Introduction to Kubernetes

Background of Kubernetes

Origins in Borg System

Kubernetes was inspired by Borg, Google's internal system managing millions of containers and automating resource allocation.

Open Source Release

Google open-sourced Kubernetes in 2014 to address the wider need for scalable, automated container orchestration.

Declarative Configuration Model

Kubernetes uses declarative configuration and desired state management to ensure applications run as defined.

Industry Standard Orchestration

Kubernetes revolutionized container orchestration and became the standard for cloud-native application management.

Use Cases of Kubernetes

Microservices Management

Kubernetes enables deployment and independent scaling of loosely coupled microservices for flexible application design.

CI/CD Automation

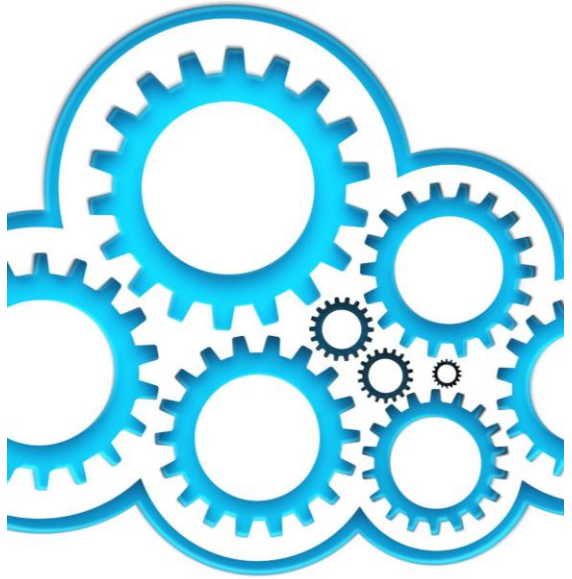
Seamless integration with CI/CD tools automates testing and deployment, streamlining software delivery.

Hybrid and Multi-Cloud

Supports workloads running across multiple cloud platforms and on-premises infrastructure for flexibility.

Self-Healing & Batch Processing

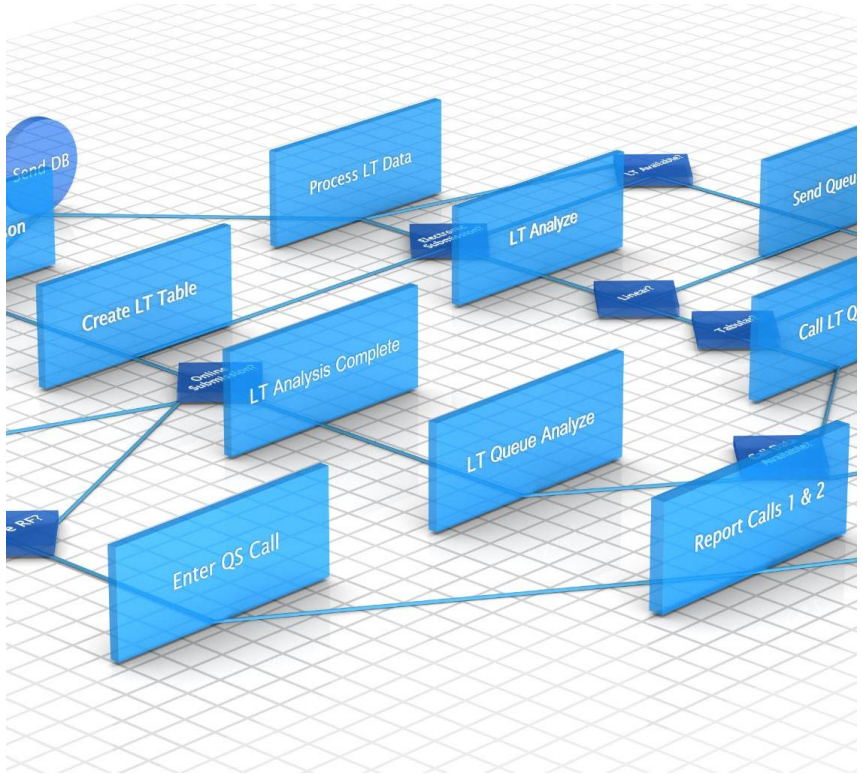
Automatically restarts containers and efficiently schedules batch and machine learning jobs dynamically.



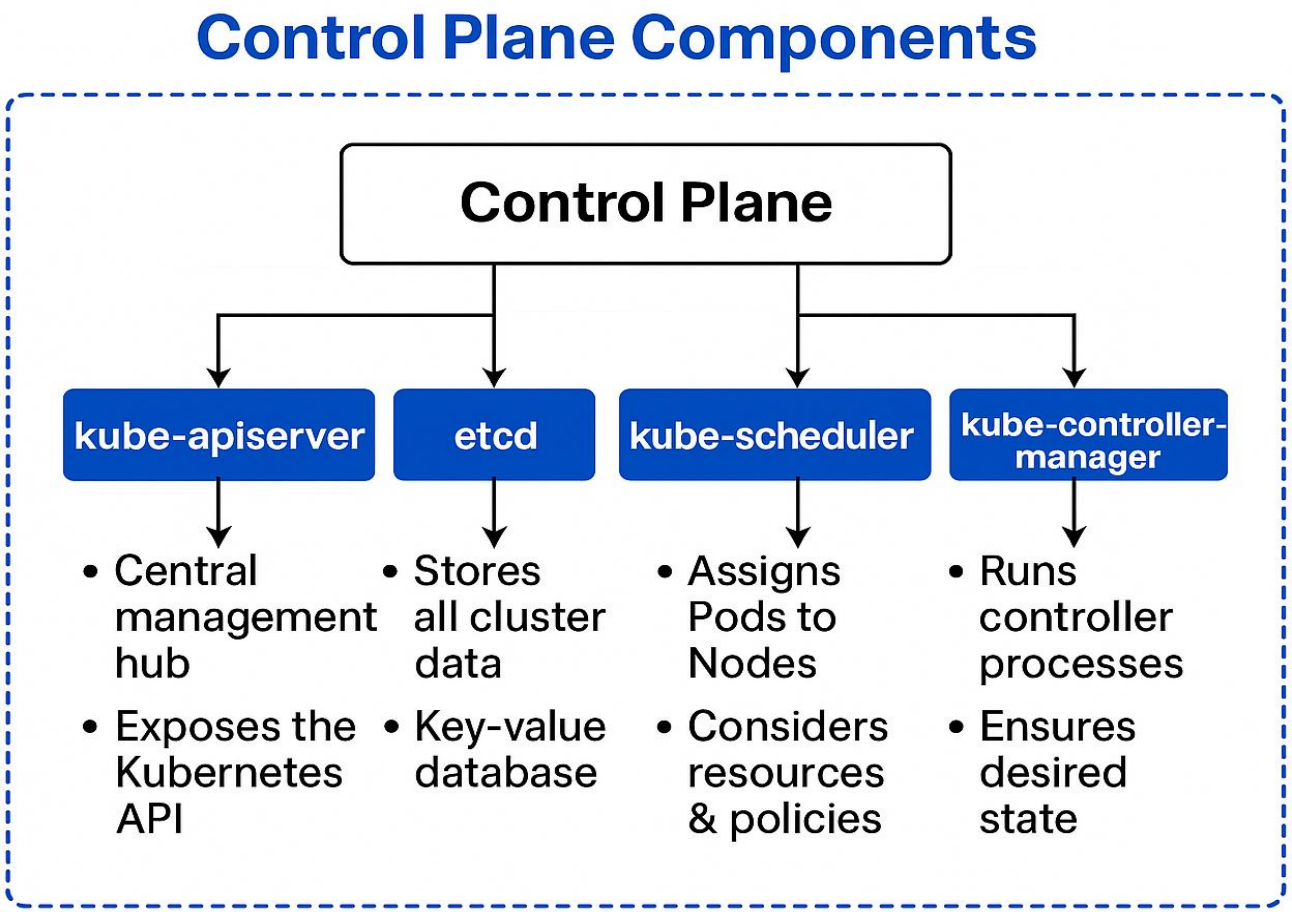
Kubernetes Architecture

Control Plane and Worker Nodes

- **Control Plane Components**
  - The Control Plane manages the Kubernetes workers
- **Worker Nodes Functionality**
  - Worker Nodes run containerized applications and include Kubelet for communication and Kube-proxy for networking.
- **Command line**
  - kubectl



Control Plane Components



Controller Manager

- Deployment Controller
  - Manages updates and rollbacks
- ReplicaSet controller
  - Ensures correct #replicas are running
- Node Controller
  - To manage node health and failures
- Job Controller
  - To manage batch jobs

Worker Nodes

Worker Node Components

Kubelet

- Agent that runs on each worker node.
- Ensures containers are running in a Pod as expected.
- Communicates with the control plane.

Container Runtime

- Software responsible for running containers (e.g., Docker, containerd).
- Pulls container images and starts/stops containers.

Kube-Proxy

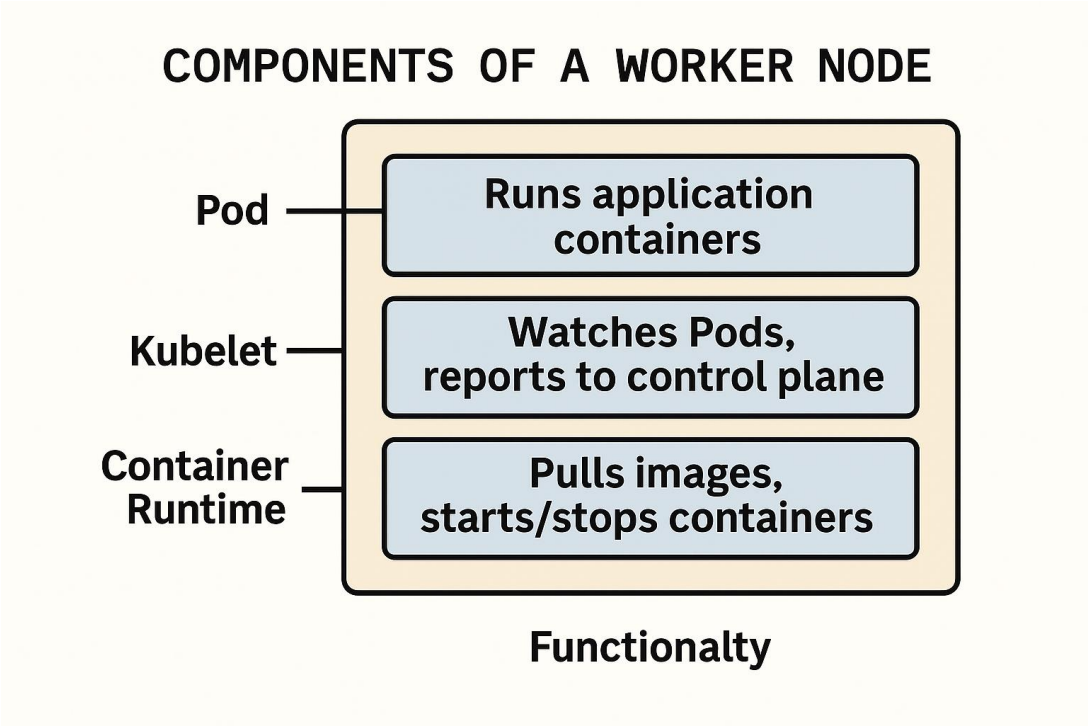
- Maintains network rules on nodes.
- Enables communication between Pods and services.
- Handles load balancing.

Pods

- Smallest deployable units in Kubernetes.
- Each Pod contains one or more containers.

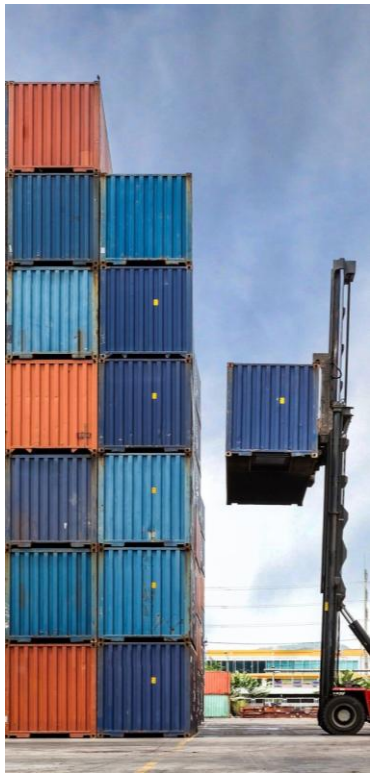
Node OS

- Underlying operating system (usually Linux).
- Provides system-level resources and networking.



Automating Deployment and Scaling

- Deployment Automation
  - Pulls container images from registries
  - Schedules them on nodes based on resource availability
  - Monitors health and restarts if necessary
- Load Balancing
  - Services distribute traffic across pods using RoundRobin
  - External traffic can be controlled using ingress controllers
- Service Discovery
  - Resolve service names to worker nodes
  - Pods can communicate using service names and not Ips
- Scaling
  - Manual – kubectl scale
  - Automatic --- Horizontal Pod Autoscaler adjusts pod replicas automatically based on resource usage metrics like CPU and memory.



Pod scheduling

Filtering

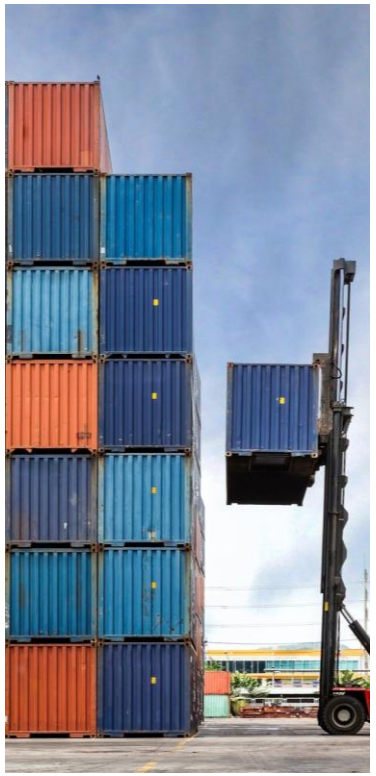
- Remove nodes that don't meet requirements
- Resources, affinity, taints

Scoring

- Rank remaining nodes based on availability and policies

Binding

- Assigns pods to the highest scoring nodes



Pod communication

- **Flat Network Model**
  - Each Kubernetes pod receives a unique IP address
  - Uses overlay networks
- **Pod IPS are ephemeral**
  - IP addresses are assigned dynamically
  - If a pod dies and is recreated, it gets a new IP
  - Direct communication using IP is not reliable

Service Resolution

- **What is a service in Kubernetes**
  - Logical set of pods
  - DNS name and a stable IP address
- **How does a pod know where a service is running**
  - DNS based service discovery
  - Uses CoreDNS inside a cluster
- **Pod A wants to talk to Pod B**
  - Pod A uses service name of Pod B
  - Kubernetes DNS resolves my-service to IP address of service IP of cluster
  - Uses IPVS or iptables to forward traffic to one of the pods
  - Uses Round Robin for load balancing
  - IPVS – kernel level load balancer

DNS Name of service

my-service.default.svc.cluster.local

Cluster IP

10.96.0.1:80

IPVS configuration

Service: 10.96.0.1:80
-> Pod A: 192.168.1.10:80
-> Pod B: 192.168.2.20:80

https://kubernetes.io/blog/2018/07/09/ipvs-based-in-cluster-load-balancing-deep-dive/

Service Resolution

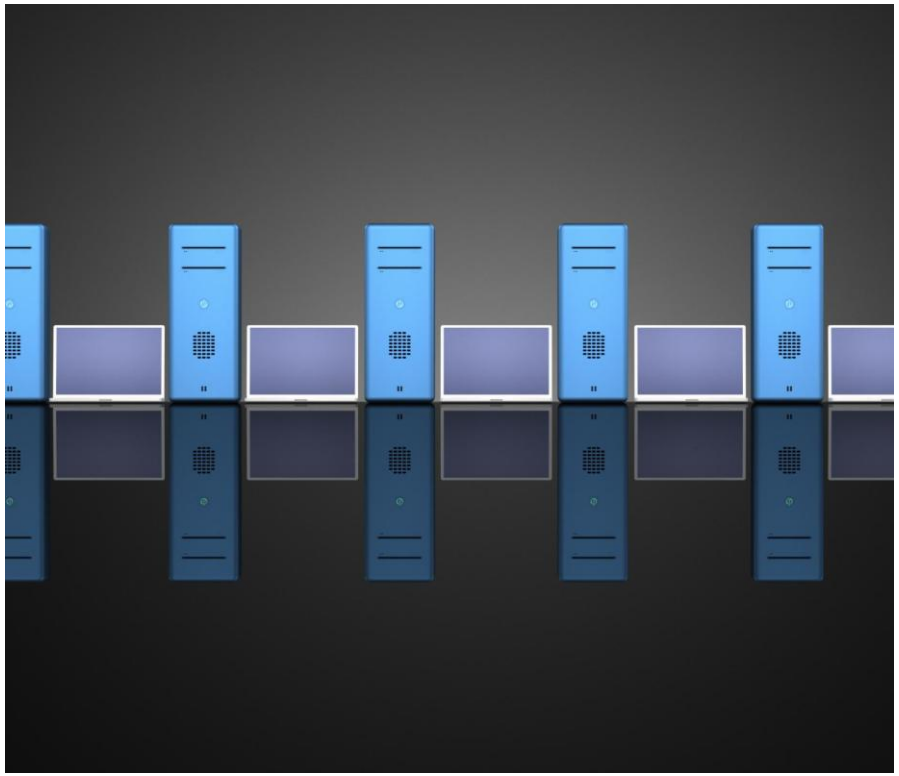
CoreDNS acts as Kubernetes' internal DNS to resolve service names into IP addresses within the cluster.

Service Naming and ClusterIP

Each Kubernetes service receives a stable ClusterIP and a DNS name for consistent addressing.

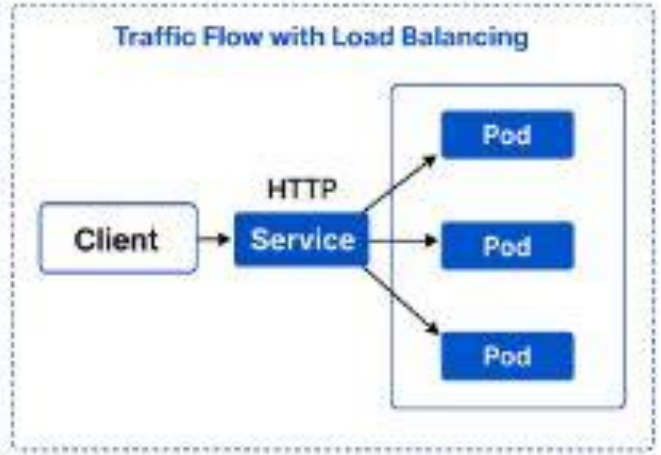
Dynamic DNS Updates

CoreDNS monitors Kubernetes API for service changes and updates DNS records dynamically in real time.



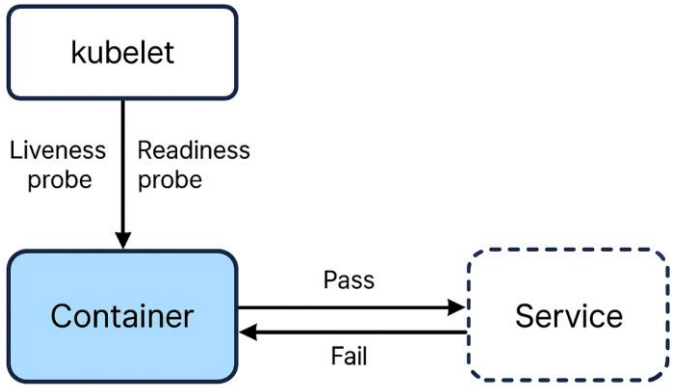
Traffic Distribution and Algorithms

- **Kubernetes Service Load Balancing**
  - Services distribute traffic across pods using label selectors and route with iptables or IPVS rules.
- **Load Balancing Algorithms**
  - IPTables – Roundrobin
  - IPVS
    - Least Connections – pods with least connections
    - Source hashing – hash source IP. Good for session persistence



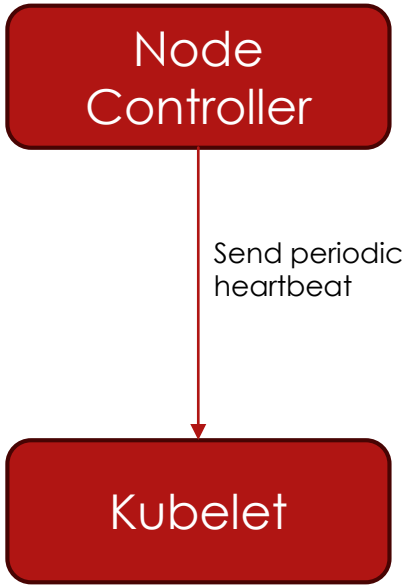
Pod High Availability

- Replica Sets
  - Define a deployment with desired #replicas
  - K8s will ensure N pods are always running
  - If a pod crashes,
    - Replica set controller automatically creates a replacement – part of the controller
- Pod distribution across nodes
  - Spreads pods across nodes to avoid single point of failure
- Probes
  - Liveness probe – detect and restart unhealthy containers
  - Readiness probe – send traffic only to healthy pods
  - Kubelet sends this to the pods running on the node



Node High Availability

- Node Monitoring
  - Node controller and kubelet
  - Checks if node is not ready
    - Periodic heartbeats from kubelet
- Set node status to ready/not ready
  - Is kubelet reporting healthy status
  - Is node reachable
- If node is marked not ready for a period
  - All pods are evicted from that node
  - Create replacement pods



Controller High Availability

- Multiple Control Plane Nodes
  - Multiple replicas of components across different nodes
    - Min 3 nodes to ensure quorum and fault tolerance
- Leader Election
  - One instance acts as leader others are standby
  - On failure, new leader is elected
- Load Balancer for API server
  - Clients connect to load balancer
- Etcd cluster
  - Cluster of odd numbered nodes to maintain quorum
  - One fails other continue serving