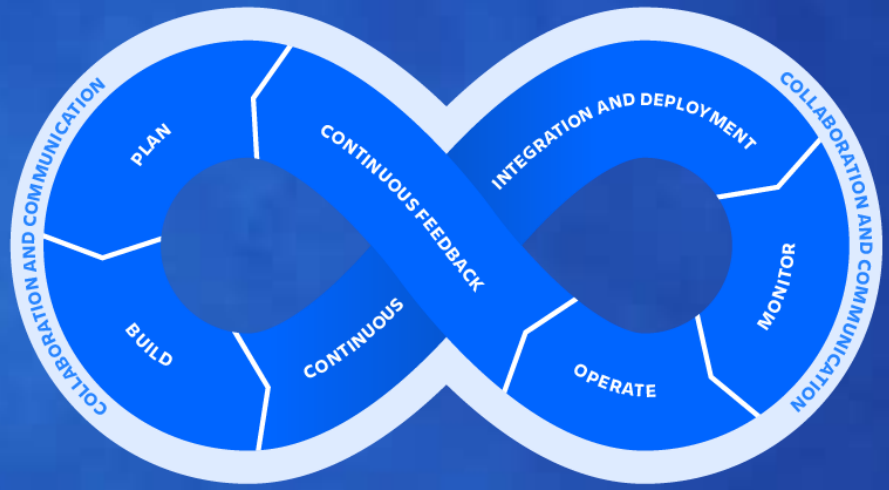


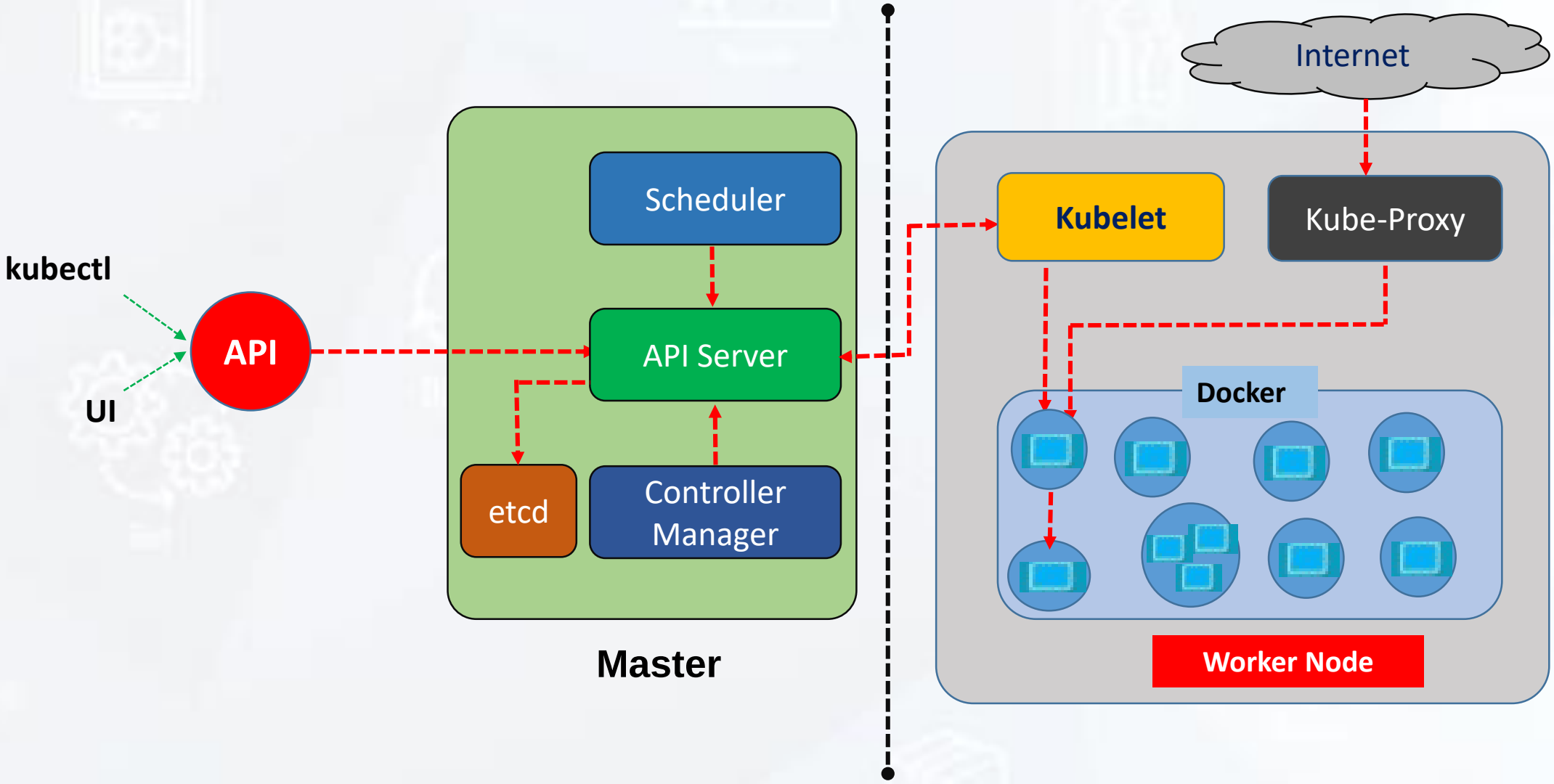
Session 17. Kubernetes Cluster Architecture

Ram N Sangwan
www.rnsangwan.com

Agenda

- Master & Minions
- Kube Apiserver
- etcd key-value store
- kube Scheduler
- kube Controller-manager
- kubelet
- kube-proxy
- kubectl Client

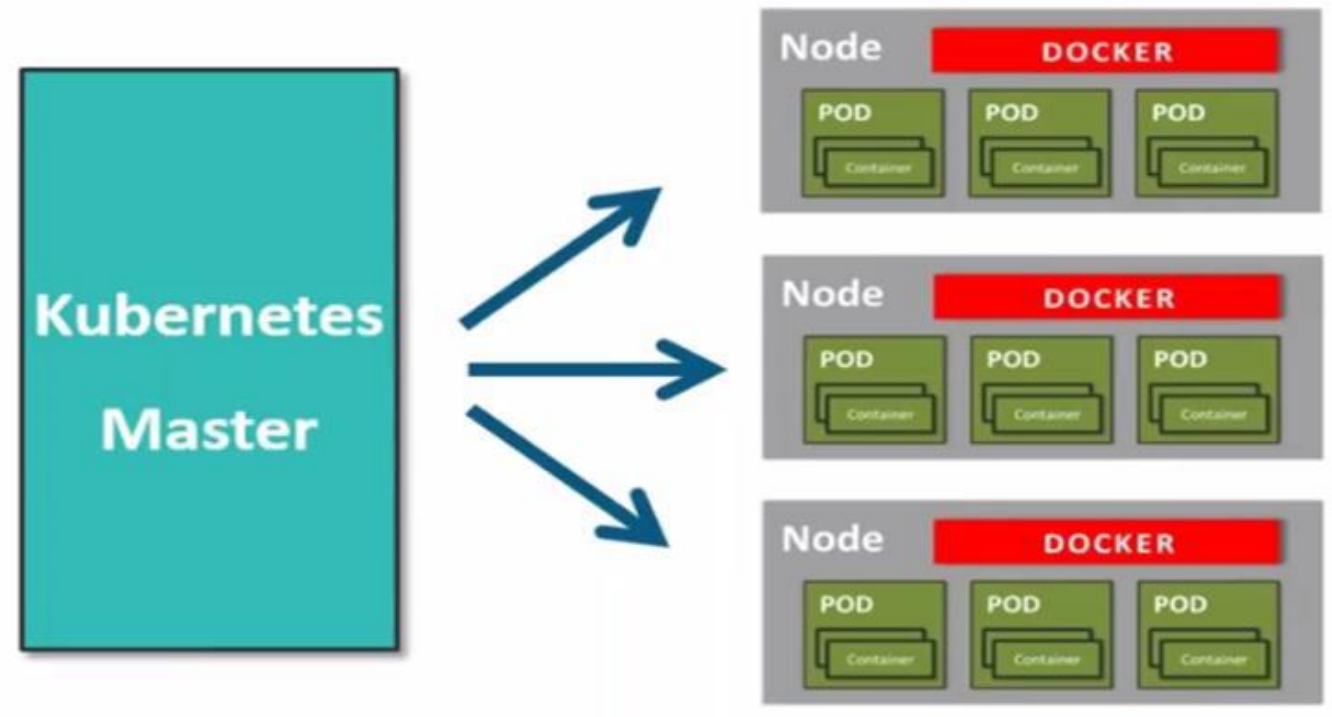
Kubernetes: Architecture



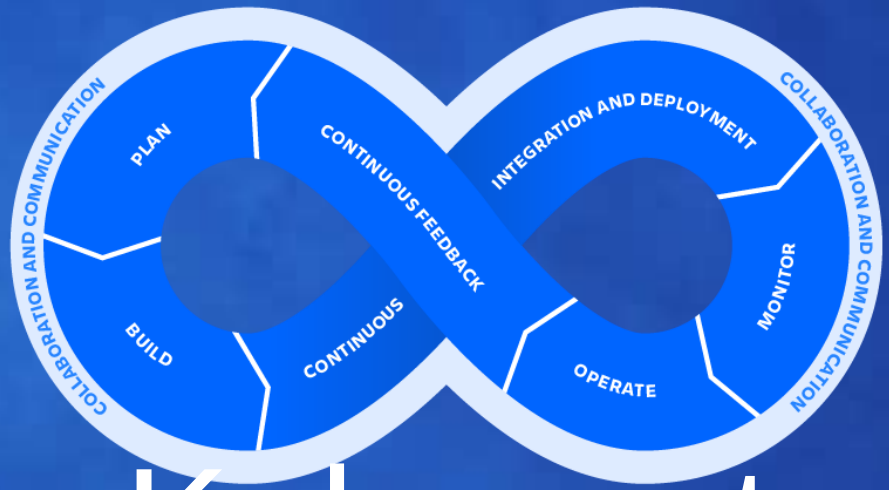
Kubernetes: Master Node

- Master node components :
  - API server
  - Scheduler
  - Controller Manager
  - etcd

Kubernetes: Working



- **Master** controls the cluster; and nodes in it
- **Nodes** host the containers inside them, Containers are inside separate PODs
- **PODSs** are logical collection of containers which need to interact with each other for an Application
- **Replication Controller** is Master's resource to ensure that requested number of pods are running on nodes always.
- **Service** is an object on Master that provides load balancing across replicated group of PODS



Kubernetes Master Machine Components

API Server

- Kubernetes is an API server which provides all the operation on cluster using the API
- API server implements an interface, which means different tools and libraries can readily communicate with it

Controller Manager

- This component is responsible for most of the collectors that regulates the state of cluster and performs a task.
- In general, it can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server

Scheduler

- This is one of the key components of Kubernetes master
- It is a service in master responsible for distributing the workload
- It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload.
- In other words, this is the mechanism responsible for allocating pods to available nodes.
- The scheduler is responsible for workload utilization and allocating pod to new node

etcd

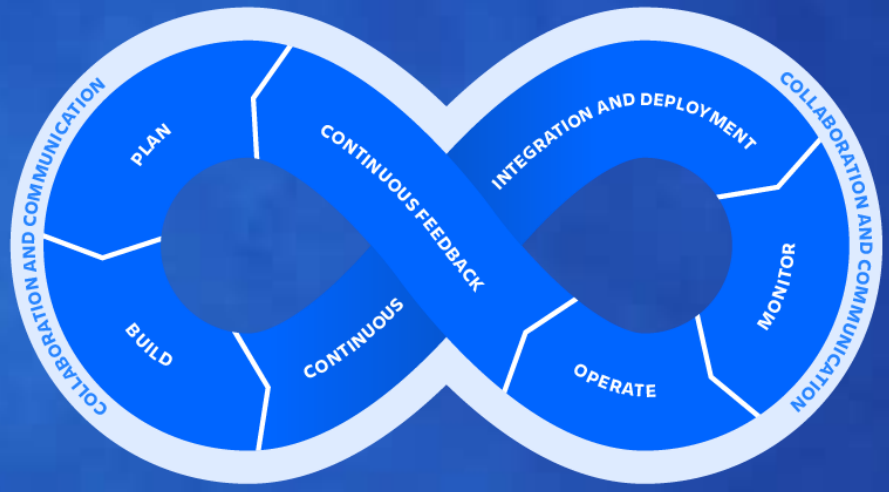
- It stores the configuration information which can be used by each of the nodes in the cluster.
- It is a high availability key value store that can be distributed among multiple nodes.
- It is accessible only by Kubernetes API server as it may have some sensitive information.
- It is a distributed key value Store which is accessible to all.

Docker

- The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment

Kubelet Service

- This is a small service in each node responsible for relaying information to and from control plane service
- It interacts with etcd store to read configuration details and wright values
- This communicates with the master component to receive commands and work
- The kubelet process then assumes responsibility for maintaining the state of work and the node server



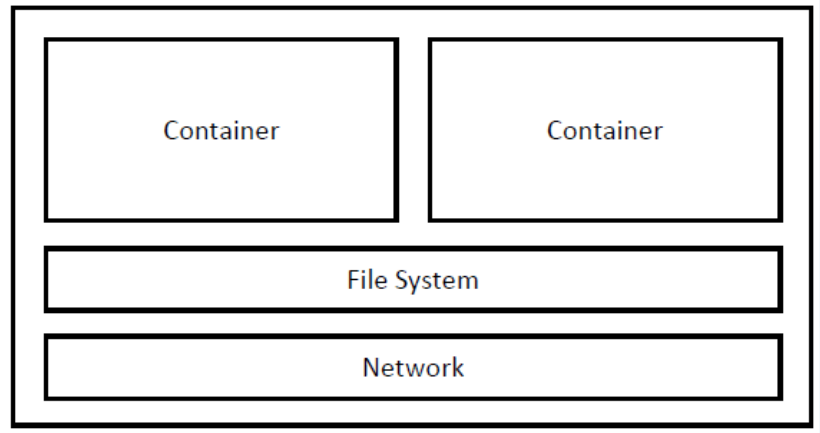
Kubernetes Node Components

kube-proxy

- To manage individual host subnetting and make services available to other components, a small proxy service called kube proxy is run on each node server
- This process forwards requests to the correct containers, can do primitive load balancing, and is generally responsible for making sure the networking environment is predictable and accessible, but isolated where appropriate

Pods

- Containers are themselves contained in objects called pods
- Pods are made up of one or more containers that work together and share a life cycle on the same node
- For example, a pod may consist of a main container that runs the application server and a helper container responsible for retrieving files when it detects changes to external repositories
- Kubernetes clusters manage each pod as a single unit



Kubernetes Service

- Kubernetes object expressing pod networking endpoint (internal / external IP address).
  - Service is associate with pod through label selector
  - ClusterIP – exposed on cluster-internal IP
  - NodePort – exposed on nodes IP .via static port
  - LoadBalancer – exposed externally on providers NLB
  - ExternalName – map service to DNS name

Kubernetes Manifest

- YAML file to declare desired state of Kubernetes object types.
  - Define Kubernetes type
  - Define type specification
  - Labels / Annotations
  - Metadata

```
---
apiVersion: v1
kind: Service
metadata:
  name: sagemath-single
spec:
  type: LoadBalancer
  ports:
  - port: 8888
    protocol: TCP
    targetPort: 8888
  sessionAffinity: ClientIP
  selector:
    app: sagemath-app
```

Kubernetes Dashboard

- Graphical user interface for interacting with a Kubernetes cluster.
  - Create, update, delete objects
  - Visual representation of state
  - Take care to properly secure

https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

Node Scaling

- Provide more or less computer infrastructure for the cluster.
  - Manually scale
  - Auto scale (depending on provider)
  - Take care to provide stateless workload

Pod Scaling

- Provide more or less processing power for an application.
  - Manually scale by increasing pod replica set count
  - Auto scale using Horizontal Pod Autoscaling (HPA)
  - Provide custom metrics for scale operations

Rolling Deployments

- Deployments are the high level objects that developers work with directly to manage the life cycles of pods
- They describe the desired state of an application
- When deployments get modified, Kubernetes automatically adjusts all replica sets, which makes it possible to perform updates without affecting application availability
- Zero downtime application updates.
  - Deployment must contain more than one replica
  - Pod are incrementally updated
  - Configurable update schema (max unavailable / available)
  - Rollback to previous version

Services

- Pods are only accessible within their Kubernetes cluster, so to make your applications available to the outside word, pods must be exposed as services
- A Kubernetes service groups together related pods and presents them to end users as a single entity

Role Based Access Control

- Provide granular access to resources and operations.
  - RBAC must be enabled on cluster
  - Role / Cluster Role: rules that represent a set of permissions
  - Role Binding / Cluster Role Binding: applies role to identity

https://kubernetes.io/docs/reference/access-authn-authz/rbac/

Thank You