Kubernetes untuk DevOps / Platform Engineering

Pengantar Kubernetes

Latar Belakang: Dari Container ke Kubernetes

Sebelum munculnya Kubernetes, aplikasi dijalankan di server fisik atau virtual. Munculnya teknologi **container** (seperti Docker) merevolusi cara aplikasi dikemas dan dijalankan. Container memungkinkan:

- **Isolasi aplikasi** dalam satu environment yang konsisten.
- **Portabilitas** antara berbagai lingkungan (dev, staging, produksi).
- **Efisiensi** penggunaan resource dibandingkan VM.

Namun, seiring meningkatnya jumlah aplikasi dan container, tim mulai menghadapi tantangan baru:

- **Bagaimana menjalankan ratusan atau ribuan container sekaligus?**
- **Bagaimana mengatur komunikasi antar container?**
- **Bagaimana memastikan container tetap hidup jika salah satunya gagal?**
- **Bagaimana melakukan update aplikasi tanpa downtime?**

Jawabannya: dibutuhkan **orchestrator** untuk mengelola container dalam skala besar. Kubernetes hadir sebagai solusi untuk mengatasi keterbatasan manajemen manual dan memberikan fitur seperti **load balancing, autoscaling, self-healing**, dan **rolling update**.

Apa itu Kubernetes?

Kubernetes, sering disingkat K8s, adalah platform open-source untuk mengotomatisasi deployment, scaling, dan manajemen aplikasi berbasis container. Kubernetes awalnya dikembangkan oleh Google berdasarkan pengalaman mereka dalam mengelola ribuan container di lingkungan produksi, dan kini dikelola oleh Cloud Native Computing Foundation (CNCF).

Kubernetes menyediakan abstraksi untuk infrastruktur sehingga developer dapat fokus pada aplikasi tanpa harus memikirkan detail hardware atau server.

Sejarah Singkat Kubernetes

- **2014**: Kubernetes diumumkan oleh Google sebagai proyek open-source.
- **2015**: CNCF mengambil alih pengelolaan proyek untuk memperluas ekosistem cloud-native.
- **Sekarang**: Kubernetes menjadi standar industri untuk container orchestration dan didukung oleh hampir semua penyedia cloud besar (AWS, Azure, GCP, dan lainnya).

Kenapa Menggunakan Kubernetes?

Kelebihan:

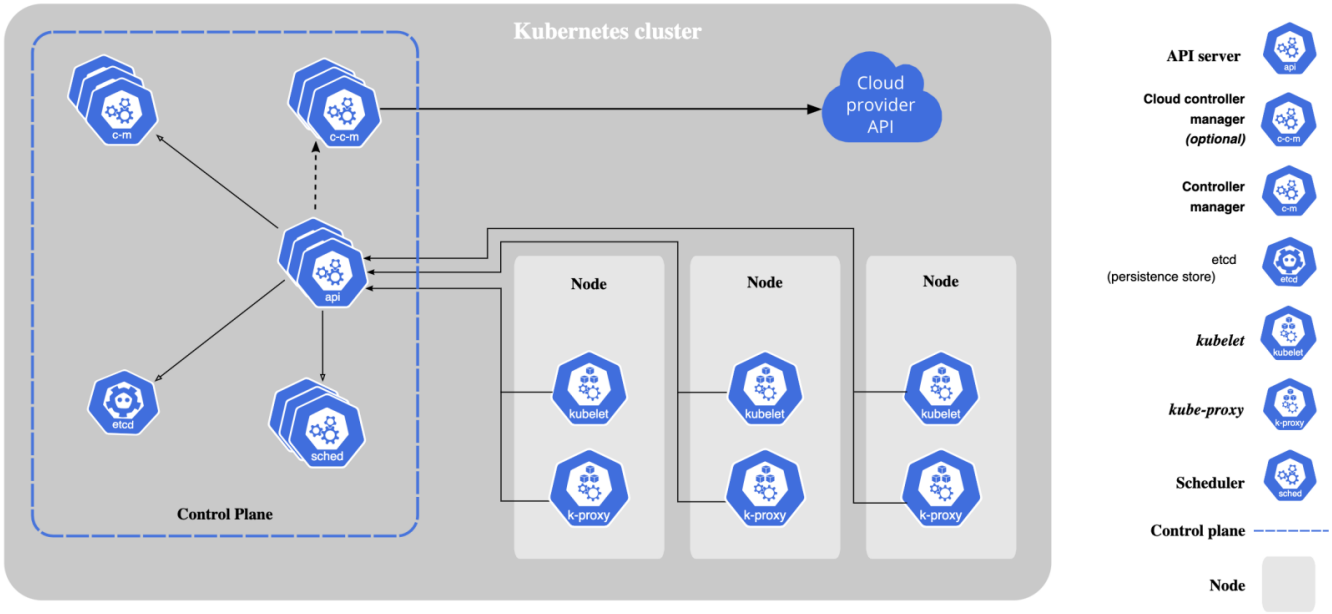
1. **Portabilitas**: Dapat berjalan di berbagai platform (on-premises, cloud, hybrid).
2. **Skalabilitas Otomatis**: Menambah atau mengurangi jumlah container sesuai kebutuhan.
3. **Self-Healing**: Kubernetes secara otomatis memulai ulang atau memindahkan container jika ada kegagalan.
4. **Rolling Updates dan Rollback**: Mempermudah pembaruan aplikasi tanpa downtime.
5. **Ekosistem yang Kaya**: Banyak tools dan integrasi seperti Helm, Istio, Prometheus, dan lainnya.

Tantangan:

1. **Kompleksitas**: Kurva belajar cukup curam untuk pemula.
2. **Resource Intensive**: Membutuhkan infrastruktur yang memadai.
3. **Keamanan dan Governance**: Perlu konfigurasi yang tepat agar tetap aman dan sesuai regulasi.

Arsitektur Kubernetes

Kubernetes memiliki arsitektur berbasis **control plane** dan **worker node**:



Control Plane

1. **API Server**: Pintu gerbang untuk semua perintah dan komunikasi.
2. **Scheduler**: Menentukan di node mana Pod akan dijalankan.
3. **Controller Manager**: Memastikan status cluster sesuai dengan yang didefinisikan.
4. **etcd**: Penyimpanan key-value untuk data konfigurasi dan status cluster.

Worker Node

1. **Kubelet**: Agen yang berjalan di setiap node untuk mengelola container.
2. **Kube-Proxy**: Mengatur komunikasi jaringan antar service.
3. **Container Runtime**: Software untuk menjalankan container (Docker, containerd, CRI-O).

Arsitektur ini memungkinkan Kubernetes untuk mengelola ribuan container secara terdistribusi, memastikan aplikasi tetap tersedia dan efisien.

Artikel ini menjadi pondasi untuk memahami materi selanjutnya dalam seri “Mastering Kubernetes”. Pada artikel berikutnya, kita akan membahas cara **menyiapkan lingkungan belajar Kubernetes menggunakan Minikube, Kind, dan K3s**.