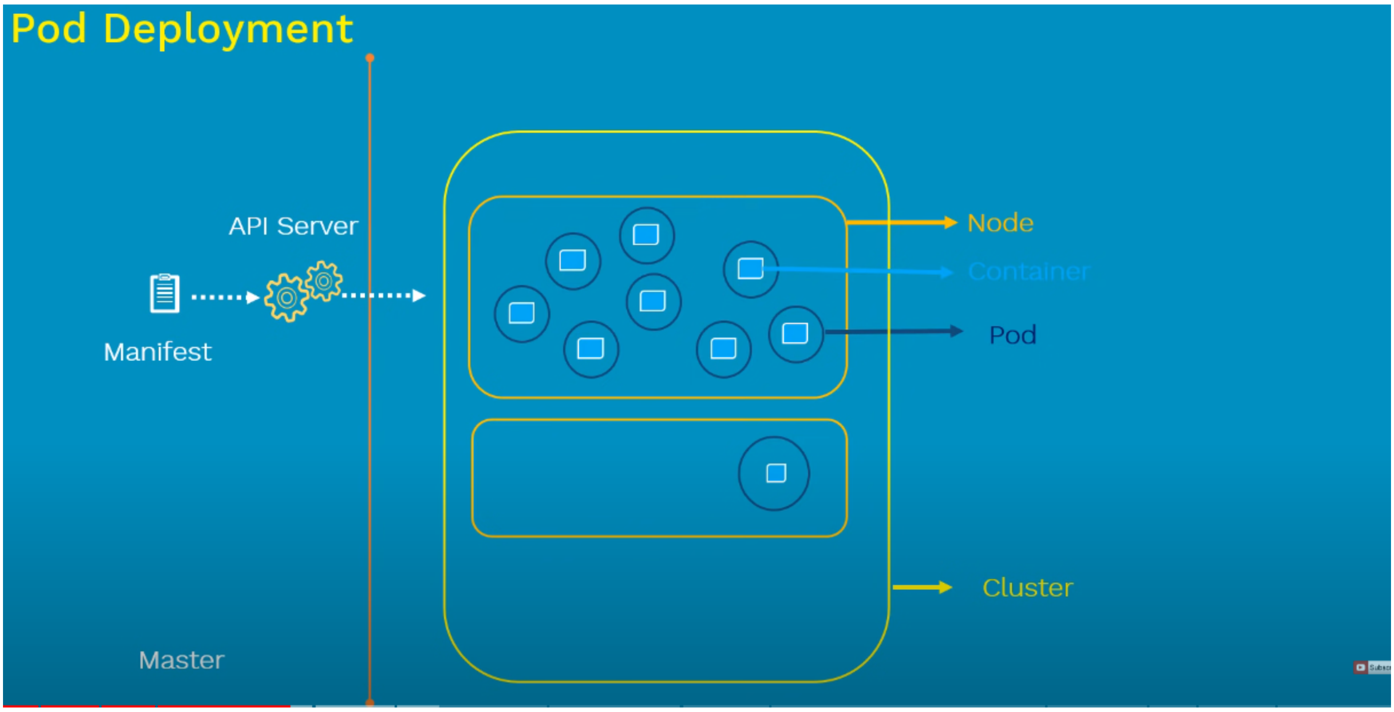
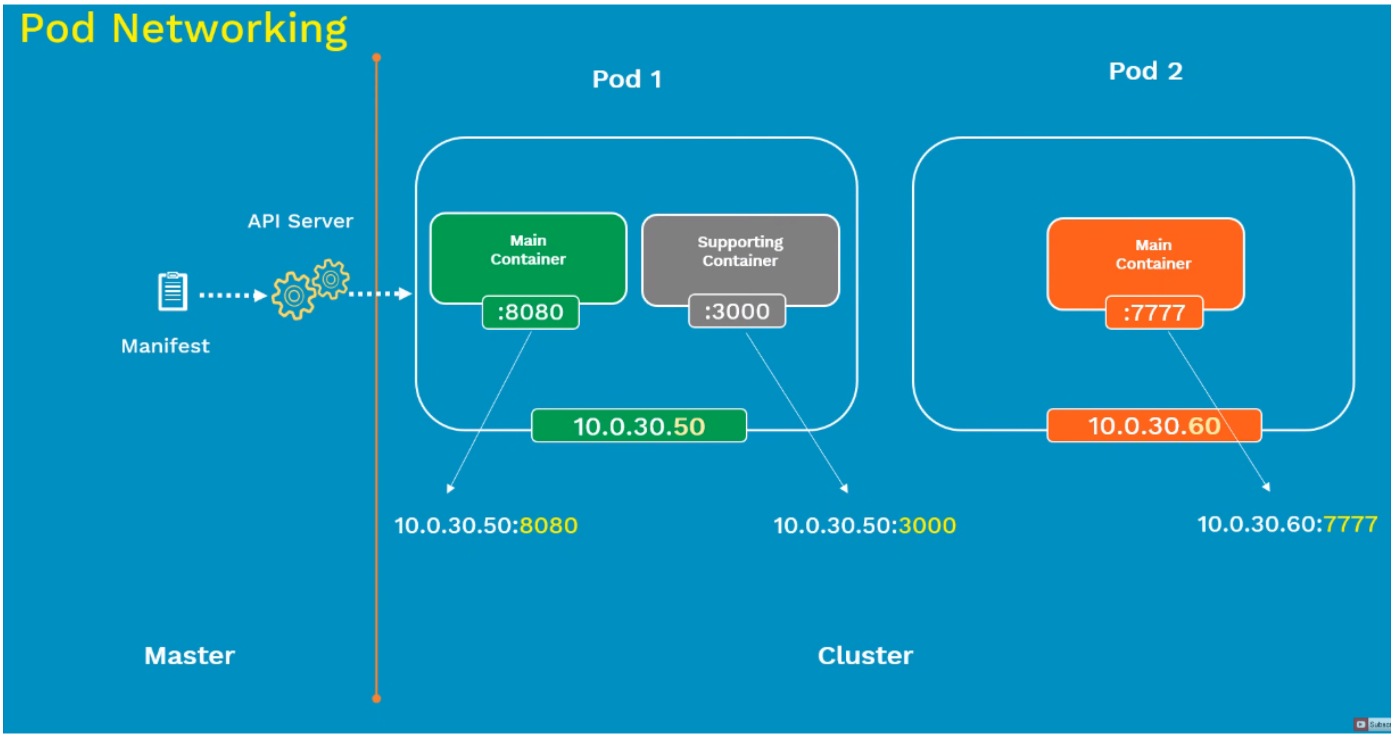
What is Pod?

- pods is the smallest unit of k8s
- Pod is rapper around one or more containers
- Pod is an abstraction layer over container
- Each worker nodes have multiple pods, then each pods contains multiple docker containers most cases single container
- Each pod has its own server and own ip addresses
- Pod can communicate with each other using ip addresses
- When a pods die, another pods create and communicate to another pod using services

Pod Deployment



Pod Networking

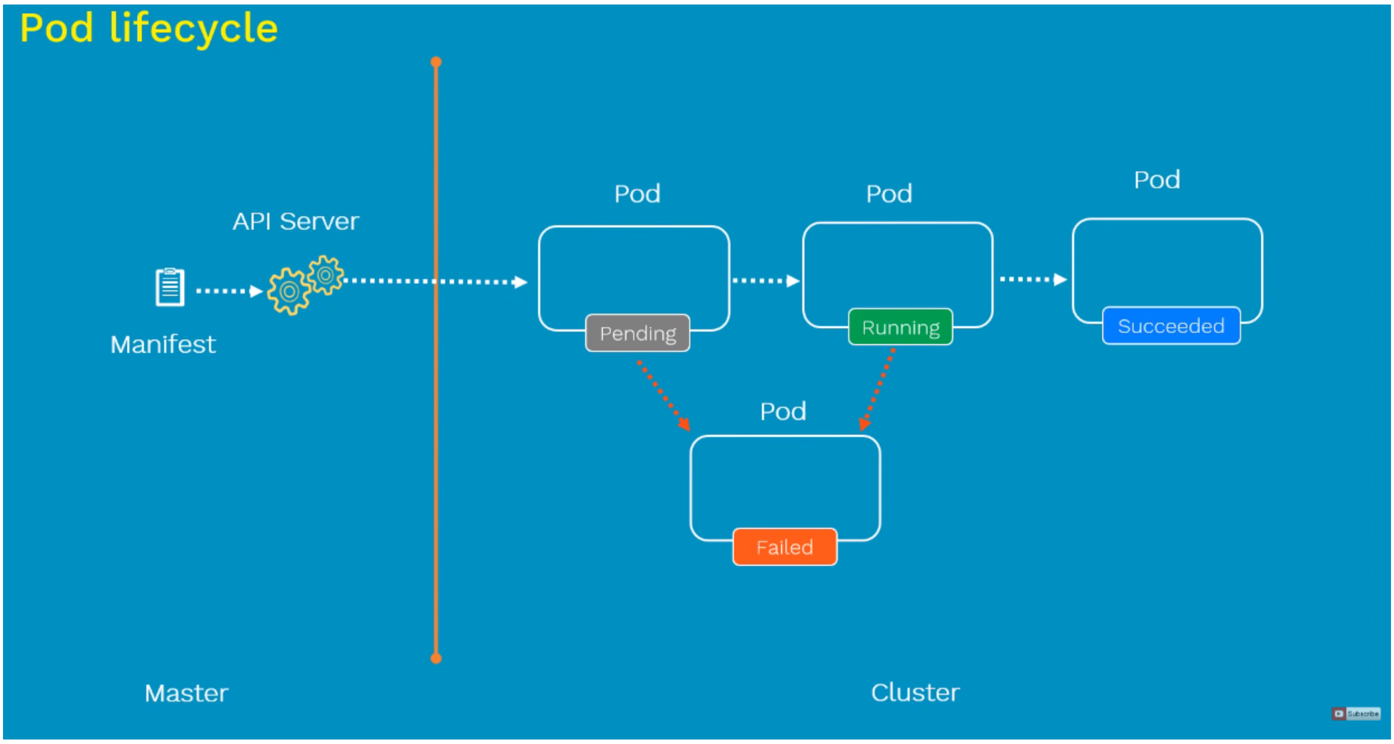


- Each pod gets its own IP address and all containers in pods run in different ports.
- The shared context of a Pod is a set of Linux namespaces, cgroups, and potentially other facets of isolation - the same things that isolate a container
- A Pod is similar to a set of containers with shared namespaces and shared file-system volumes.

Inter-Pod vs Intra pod communication

- Containers inside a Pod can communicate between each other using localhost.
- Containers that are not in the same Pod can communicate using Pods IP addresses.
- To access a container we need to expose port of the container

Pod lifecycle



- Note: once a pot dies, we can not bring him back.

Manifest File

Each manifest file has 4 common fields.

- apiVersion : defines the version number of this object
- kind : name of object
- metadata: contains 2 fields
  - name: name of the pod
  - labels: its just a tag to give this pod for future identify.
- spec : define specification of this objects

```
apiVersion: v1
kind: pod
metadata:
 name: nginx-pod
 labells:
  app: nginx
  tier: test
spec:
 containers:
  -name: nginx-container
  image: nginx
```

Create,display,delete & describe pod

```
- $ Kubectl create -f `<pod path>`
- $ Kubectl  get pods
- $ Kubectl get pod -o wide
- $ Kubectl get pod `<pod-name>` -o yaml
- $ Kubectl describe pod nginx
- $ Kubectl delete pod `<pod-name>`
- $ kubectl logs `<pod-name>` -f
```

- ## Pod Testing

- Ping `<pod IP>`
- Kubectl exec -it `<pod-name>` – bash

- ## Health Check

** ** - Liveness Probe

** ** - we add a liveness probe to our k8s container, which runs an HTTP request

against the /healthy path on our container

apiVersion: v1kind: Podmetadata:name: bookserverspec:containers:- image: anisurrahman75/book-server-api:v1.4name: bookserverlivenessProbe:httpGet:path: /api/booksport: 3030initialDelaySeconds: 5timeoutSeconds: 1periodSeconds: 10failureThreshold: 3ports:- containerPort: 3030name: httpprotocol: TCP``

- kubectl port-forward bookserver 3030:3030
- Kubectl logs -f bookserver

- # Persist Volume

- Used when applications need to access to underlying host file system
- LocalHost:

apiVersion: v1kind: Podmetadata:name: bookservervolumespec:volumes:- name: "bookservervolume"hostPath:path: "/var/log"containers:- image: anisurrahman75/book-server-api:v1.4name: bookservervolumevolumeMounts:- mountPath: "/var/log"name: "bookservervolume"resources:requests:cpu: "500m"memory: "128Mi"limits:cpu: "1000m"memory: "256Mi"ports:- containerPort: 3030name: httpprotocol: TCP``

**