Kubernetes Core Concepts

starts at 9:05 pm

AGENDA

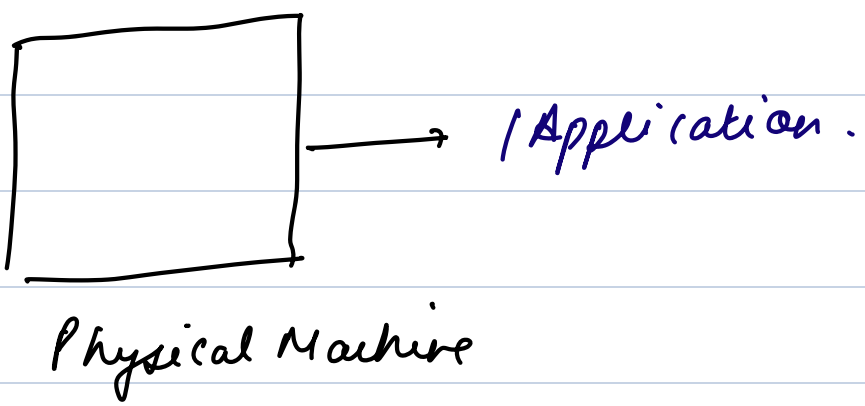
1. K8s Introduction
2. CKAD overview
3. K8s Architecture
4. K8s Types
5. Demo of kind
6. kubectl

K8s Introduction

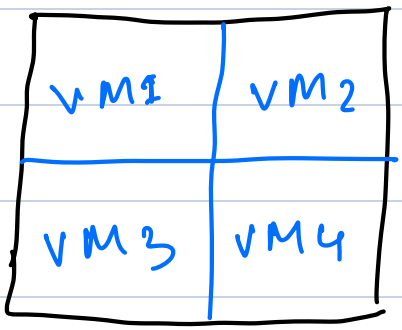
KUBERNETES → 10 letter

K 8 S

Why Kubernetes ?



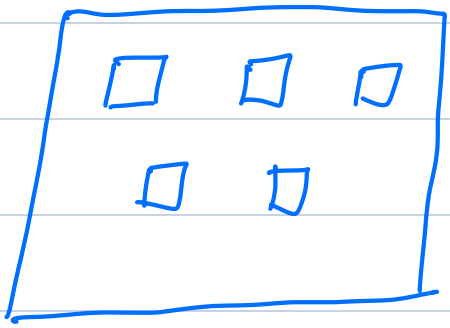
→ VM.



4 Applications.

Physical machine

→ Containers



× number applications can be run.

VM

1000s containers

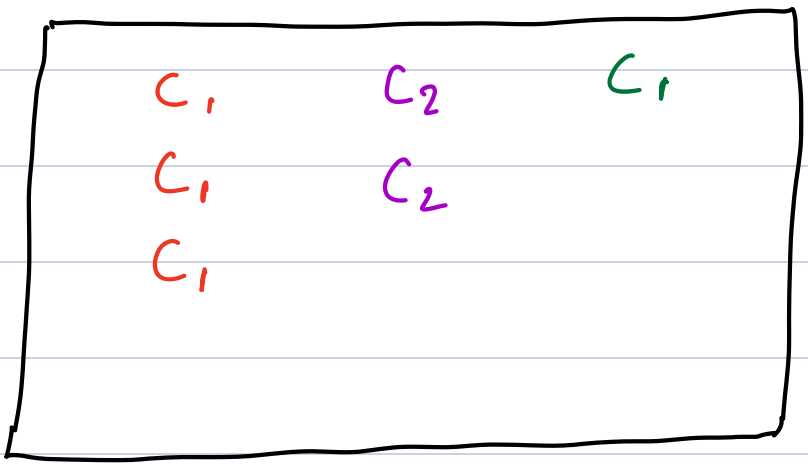
→ scheduling
→ resource allocation
→ health checks.
→ reroute failed node containers.

Manual process
→ time consuming
→ error prone

→ difficult to scale efficiently.

Amazon 3 services

① Add to Cart	→ C_1	1000	1000
② Review Cart	→ C_2	100	1000
③ Placing the order.	→ C_3	10	1000



Kubernetes Comes into Picture.

① Central Controller.

② Scalability.

→ scale more nodes.

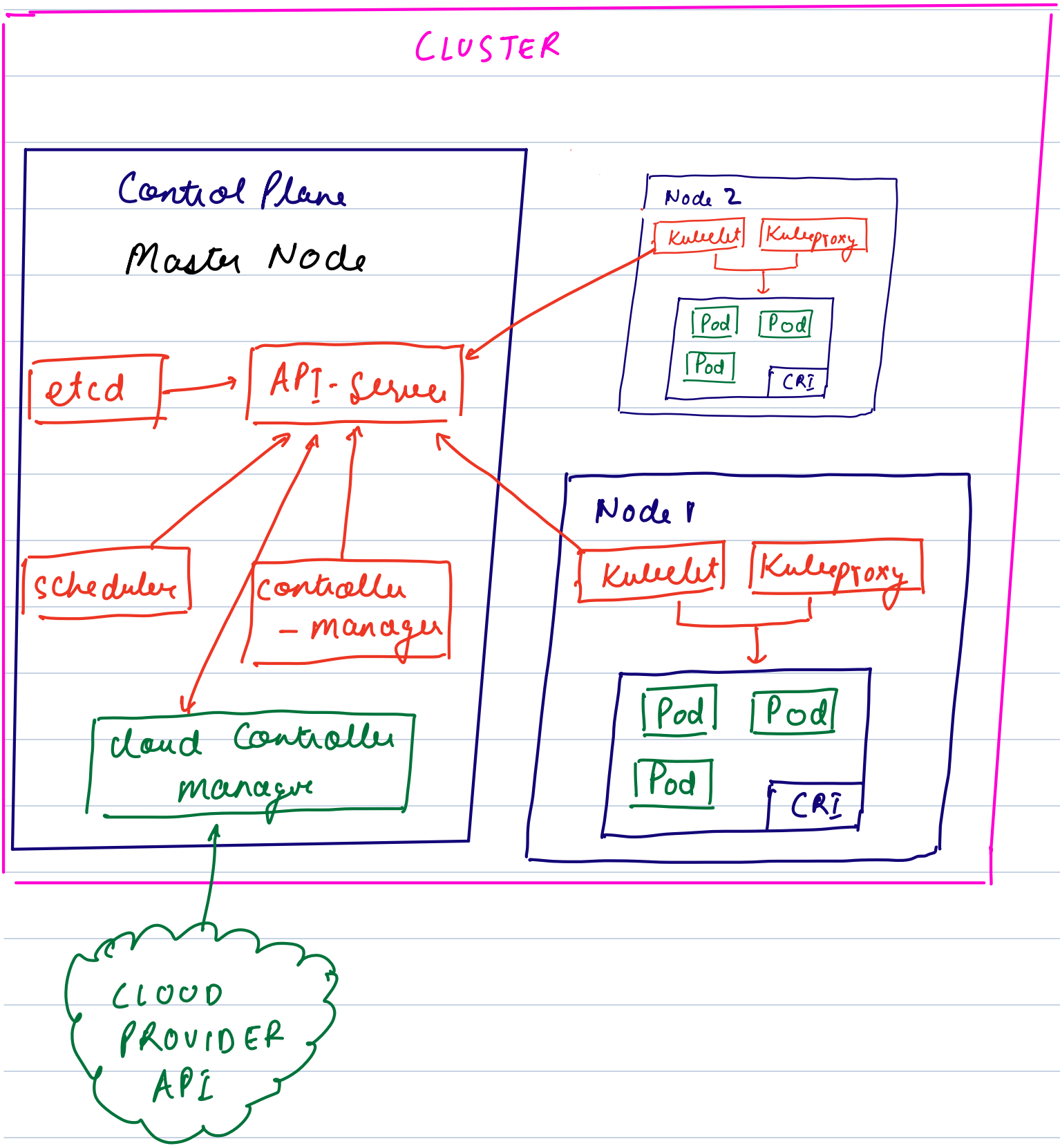
→ scale in nodes. (cost efficient)

③ Health Monitoring.

④ Portability.

Break: 10:00 pm

K8S ARCHITELTURE



Pod → Smallest unit of deployment in K8s

KUBE API server →

Authentication & Authorisation.

All interactions with the cluster happen through the API server.

Scheduler

Assigns nodes to workload (pods)

node 1
&
app.

node 2

→ 2 containers

→ 2 nodes.

→ Controller Manager

node = 2

Node Controller:

- Monitors the state of nodes.

- Marks a node as **unavailable** if it stops responding.

- Deletes Pods running on a failed node after a timeout.

- **Replication Controller:**

- Ensures that a specified number of replicas for a Pod are running at all times. $r = 5$

- Deletes or creates Pods to maintain the desired count.

- **Endpoint Controller:**

- Populates the `Endpoints` object with Pod IPs to support Services.

- **Service Account & Token Controller:**

- Manages default service accounts and API tokens for namespaces.

- **Namespace Controller:**

- Cleans up resources (e.g., Pods, Services) in a namespace marked for deletion.

- **Persistent Volume (PV) Controller:**

- Manages the lifecycle of Persistent Volumes and Persistent Volume Claims.

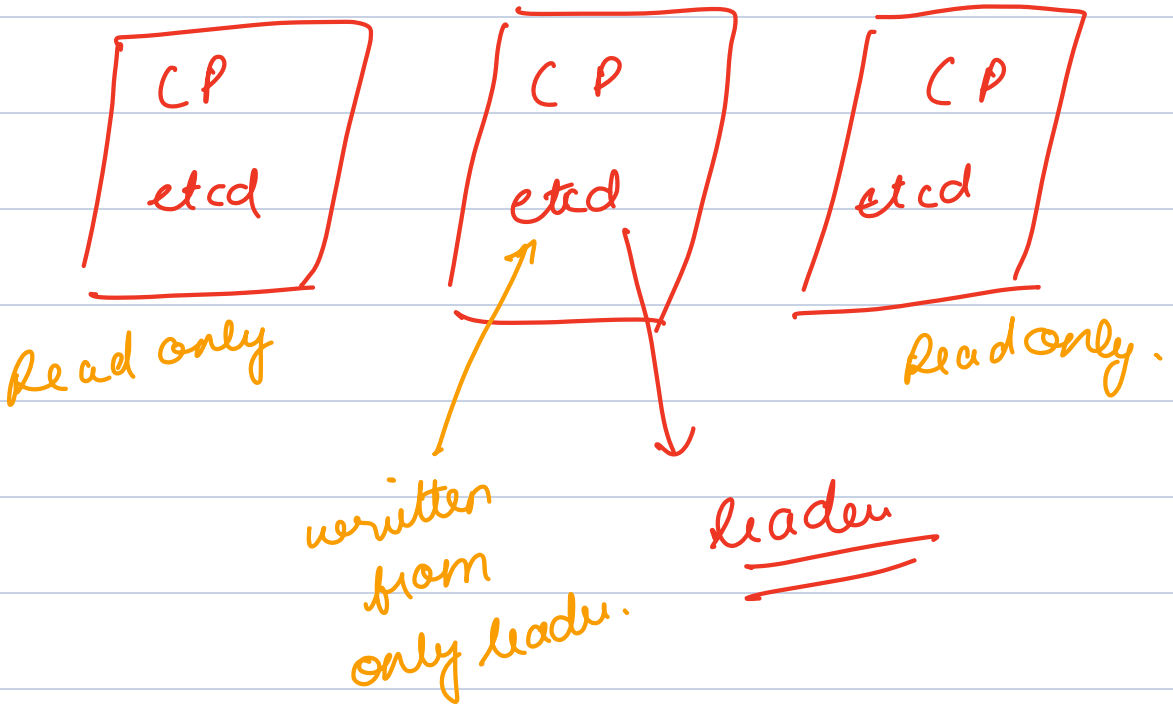
- **Job Controller:**

- Manages the execution of Jobs and ensures their completion.

ETCD → Elastic Transient Consistent Database.

It is a distributed key-value store used in Kubernetes to store all cluster-related data.

Etcd is highly reliable, consistent, and designed to provide distributed coordination and state management for clusters.



etcd is replicated across all master nodes.

ETCD stores

→ cluster state

Cloud Controller manager

→ manages cloud specific components

→ Integrates K8s with underlying Cloud Provider.

- Kubernetes uses etcd to store:

  - **Cluster state**: Information about all objects in the cluster (e.g., pods, services, ConfigMaps, secrets).

  - **Configuration data**: Stores persistent cluster-wide configuration, such as resource quotas and policies.

- **Metadata**: Includes timestamps, labels, and annotations for objects.

- **Leader election**: Facilitates Kubernetes controller manager leader elections to prevent duplication of tasks.

Worker Node.

→ Kubelet

→ responsible for running pods

→ report the nodes status.

Podspec. → Blueprint. (info of how a pod should be run)

Scheduler → Podspec → node information.

↓

Kubelet ensures containers mentioned inside podspec are running and healthy.

Few Responsibilities of Kubelet

- **Pod Lifecycle Management**:

  - Watches the **API Server** for assigned `PodSpecs`.

- Schedules and ensures the pods are running in the desired state on the node.

- Monitors the health of containers and restarts them if they fail.

- **Node Status Reporting**:

- Reports the node's condition (e.g., CPU, memory, disk usage) and availability to the API server.

- Updates conditions like `Ready`, `DiskPressure`, `MemoryPressure`, etc.

- **Resource Management**:

- Enforces **resource limits and requests** defined in the pod specifications.

- Manages Quality of Service (QoS) for pods.

- **Pod Volume Management**:

- Mounts and unmounts volumes for containers, handling storage according to the PodSpec.

- **Logging and Metrics**:

- Collects logs and metrics from running containers and provides them to monitoring or logging tools.

- Integrates with cAdvisor for monitoring resource usage.

- **Secrets and ConfigMaps**:

- Retrieves secrets and ConfigMaps from the API server and injects them into containers as needed.

Container Runtime Interface.
→ plugin which enables Kubelet to use wide variety of container runtimes.

Container Runtimes ?

Role ?

1. Image management.
2. Container lifecycle management
3. Networking
4. Storage management.

Docker

Containerd

CRI-O

/etc/default/kubelet

--container-runtime.

KUBELET_EXTRA_ARGS=--container-runtime=remote --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock

→ Kubeproxy.

KubeProxy is the component that manages network traffic routing, ensuring that requests sent to pods are correctly directed.

It runs on every node in the cluster and ensures that network traffic is correctly routed to the appropriate pods.

Types of K8s

1. Vanilla K8s
2. K8s for developers → for local dev and testing.
3. Managed K8s

   - kind. K8s in docker
   - minikube
   - Micro K8s.

   - GKE
   - EKS
   - AKS
   - OKE
   - openshift

→ Kubectl

→ docker ps	→ kubectl get pods
→ docker run nginx	→ kubectl run nginx --image=nginx

docker → Docker Engine

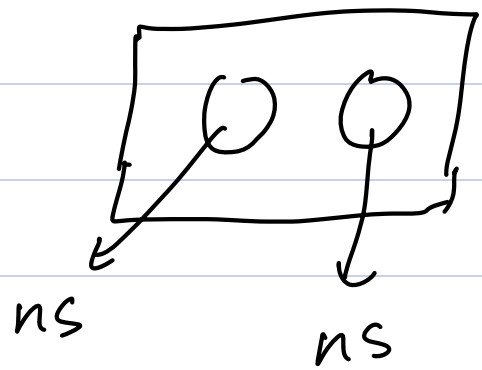
kubectl → Kubernetes.

Context (part of Kubeconfig file)

→ Cluster

→ users

→ namespaces.



Kubeconfig file.

kubectl → KC file → access cluster.

↓

API

export KUBECONFIG=/path/to/config1:/path/to/config2

View Current Context

kubectl config current-context

List All Contexts

kubectl config get-contexts

Switch Context

kubectl config use-context context-name

Set a Default Namespace for a Context

kubectl config set-context demo-context --namespace=dev

→ multi-node.yaml

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  - role: control-plane
  - role: worker
  - role: worker
```

kind create cluster --config=multi-node.yaml --name my-cluster