**Basic Kubernetes Deployment Example**

In this example, we will create a simple Kubernetes Deployment that runs an NGINX web server.

**Deployment YAML for NGINX**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment # Name of the deployment

spec:

replicas: 3 # Number of desired replicas (Pods)

selector:

matchLabels:

app: nginx # This label is used to find the Pods managed by this Deployment

template:

metadata:

labels:

app: nginx # Label for identifying the Pods

spec:

containers:

- name: nginx

image: nginx:latest # NGINX container image

ports:

- containerPort: 80 # Exposing port 80 for HTTP traffic

**Explanation of the YAML:**

* **apiVersion**: Specifies the Kubernetes API version (in this case, apps/v1).
* **kind**: Defines the type of object, here it's a Deployment.
* **metadata**: Metadata about the deployment, such as the name of the deployment.
* **spec.replicas**: Defines how many copies (replicas) of the NGINX Pod you want to run.
* **spec.selector**: This section ensures that only Pods with the label app: nginx are part of this Deployment.
* **spec.template**: This template describes the Pods that should be created by the Deployment. Each Pod will have one container (nginx) running the nginx:latest image and exposing port 80.

**Create the Deployment**

Run the following command to create the deployment from the YAML file:

kubectl apply -f nginx-deployment.yaml

**Verify Deployment and Pods**

To verify the deployment was created successfully:

kubectl get deployments

This will show the status of the deployment, including how many Pods are running.

To check the status of the Pods:

kubectl get pods

**Scaling a Deployment**

One of the advantages of a Kubernetes Deployment is that it can easily scale the number of replicas (Pods) up or down.

**Scale the Deployment to 5 Replicas**

To scale the deployment to 5 replicas:

kubectl scale deployment nginx-deployment --replicas=5

This command adjusts the number of Pods running for the deployment to 5.

**Verify the Scaling**

To check that the scaling was successful, run:

kubectl get pods

You should see 5 Pods running, all with the label app=nginx.

**Rolling Update**

Kubernetes supports **Rolling Updates**, which allow you to update your application without downtime. When you modify a deployment, Kubernetes will gradually replace the old Pods with new ones.

**Update the Deployment with a New Image**

Let's update the Deployment to use a different version of NGINX (for example, NGINX 1.21).

kubectl set image deployment/nginx-deployment nginx=nginx:1.21

**Monitor the Rolling Update**

Kubernetes will perform a rolling update, replacing old Pods with new ones gradually. You can monitor the update progress with the following command:

kubectl rollout status deployment/nginx-deployment

This will show you the progress of the update.

**Rollback the Update**

If there’s an issue with the new version, you can rollback to the previous version of the Deployment using:

kubectl rollout undo deployment/nginx-deployment

This command reverts the Deployment back to the previous version.

**Expose the Deployment Using a Service**

To allow external traffic to reach your Pods, you can expose the Deployment using a **Service**.

**Service YAML**

apiVersion: v1

kind: Service

metadata:

name: nginx-service # Name of the service

spec:

selector:

app: nginx # The service will target Pods with this label

ports:

- protocol: TCP

port: 80 # Port on which the service is exposed

targetPort: 80 # Port inside the container (NGINX runs on port 80)

**Create the Service**

kubectl apply -f nginx-service.yaml

**Verify the Service**

To check that the service has been created successfully:

kubectl get services

**Port Forwarding for Local Access**

To access the service locally:

kubectl port-forward service/nginx-service 8080:80

Now you can access the NGINX service via http://localhost:8080.

**Delete the Deployment**

If you no longer need the Deployment, you can delete it with:

kubectl delete deployment nginx-deployment

This will remove the deployment and all the associated Pods.

**Summary of Important kubectl Commands:**

1. **Create or update a Deployment:**

kubectl apply -f <deployment-file.yaml>

1. **Scale a Deployment:**

kubectl scale deployment <deployment-name> --replicas=<number>

1. **Monitor Deployment status:**

kubectl rollout status deployment/<deployment-name>

1. **Roll back a Deployment:**

kubectl rollout undo deployment/<deployment-name>

1. **Get Deployment information:**

kubectl get deployments

1. **Get Pod information:**

kubectl get pods

1. **Expose a Deployment via a Service:**

kubectl expose deployment <deployment-name> --port=<port> --target-port=<container-port>

1. **Delete a Deployment:**

kubectl delete deployment <deployment-name>