Course 5: Kubernetes Core Concepts & Managed Cluster Specifics

---

1. Introduction

In this course, we dive into the heart of Kubernetes.

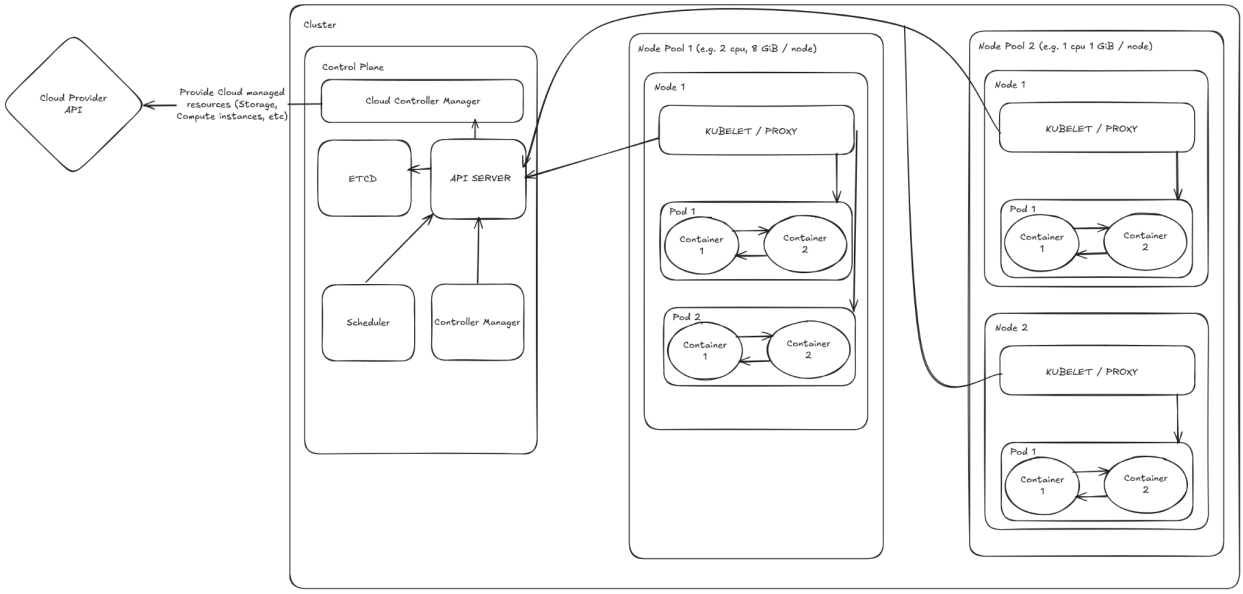
We will cover:

- The core concepts and components of Kubernetes.
- How workloads are defined, scheduled, and scaled.
- The specificities of **managed Kubernetes clusters** such as GKE (Google Kubernetes Engine) and EKS (Amazon Elastic Kubernetes Service).
- The role of **node pools** and how IAM works differently in managed clusters.

By the end of this course, you should understand the fundamental building blocks of Kubernetes and how cloud providers abstract and extend them.

---

2. Kubernetes Core Concepts



2.1 Cluster, Nodes & Control Plane

- A **Cluster** is a set of machines (nodes) managed together by Kubernetes.
- **Nodes** are the worker machines (VMs or physical servers) where pods run.
- The **Control Plane** manages the cluster state. Key components:
    - **API Server (kube-apiserver)**: the central entry point for all commands and communications.
    - **Scheduler**: assigns pods to nodes based on available resources and constraints.
    - **Controller Managers**: reconcile loops (ensuring deployments, replicasets, jobs, etc. match desired state).
    - **etcd**: a consistent, distributed key-value store holding the cluster state.

2.2 Pods, ReplicaSets, Deployments, and Other Workload Controllers

Kubernetes workloads are all about declaring a **desired state** and letting controllers enforce it.

Pods

- Encapsulates one or more containers with:
    - Shared **network namespace** (same IP/ports).
    - Shared **volumes** for storage.
- **Ephemeral**: pods can be stopped, rescheduled, or replaced at any time.

ReplicaSet

- Ensures a specified number of identical pods are always running.
- Automatically replaces failed or deleted pods.
- Usually created and managed by **Deployments** rather than directly.

Deployment

- The most common controller for **stateless applications**.
- Manages ReplicaSets and Pods.
- Supports:
    - **Rolling updates** (gradual rollout of a new version).
    - **Rollbacks** (return to a stable version if something breaks).
- Ideal for APIs, web apps, and microservices.

DaemonSet

- Ensures one pod runs **on every node** (or selected nodes).
- Used for workloads that must be present cluster-wide as:
    - Monitoring agents (Prometheus node exporter).
    - Security and observability tools.

StatefulSet

- Designed for **stateful applications** that need stable identities.
- Features:
    - Predictable pod names (`app-0`, `app-1`).
    - Persistent volume claims bound to specific pods.
    - Ordered startup and termination.

Job

- Runs pods **to completion**.
- Good for batch tasks or one-time operations.
- Retries failed pods until success or max retries is reached.

CronJob

- Runs Jobs on a **cron schedule**.
- Ideal for recurring tasks (nightly DB backups, reports, cleanup).

Key Concepts Across Controllers

- **Spec vs. Status**:
    - *Spec*: your desired state (e.g. 3 replicas).
    - *Status*: observed reality (e.g. 2 replicas ready).
    - Kubernetes continuously reconciles the two.
- **Labels & Selectors**:
    - Controllers use labels to manage pods.
    - Example: `app=myapi` groups pods for a deployment and service.
- **Immutability**:
    - Pods are replaced, not updated.
    - Deployments create new ReplicaSets when rolling out updates.

2.3 Services & Networking

- Pods get ephemeral IPs — Services provide a stable endpoint.
- Types of Services:
    - **ClusterIP** – internal-only access.
    - **NodePort** – exposes node's port.
    - **LoadBalancer** – provisions a cloud load balancer (managed clusters).
    - **Headless** – no proxy, direct DNS records.
- **CoreDNS** provides service discovery.
- **Network Policies** define allowed communication between pods/namespaces.

2.4 Storage & Volumes

- **Persistent Volumes (PVs)** represent storage resources.
- **Persistent Volume Claims (PVCs)** let pods request storage.
- **StorageClasses** define provisioners (SSD, HDD, network-attached).
- In managed clusters, PVs often map to cloud-native storage:
    - GKE → Persistent Disks
    - EKS → Elastic Block Store (EBS)

2.5 Scaling & Autoscaling

- **Horizontal Pod Autoscaler (HPA)**: scales pods based on CPU or custom metrics.
- **Vertical Pod Autoscaler (VPA)**: adjusts resource requests/limits automatically.
- **Cluster Autoscaler**: adds/removes nodes to match pod requirements.
- **Node Pool Autoscaling**: managed clusters can scale specific pools independently.

2.6 Namespaces & RBAC

- **Namespaces**: logical isolation (dev, staging, prod).
- **RBAC (Role-Based Access Control)**: grants fine-grained permissions to users, groups, or service accounts.
- In managed clusters, RBAC often works in tandem with **cloud IAM**.

---

3. Managed Kubernetes Cluster Specifics

3.1 Control Plane is Managed

- In GKE/EKS, the provider runs the control plane (API server, etcd, controllers).
- You don’t manage or upgrade these directly.
- The provider ensures high availability and patches security issues.

3.2 Node Pools

- Groups of worker nodes with the same configuration.
- Use node pools to:
    - Isolate workloads (runners vs. applications).
    - Apply autoscaling rules per pool.
    - Upgrade nodes incrementally.
- Each pool can have its own VM types, taints, and labels.

3.3 IAM & Identity

- Managed clusters combine **cloud IAM** with Kubernetes RBAC.
- Access depends on:
    - Your **cloud IAM role** (can you even connect to the cluster?).
    - Your **Kubernetes RBAC role** (what can you do inside?).
- For CI/CD (GitHub Actions):
    - Use **OIDC Identity Federation** to request short-lived tokens.
    - Tokens must be validated against the **Git project ID/path** and the **current environment** (dev/prod).
    - This prevents cross-project or cross-env misuse.

3.4 Upgrades & Versions

- Managed clusters provide **release channels** for K8s versions.
- Control plane upgrades are automatic; node pool upgrades must be scheduled.
- You can define maintenance windows.

3.5 Networking

- Managed clusters integrate directly into the cloud VPC.
- Services of type `LoadBalancer` map to managed load balancers.
- Pods/nodes get IPs from your VPC subnets.
- NAT gateways handle outbound internet if nodes are in private subnets.

---

4. Key Takeaways

- Kubernetes is structured around **control plane, worker nodes, pods, services, and controllers**.
- Workloads are deployed through controllers: **Deployments, StatefulSets, DaemonSets, Jobs, CronJobs**.
- **Services** and **DNS** provide stable networking across ephemeral pods.
- **Managed Kubernetes clusters** simplify control plane management, upgrades, and networking, but introduce cloud-specific IAM and operational rules.
- **Node pools** are the main lever for customizing and scaling worker nodes.
- IAM is hybrid: **cloud IAM for access + Kubernetes RBAC for in-cluster actions**.