**# Docker Compose**

sudo apt update && sudo apt upgrade -y

sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io -y

sudo systemctl status docker

sudo docker run hello-world

sudo usermod -aG docker $USER

newgrp docker

sudo curl -L "https://github.com/docker/compose/releases/download/v2.17.3/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

**# Kubernetes**

curl -LO "https://dl.k8s.io/release/v1.30.3/bin/linux/amd64/kubectl"

curl -LO "https://dl.k8s.io/release/v1.30.3/bin/linux/amd64/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

kubectl version --client --output=yaml

curl -LO https://github.com/kubernetes/minikube/releases/v1.33.1/download/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64

minikube start

kubectl version

kubectl config current-context

minikube stop

minikube delete

minikube start

# Run image using Kubernetes

kubectl run nginx --image=nginx

Explanation:

kubectl: The Kubernetes command-line tool used to interact with the cluster.

run: Creates a new pod in the cluster.

nginx: The name assigned to the pod.

--image=nginx: Specifies the container image to use for the pod. This will pull the official Nginx image from Docker Hub and run it inside the pod.

# Pod is created or Not

kubectl get pods

# To know more information

kubectl describe <RESOURCE> <NAME>

--> Resource can be "POD, SERVICE & Deployment"

--> Name is the name of the objects

kubectl describe pod nginx

# For logs

kubectl logs <Name of the Pod>

kubectl logs nginx

# When a pod has more than 1 containers

kubectl logs nginx --container=nginx

# kubectl delete <RESOURCE> <NAME>

kubectl delete pod nginx

or

kubectl delete pod --force=true nginx

# Exposing pod as a service so that the pod can access using it's name

kubectl expose pod nginx --type=NodePort --port=80

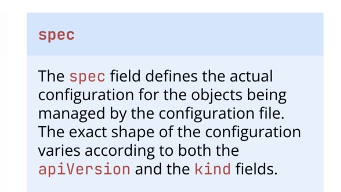
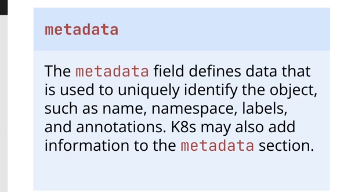
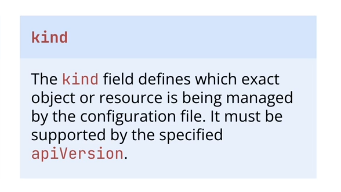
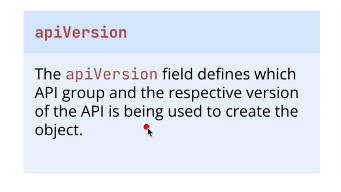
kubectl get service

# For install curl

apk --update add curl

### K8s MAnifest File





# # Imperative Commands

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

labels:

name: nginx

spec:

containers:

- name: nginx-container

image: nginx:1.27.0

ports:

- containerPort: 80

kubectl create -f nginx-pod.yaml

## While expose port use Service and during Service doesn’t need **containers** in the **spec**.

nginx-service.yaml

# For documentation, equivalent to the imperative command;

# kubectl expose pod ngin-pod --type=NodePort --port=80

apiVersion: v1

kind: Service

metadata:

name: nginx-svc

labels:

name: nginx

spec:

type: NodePort

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

name: nginx # Service find out the pod which label is nginx

# For run the alpine image

kubectl run -it alpine --image=alpine:3.20 sh

# Generating Kubernetes Manifests with kubectl

kubectl run hello --image=hello-world --dry-run=client -o yaml

# Choose output of the pod in yaml is

kubectl get pod nginx-pod -o yaml

# During update the yaml file have to use the declarative method

kubectl apply -f nginx-pod.yaml

# Declarative apply method also work on the full directory

kubectl apply -f “directory name”

# For check any modification in the yaml file have to use

kubectl diff -f “file name.yaml” → It’s just check the file modification or update then you can run the “kubectl apply -f “file\_name.yaml” for update the pod with the modification.

# Multiple object can define within a same file. For it, have to use “---” inside the both object

Example:

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

labels:

app: nginx # Label for the pod

spec:

containers:

- name: nginx-container

image: nginx:1.27.0 # Using nginx version 1.27.0

ports:

- containerPort: 80 # Exposing port 80 inside the container

---

apiVersion: v1

kind: Service

metadata:

name: nginx-svc

labels:

app: nginx # Label for the service

spec:

type: NodePort # Exposes the service externally on a NodePort

ports:

- port: 80 # Service port

protocol: TCP

targetPort: 80 # Port inside the pod to route traffic to

selector:

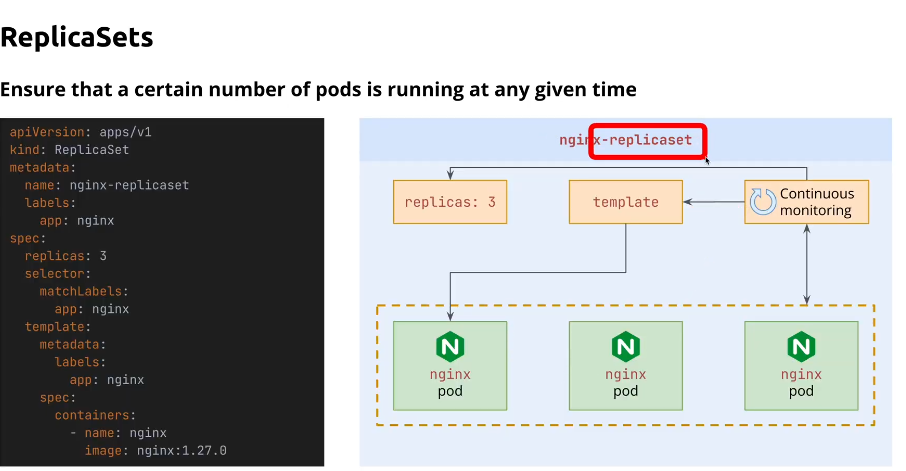
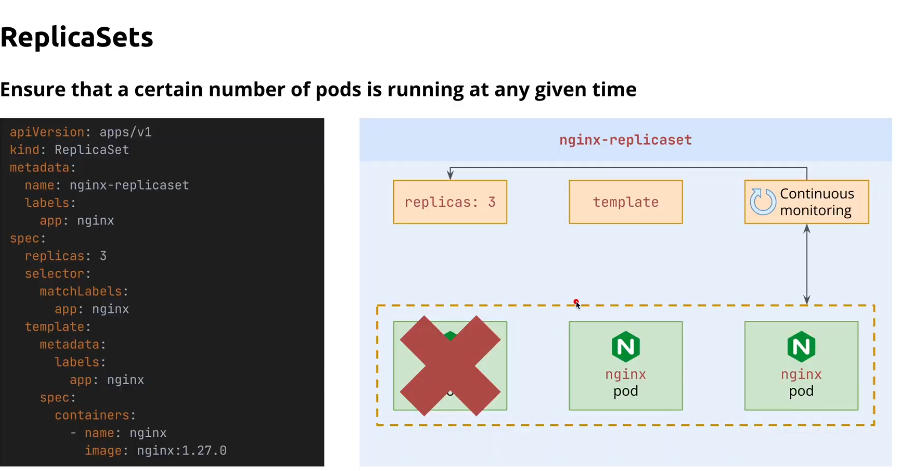
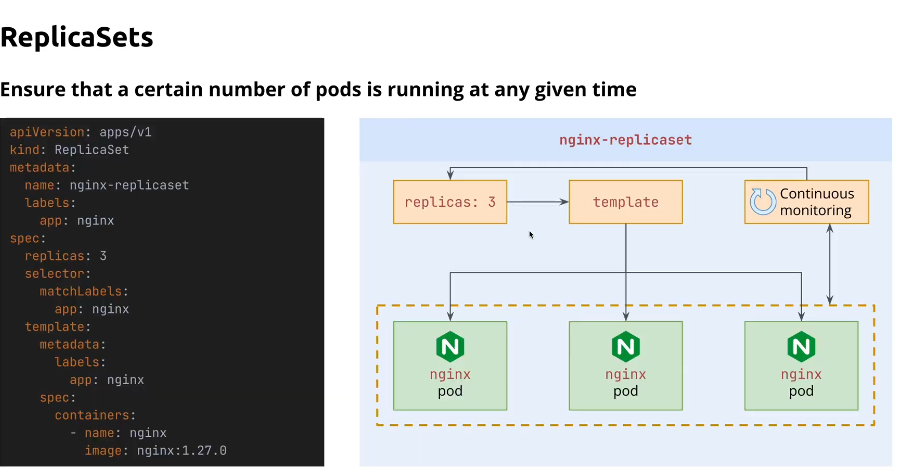
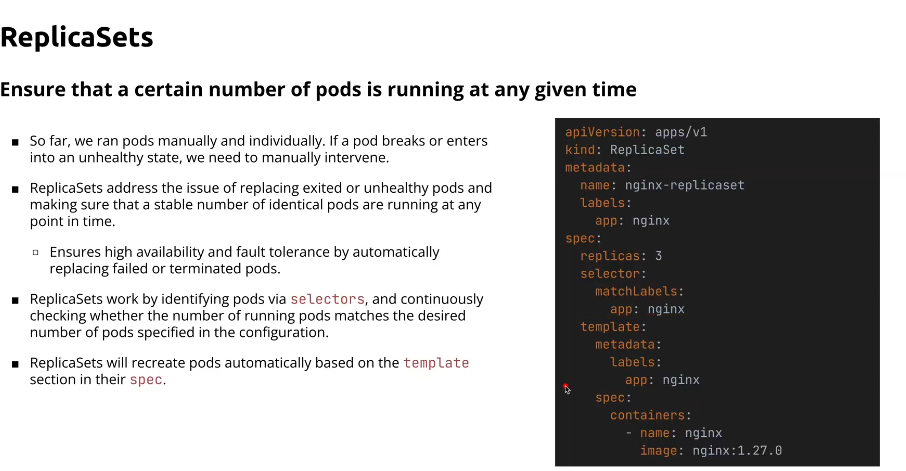
app: nginx # This must match the label in the Pod metadata

kubectl apply -f file-name.yaml

**# Kubernetes ReplicaSets and Deployments**

### **ReplicaSets in Kubernetes**

A **ReplicaSet (RS)** in Kubernetes is a controller that ensures a specified number of identical **pod replicas** are running at all times. If a pod fails, the ReplicaSet automatically creates a new one to maintain the desired state.



apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx-replicaset # Name of the ReplicaSet

spec:

replicas: 3 # Ensures 3 pods are always running

selector:

matchLabels:

app: nginx # Must match the pod's labels below

template:

metadata:

labels:

app: nginx # Labels applied to the pod for matching with the ReplicaSet

spec:

containers:

- name: nginx # Container name

image: nginx:1.27.0 # Nginx image version 1.27.0

ports:

- containerPort: 80 # Exposing port 80 inside the container

kubectl apply -f file-name.yaml

# Delete a specific pod

kubectl delete pod “pod name”

# Describe the replicase

kubectl describe rs nginx-replicaset

# For watching the live like nvidia-smi of pod, replicaset

kubectl get pod --watch

kubectl get rs --watch

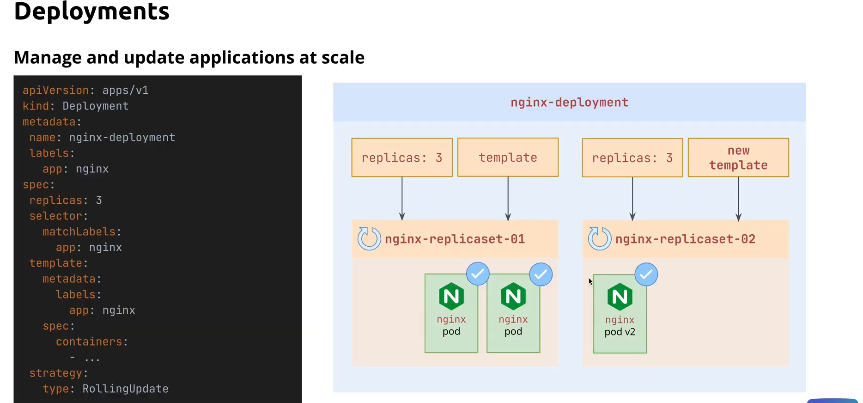
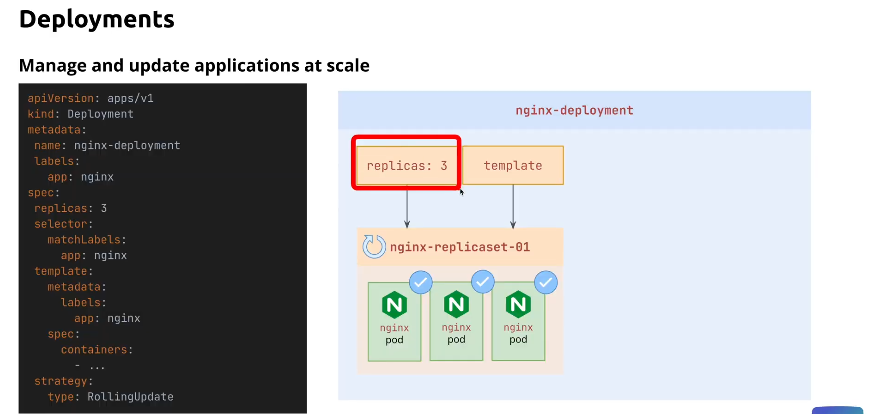
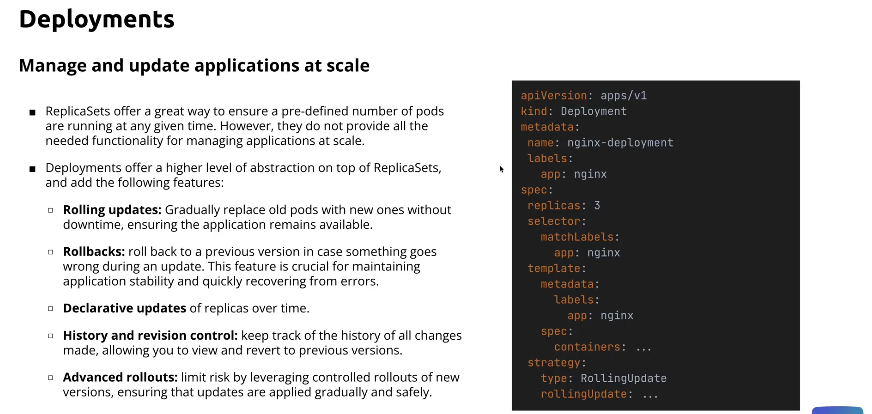
# For cheking the image of a pod

kubectl describe pod “pod name” | grep Image

# For replicaset description

kubectl describe rs “name of the replicaset from the metadata” which is nginx-replicaset

**Deployments**



apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 5

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.27.0

ports:

- containerPort: 80

strategy:

type: RollingUpdate

kubectl describe deploy nginx-deployment

**Rollout in K8s**

kubectl rollout history deployment nginx-deployment

# Go back previous deployment

kubectl rollout undo deployment nginx-deployment

# For revision history

kubectl rollout history deployment nginx-deployment --revision=2 -o yaml

# Add annotations for tracking the changes

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

annotations:

"kubernetes.io/change-cause": "Update nginx:1.27.0 to nginx:1.27.0-alpine"

spec:

replicas: 5

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.27.0-alpine

ports:

- containerPort: 80

strategy:

type: RollingUpdate

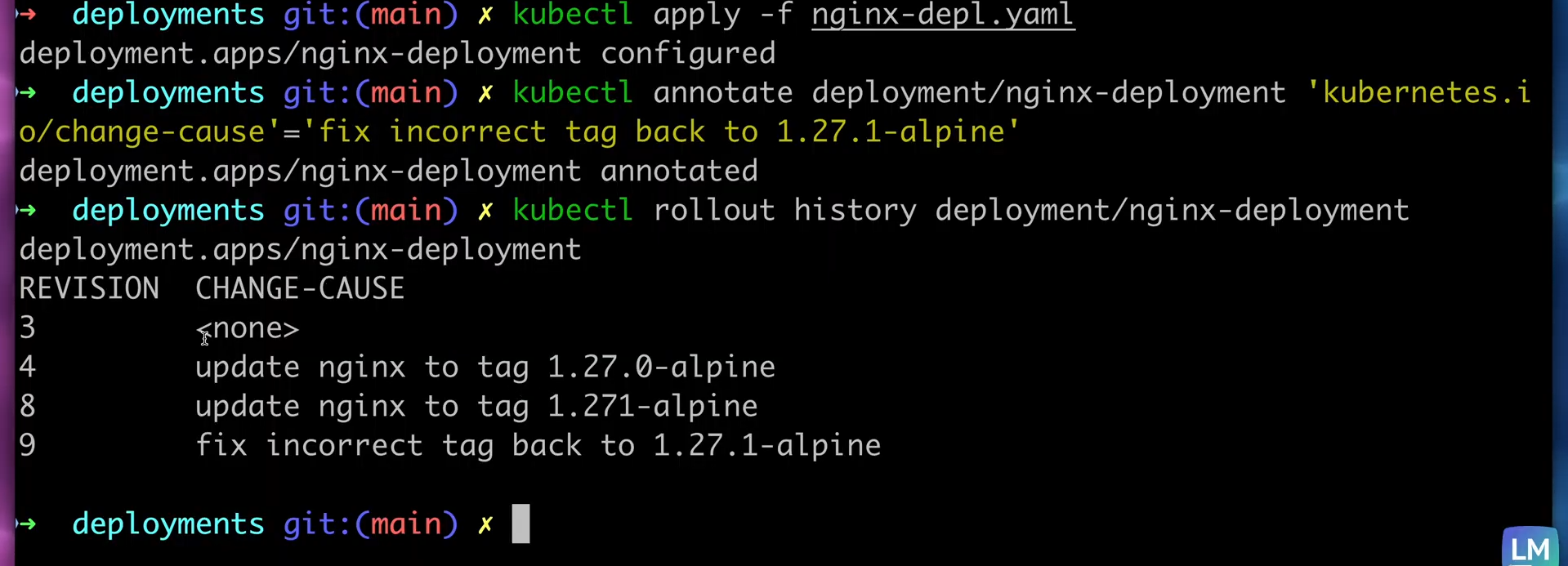
# Annotation can also provide from the command

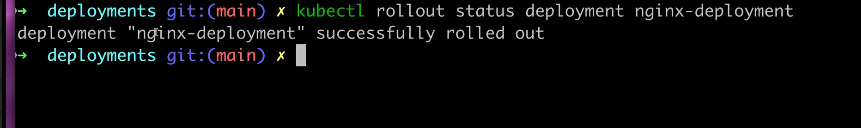
kubectl annotate deployment nginx-deployment "kubernetes.io/change-cause": "Update nginx:1.27.0 to nginx:1.27.0-alpine"

**Scaling Deployments with Kubectl**

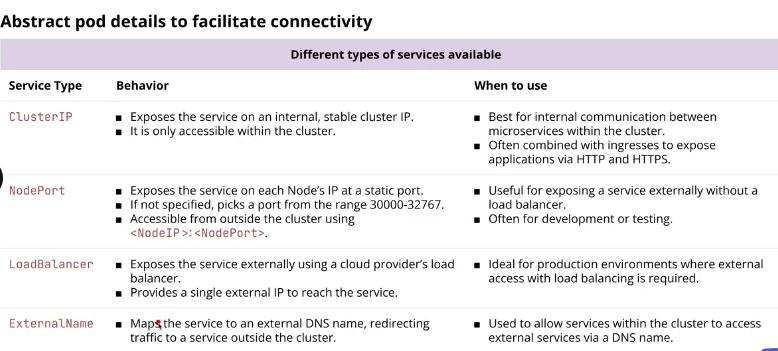
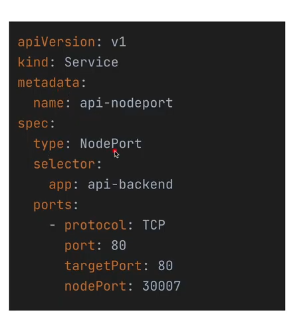
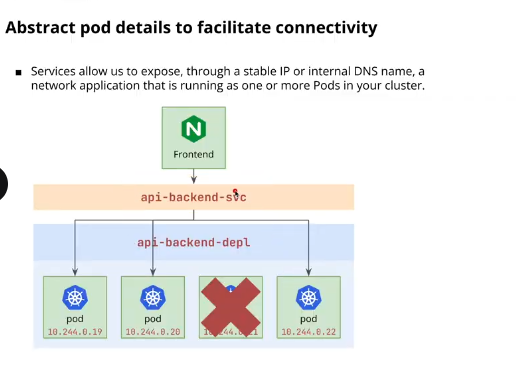
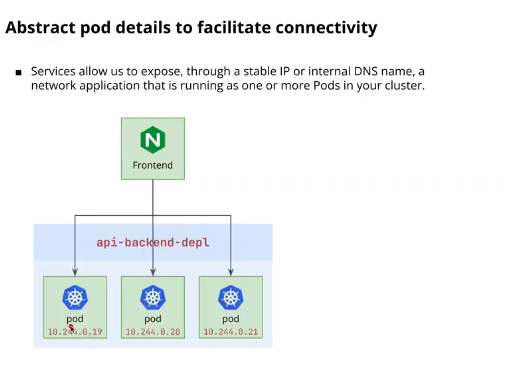
# Pod can be fluctuate using Scaling

kubectl scale deploy nginx-deployment --replicas=3



****

**Services**

****

Completed: **170**