan kompleksitas.

Cloud Native Computing Foundation (CNCF), organisasi perangkat lunak sumber terbuka yang berfokus pada promosi pembangunan aplikasi berbasis cloud dan pendekatan penerapan, mendefinisikan teknologi cloud native sebagai teknologi yang "memberdayakan organisasi untuk membangun dan menjalankan aplikasi yang dapat diskalakan di lingkungan modern dan dinamis seperti cloud publik, privat, dan hybrid."

Ada tiga elemen kunci untuk setiap arsitektur cloud-native:

- *It is containerize*. Setiap bagian (aplikasi, proses, dll.) dikemas dalam wadahnya sendiri. Ini memfasilitasi reproduktifitas, transparansi, dan isolasi sumber daya.
- *It is dynamically managed*. Wadah diatur secara aktif untuk mengoptimalkan pemanfaatan sumber daya.
- *It is microservices-oriented.* Aplikasi disegmentasi ke dalam layanan mikro, yang secara signifikan meningkatkan kelincahan dan pemeliharaannya secara keseluruhan.

2.7 Gitops

Istilah Gitops pertama kali diperkenalkan oleh sebuah Perusahaan Weaveworks pada Agustus 2017. Dan istilahnya berasal dari praktik Devops yang sudah ada. Gitops adalah salah satu cara untuk melakukan Continous Delivery. Lebih spesik pada model operasi untuk membangun aplikasi berbasis

cloud native yang menyatukan Deployment, Monitoring dan Management. Ini berjalan dengan menggunakan git sebagai source of truth untuk mendefinisikan infrastuktur dan aplikasi secara deklaratif. Tujuan akhir gitops adalah untuk mempercepat pengembangan sehingga tim dapat melakukan perubahan dan pembaruan denga aman dan terjamin pada aplikasi kompleks yang berjalan pada Kubernetes[23].

2.7.1 Prinsip Gitops

1. Seluruh sistem dideskripsikan secara deklaratif.

   Salah satu aspek kunci dari GitOps adalah deskripsi deklaratif sistem. Dalam konteks ini, Kubernetes adalah salah satu contoh utama alat cloud native modern yang memungkinkan deskripsi deklaratif sistem. Pendekatan deklaratif menjamin konfigurasi sistem didasarkan pada seperangkat fakta, bukan pada seperangkat instruksi. Dengan menyimpan deklarasi aplikasi dalam Git, pengembang memiliki sumber kebenaran tunggal. Ini mempermudah proses penyebaran dan pengembalian keadaan aplikasi ke dan dari Kubernetes, serta memungkinkan reproduksi cepat dan andal infrastruktur cluster dalam situasi bencana[23].

2. Versioning State Sistem di Git

   State sistem yang diinginkan secara kanonikal diberi versi di Git, yang berfungsi sebagai sumber kebenaran utama. Dengan memiliki deklarasi sistem disimpan dalam sistem kontrol versi, pengembang memiliki satu tempat sentral dari mana segala sesuatu dapat diturunkan dan dijalankan[23].

3. Aplikasi perubahan secara otomatis

   Setelah state yang dideklarasikan disimpan di Git, langkah selanjutnya adalah memungkinkan perubahan yang disetujui secara otomatis diterapkan pada sistem. Dalam konteks GitOps, perubahan ini dapat diterapkan tanpa memerlukan kredensial klaster, karena lingkungan state

definition berada di luar lingkungan klaster. Hal ini memungkinkan pemisahan yang jelas antara apa yang ingin dicapai dan bagaimana hal tersebut dilakukan[23].

4. Memastikan kebenaran dan memonitor divergensi

   Dengan state sistem dideklarasikan dan disimpan di bawah kontrol versi, agen perangkat lunak dapat memberikan peringatan ketika realitas tidak sesuai dengan ekspektasi. Penggunaan agen perangkat lunak tidak hanya memastikan penanganan kegagalan node atau pod secara otomatis oleh Kubernetes, tetapi juga memberikan umpan balik dan kontrol yang diperlukan dalam kasus kesalahan manusia atau perbedaan yang tidak diinginkan dalam konfigurasi sistem[23].

Manfaat Utama Gitops[23]

1. Produktivitas Meningkat

   Automatisasi penyebaran berkelanjutan dengan loop kontrol umpan balik terintegrasi mempercepat waktu rata-rata penyebaran Anda. Hal ini memungkinkan tim Anda untuk mengirimkan perubahan 30-100 kali lipat lebih banyak per hari, dan meningkatkan output pengembangan secara keseluruhan 2-3 kali lipat.

2. Pengalaman Pengembang Ditingkatkan

   Mendorong kode dan bukan kontainer. Dengan menerapkan praktik terbaik GitOps, pengembang menggunakan alat yang sudah dikenal seperti Git untuk mengelola pembaruan dan fitur pada Kubernetes dengan lebih cepat. Pengembang yang baru bergabung dapat dengan cepat memahami dan menjadi produktif dalam beberapa hari daripada beberapa bulan.

3. Kepatuhan dan Stabilitas yang Ditingkatkan

   Ketika Anda menggunakan Git untuk mengelola kluster Kubernetes Anda, secara otomatis Anda mendapatkan catatan audit yang nyaman dari semua perubahan kluster di luar Kubernetes. Jejak audit tentang siapa yang melakukan apa, dan kapan terhadap kluster Anda dapat digunakan untuk memenuhi kepatuhan SOC 2 dan memastikan stabilitas.

4. Keandalan yang Lebih Tinggi

Dengan kemampuan Git untuk mengembalikan perubahan dan fork, Anda mendapatkan rollback yang stabil dan dapat direproduksi. Karena seluruh sistem Anda dijelaskan di Git, Anda memiliki sumber kebenaran tunggal dari mana untuk memulihkan setelah kegagalan, mengurangi waktu pemulihan (MTTR) dari jam menjadi menit.

5. Konsistensi dan Standarisasi yang Lebih Tinggi

   Karena GitOps menyediakan satu model untuk membuat infrastruktur, aplikasi, dan perubahan tambahan, Anda memiliki alur kerja ujung ke ujung yang konsisten di seluruh organisasi Anda. Tidak hanya pipa integrasi dan penyebaran berkelanjutan Anda semua didorong oleh permintaan tarik, tetapi tugas operasional Anda juga sepenuhnya dapat direproduksi melalui Git.

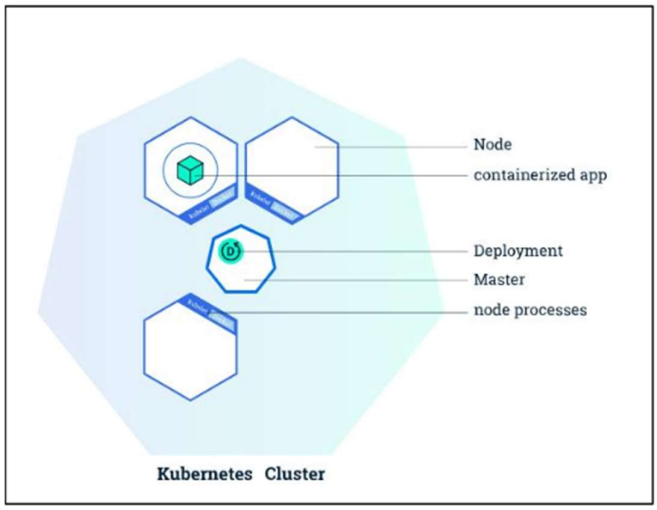
6. Jaminan Keamanan yang Lebih Kuat

   Jaminan kebenaran dan keamanan Git, didukung oleh kriptografi yang kuat yang digunakan untuk melacak dan mengelola perubahan, serta kemampuan untuk menandatangani perubahan untuk membuktikan kepemilikan dan asal, merupakan kunci untuk mendefinisikan keadaan yang diinginkan dari kluster secara aman.

2.8 Kubernetes

Kubernetes adalah sebuah program open source yang dapat melakukan otomasi untuk menjalankan, skalasi dan mengelola sebuah aplikasi yang dijalankan menggunakan container. Kubernetes awalnya dikembangkan oleh internal Google untuk kepentingan infrastruktur mereka yang masif dengan judul proyek “Project Seven”. Pada 2014 Google mengumumkan pengembangan Kubernetes ke publik dan pada 2015 Kubernetes v1.0 dirilis. Bersamaan dengan dirilisnya Kubernetes v1.0 Google juga mengumumkan kerjasama dalam pengembangan Kubernetes dengan Linux Foundation dan Cloud Native Computing Foundation yang dengan 11 kata lain Google menjadikan Kubernetes sebagai sebuah perangkat lunak open source[24].

Dalam implementasinya Kubernetes memiliki beberapa unit operasi utama yaitu cluster, node, pods dan deployment dan environment dapat dipisahkan menggunakan namespace.



Gambar 2. 3 Komponen Kluster Kubernetes

1. Cluster

   Cluster merupakan sebuah unit keseluruhan dari sebuah sistem yang berjalan dengan menggunakan Kubernetes. Dalam sebuah cluster Kubernetes terdapat beberapa node.

2. Node

   Node merupakan satu unit mesin komputer dalam cluster Kubernetes yang digunakan untuk menjalankan container dari aplikasi yang ingin dijalankan dalam Kubernetes.

3. Pods dan Deployment

   Pods adalah unit untuk sebuah aplikasi yang berjalan dalam cluster. Pod berisi konfigurasi dari container yang digunakan untuk menjalankan aplikasi dan perilaku dari pod dapat diatur melalui konfigurasi deployment.

4. Service

Unit service dalam kubernetes adalah sebuah konfigurasi endpoint yang memungkinkan aplikasi yang berjalan pada pod dapat diakses melalui jaringan di internal maupun eksternal cluster. Dengan demikian perangkat lunak yang berjalan di dalam pod dapat diakses oleh pod lain dan melalui internet. Unit-unit yang ada di dalam sebuah cluster Kubernetes dikelola pada sebuah mesin yang dikonfigurasikan dengan tool kubectl, yaitu sebuah software yang dapat dapat melakukan berbagai macam konfigurasi Kubernetes yang akan diterapkan di seluruh node dengan menggunakan Kubernetes API yang berjalan disetiap daemon service kubernetes pada setiap node.

5. Namespace

Namespace merupakan suatu metode pada Kubernetes untuk melakukan isolasi environment sehingga objek-objek yang ada dalam suatu namespace tidak saling berpengaruh dengan objek-objek pada namespace lain

No	Istilah	Definisi
1	Kubernetes	Platform orkestrasi container open-source yang mengotomatiskan penyebaran, penskalaan, dan manajemen aplikasi containerized.
2	Pod	Unit terkecil dalam Kubernetes, terdiri dari satu atau lebih container yang berjalan bersama-sama dan berbagi

		penyimpanan serta jaringan.
3	Node	Satu mesin fisik atau virtual dalam kluster Kubernetes yang menjalankan Pod.
4	Namespace	Ruang lingkup logis dalam kluster Kubernetes untuk memisahkan dan mengelola resource.
5	Deployment	Objek Kubernetes yang mengatur penyebaran dan update aplikasi secara deklaratif, memungkinkan rolling update dan rollback.
6	Service	Abstraksi dalam Kubernetes untuk mendefinisikan kumpulan Pod yang tersedia secara stabil melalui alamat IP atau DNS.
7	ConfigMap	Penyimpanan data konfigurasi tidak rahasia dalam Kubernetes yang digunakan oleh Pod dan aplikasi.

8	Secret	Penyimpanan data sensitif seperti password atau token dalam Kubernetes yang terenkripsi dan dapat digunakan oleh Pod.
9	Ingress	Sumber daya dalam Kubernetes yang mengatur akses HTTP/HTTPS dari luar kluster ke layanan di dalam kluster.
10	PersistentVolume (PV)	Penyimpanan data yang tidak volatil di Kubernetes yang dikelola secara terpisah dari Pod.
11	PersistentVolumeClaim (PVC)	Klaim sumber daya penyimpanan yang diminta oleh Pod dalam Kubernetes yang terkait dengan PersistentVolume.
12	ArgoCD	Tool untuk continuous delivery yang berbasis GitOps, mengotomatiskan penyebaran aplikasi ke kluster Kubernetes.
13	Application (ArgoCD)	Sumber daya dalam ArgoCD yang merepresentasikan aplikasi yang dideploy di

		Kubernetes, menghubungkan Git repository dengan state aplikasi di kluster.
14	Sync	Proses di ArgoCD untuk menyamakan state yang didefinisikan dalam Git repository dengan state aktual di kluster Kubernetes.
15	Rollback	Proses mengembalikan aplikasi ke versi sebelumnya jika terjadi kegagalan atau masalah pada update terbaru.
16	GitOps	Pendekatan continuous deployment di mana deklarasi infrastruktur dan aplikasi disimpan dalam Git, dan perubahan Git memicu otomatisasi deployment.
17	Repository (GitOps)	Tempat penyimpanan di Git yang menyimpan konfigurasi deklaratif dan manifest yang menggambarkan state infrastruktur dan aplikasi.

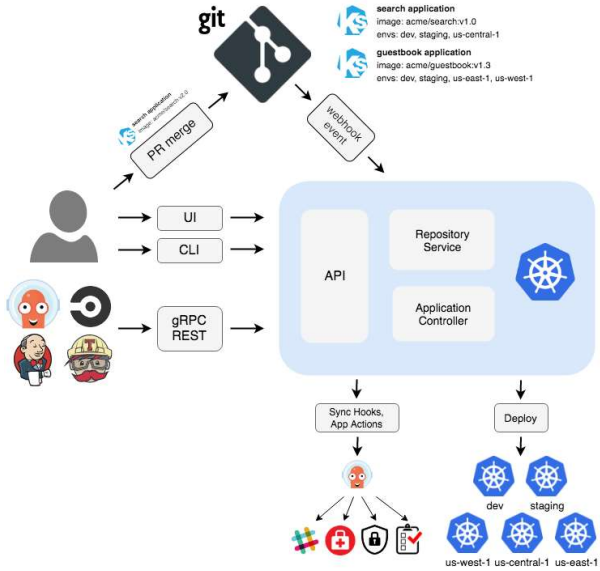
18	Pipeline	Serangkaian tahapan otomatis dalam pengembangan perangkat lunak yang meliputi build, test, dan deployment.
19	Workflow	Definisi urutan pekerjaan atau tugas dalam pipeline, termasuk build, test, deployment, dan monitor.
20	Manifest	File YAML atau JSON yang mendeskripsikan konfigurasi dan sumber daya yang akan dideploy di kluster Kubernetes.
21	Helm	Package manager untuk Kubernetes yang mempermudah manajemen aplikasi menggunakan chart yang berisi template manifest.

2.9 Argocd

ArgoCD adalah sebuah tool yang digunakan untuk mengimplementasikan GitOps, sebuah prinsip yang memungkinkan otomatisasi yang mulus dan memastikan bahwa perubahan dalam repositori Git secara otomatis tercermin dan diaplikasikan ke dalam cluster Kubernetes. ArgoCD didukung dengan user interface yang user-friendly, memudahkan pengguna dalam mengolah deployment aplikasi[10].

2.9.1 Arsitektur Argocd

Pada Gambar 2.4 dapat dilihat arsitektur dari Argocd, argocd teridiri dari beberapa komponen utama seperti git, kubernetes, server, cli



Gambar 2. 4 Arsitektur ArgoCD

ArgoCD terdiri dari beberapa komponen utama, yaitu:

1. ArgoCD Server: Komponen ini berfungsi sebagai server yang mengelola aplikasi dan melakukan sinkronisasi dengan repositori Git. Server ini dapat diakses melalui port 443.
2. ArgoCD CLI: Komponen ini berfungsi sebagai perintah baris komando yang digunakan untuk mengelola aplikasi dan melakukan operasi seperti deploy, rollback, dan lain-lain.
3. Git Repository: Repositori ini berisi kode aplikasi yang akan di-deploy ke dalam cluster Kubernetes.
4. Kubernetes Cluster: Cluster ini adalah tempat di mana aplikasi akan di-deploy dan dijalankan.

2.9.2 Cara Kerja Argocd

ArgoCD bekerja dengan mengelola aplikasi Kubernetes melalui pendekatan GitOps, di mana konfigurasi dan definisi aplikasi disimpan dalam repositori Git. Berikut adalah cara kerja ArgoCD:

1. Installasi: ArgoCD diinstall menggunakan Helm, sebuah tool yang digunakan untuk mengelola aplikasi Kubernetes.
2. Konfigurasi: Pengguna harus mengkonfigurasi ArgoCD dengan menentukan nama aplikasi, repositori Git, dan cluster Kubernetes yang akan digunakan.
3. Deploy: ArgoCD melakukan deploy aplikasi ke dalam cluster Kubernetes dengan cara mengunduh kode aplikasi dari repositori Git dan mengaplikasikannya ke dalam cluster.
4. Sinkronisasi: ArgoCD melakukan sinkronisasi terhadap repositori Git secara berkala untuk memastikan bahwa perubahan dalam kode aplikasi tercermin ke dalam cluster.
5. Rollback: Jika terjadi kesalahan dalam proses deploy, ArgoCD dapat melakukan rollback ke versi sebelumnya dari aplikasi.
6. Monitoring: ArgoCD memberikan dashboard yang memungkinkan pengguna untuk memantau status aplikasi dan melakukan perubahan jika dibutuhkan.

Dengan menggunakan ArgoCD, pengguna dapat mengelola aplikasi dengan cara yang lebih efektif dan efisien, serta memastikan bahwa aplikasi selalu up-to-date dan sesuai dengan perubahan dalam kode aplikasi.

Fitur yang dapat dimanfaatkan jika mengadopsi ArgoCD sebagai berikut :

1. Deploy IaS code ke dalam Kubernetes
2. Dashboard status deployment untuk memastikan berhasil atau tidaknya deployment.
3. Alert jika ada perubahan pada yaml code.
4. RBAC untuk developer dan Admin

5. Melakukan pengeditan yaml secara realtime pada dashboard ArgoCD dan secara otomatis terdeploy di atas OpenShift
6. Dahsboard integrasi antar container/part yang terhubung.
7. Rollback

2.10 Hashicorp Vault

Hashicorp Vault adalah sebuah sistem manajemen kunci dan keamanan yang dikembangkan oleh HashiCorp. Sistem ini dirancang untuk memastikan keamanan aplikasi dan data dengan cara mengelola kunci, kredensial, dan konfigurasi aplikasi secara efektif dan terintegrasi dengan infrastruktur aplikasi. Hashicorp Vault memungkinkan pengguna untuk mengelola kunci dan kredensial aplikasi secara sentral dan aman, serta memastikan bahwa hanya orang yang berwenang yang dapat mengakses aplikasi dan data yang dilindungi. Dengan menggunakan Vault, pengguna dapat meningkatkan keamanan aplikasi dan mengurangi risiko keamanan yang terkait dengan penggunaan kunci dan kredensial yang tidak aman[25].

Cara kerja Hashicorp Vault melibatkan beberapa langkah utama:[25]

1. Autentikasi: Pengguna harus melakukan autentikasi terlebih dahulu sebelum dapat mengakses Vault. Autentikasi dapat dilakukan menggunakan kredensial seperti username dan password, atau menggunakan metode autentikasi lain seperti biometrik atau token.
2. Kunci dan Kredensial: Vault menyimpan kunci dan kredensial aplikasi secara aman dan terorganisir. Kunci dan kredensial ini dapat diatur untuk memiliki masa berlaku yang spesifik dan dapat diaktifkan atau dinonaktifkan secara terprogram.
3. Konfigurasi Aplikasi: Vault memungkinkan pengguna untuk mengelola konfigurasi aplikasi secara efektif dan terintegrasi dengan infrastruktur aplikasi. Konfigurasi aplikasi ini dapat diatur untuk memiliki tingkat akses yang berbeda-beda, sehingga hanya orang yang berwenang yang dapat mengakses aplikasi dan data yang dilindungi.
4. Penyimpanan Data: Vault memungkinkan pengguna untuk menyimpan data aplikasi secara aman dan terorganisir. Data ini dapat diatur untuk

memiliki tingkat akses yang berbeda-beda, sehingga hanya orang yang berwenang yang dapat mengakses data yang dilindungi.

5. Monitoring dan Logging: Vault memungkinkan pengguna untuk melakukan monitoring dan logging aktivitas aplikasi secara efektif dan terintegrasi dengan infrastruktur aplikasi. Monitoring dan logging ini dapat membantu dalam deteksi dan tanggapan terhadap ancaman keamanan yang mungkin terjadi pada aplikasi.

Dengan menggunakan Hashicorp Vault, pengguna dapat meningkatkan keamanan aplikasi dan mengurangi risiko keamanan yang terkait dengan penggunaan kunci dan kredensial yang tidak aman. Vault memungkinkan pengguna untuk mengelola kunci, kredensial, dan konfigurasi aplikasi secara efektif dan terintegrasi dengan infrastruktur aplikasi, serta memastikan bahwa hanya orang yang berwenang yang dapat mengakses aplikasi dan data yang dilindungi[25]

2.11 External Secrets Operator

External Secret Operator (ESO) adalah operator Kubernetes yang dirancang untuk mengintegrasikan sistem manajemen rahasia luar seperti AWS Secrets Manager, HashiCorp Vault, Google Secrets Manager, Azure Key Vault, dan banyak lainnya. Operasi ini membaca informasi dari API-APInya dan secara otomatis menginjeksikan nilai-nilai ke dalam Secrets Kubernetes.

Arsitektur

Arsitektur External Secret Operator melibatkan komponen-komponen berikut:

1. ExternalSecret: Ini adalah sumber daya utama yang mendefinisikan apa data yang harus diambil dari sistem manajemen rahasia luar, bagaimana data harus diubah, dan bagaimana data harus disimpan sebagai Secrets Kubernetes.
2. SecretStore: Ini adalah sumber daya cluster-wide yang mendefinisikan sistem manajemen rahasia luar dan detail autentikasinya. Itu digunakan oleh ExternalSecret untuk mengambil data rahasia.

3. ClusterSecretStore: Ini adalah sumber daya cluster-wide SecretStore yang dapat diacu oleh semua ExternalSecret dari semua namespace. Itu memberikan cara sentral untuk mengelola rahasia di seluruh cluster.
4. External Secret Operator: Ini adalah operator Kubernetes yang mengelola sumber daya ExternalSecret dan SecretStore. Itu bertanggung jawab untuk mengambil data rahasia dari sistem luar, mengubahnya sesuai dengan spesifikasi ExternalSecret, dan menginjeksikan ke dalam Secrets Kubernetes.

Cara Kerja

Alur kerja External Secret Operator melibatkan langkah-langkah berikut:

1. Buat ExternalSecret: Pengguna membuat sumber daya ExternalSecret yang mendefinisikan apa data yang harus diambil dari sistem manajemen secret external, bagaimana data harus diubah, dan bagaimana data harus disimpan sebagai Secrets Kubernetes.
2. Buat SecretStore: Pengguna membuat sumber daya SecretStore yang mendefinisikan sistem manajemen rahasia luar dan detail autentikasinya.
3. Buat ClusterSecretStore (opsional): Pengguna membuat sumber daya ClusterSecretStore yang memberikan cara sentral untuk mengelola secret di seluruh cluster.
4. External Secret Operator: External Secret Operator mengawasi perubahan pada sumber daya ExternalSecret dan SecretStore. Ketika deteksi perubahan, operasi ini mengambil data secret dari sistem luar, mengubahnya sesuai dengan spesifikasi ExternalSecret, dan menginjeksikan ke dalam Secrets Kubernetes.
5. Injeksikan Rahasia: Data secret yang diubah kemudian diinjeksikan ke dalam Secrets Kubernetes, membuatnya tersedia untuk digunakan oleh aplikasi di dalam cluster.

External Secret Operator memberikan cara yang sederhana dan skalabel untuk mengelola rahasia di lingkungan Kubernetes, memungkinkan pengguna untuk

memanfaatkan kekuatan sistem manajemen rahasia luar sementara tetap mempertahankan keamanan dan isolasi cluster Kubernetes.