## 📄 Task 4: Implement Rolling Updates

### 🎯 Objective

Configure the Deployment to use **RollingUpdate** strategy with controlled surge and availability, then trigger an image version change and observe the rollout.

### 🛠 Steps Taken

1. Edited the Deployment manifest to include the following rolling update strategy:

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 1

maxUnavailable: 0

1. Applied the updated Deployment using:  
   kubectl apply -f deployment.yaml
2. Check on the pods until updated:

kubectl get pods -l app=nodejs -w

1. Applying the updated-deployment manifest:

Kubectl apply -f updated-deployment.yaml

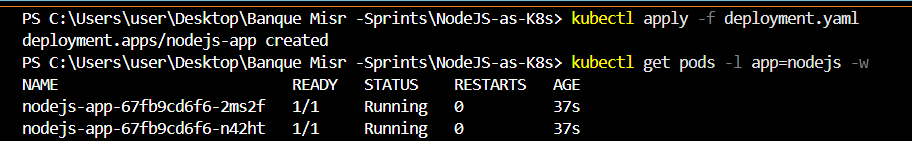
1. Monitored the rollout progress:  
   kubectl rollout status deployment/my-app
2. Described the pods to inspect rollout details:  
   kubectl describe pods -l app=nodejs
3. Performed a rollback to the previous version:  
   kubectl rollout undo deployment/nodejs-app
4. Verified the rollback via pod description:  
   kubectl describe pods -l app=nodejs

### 📸 Screenshots

• **Screenshot 1:**

kubectl apply -f deployment.yaml

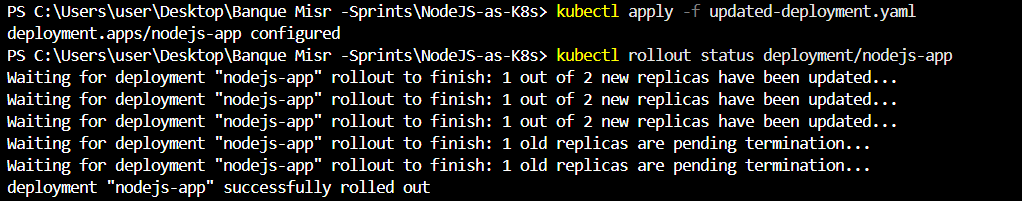
kubectl get pods -l app=nodejs -w



• **Screenshot 2:**

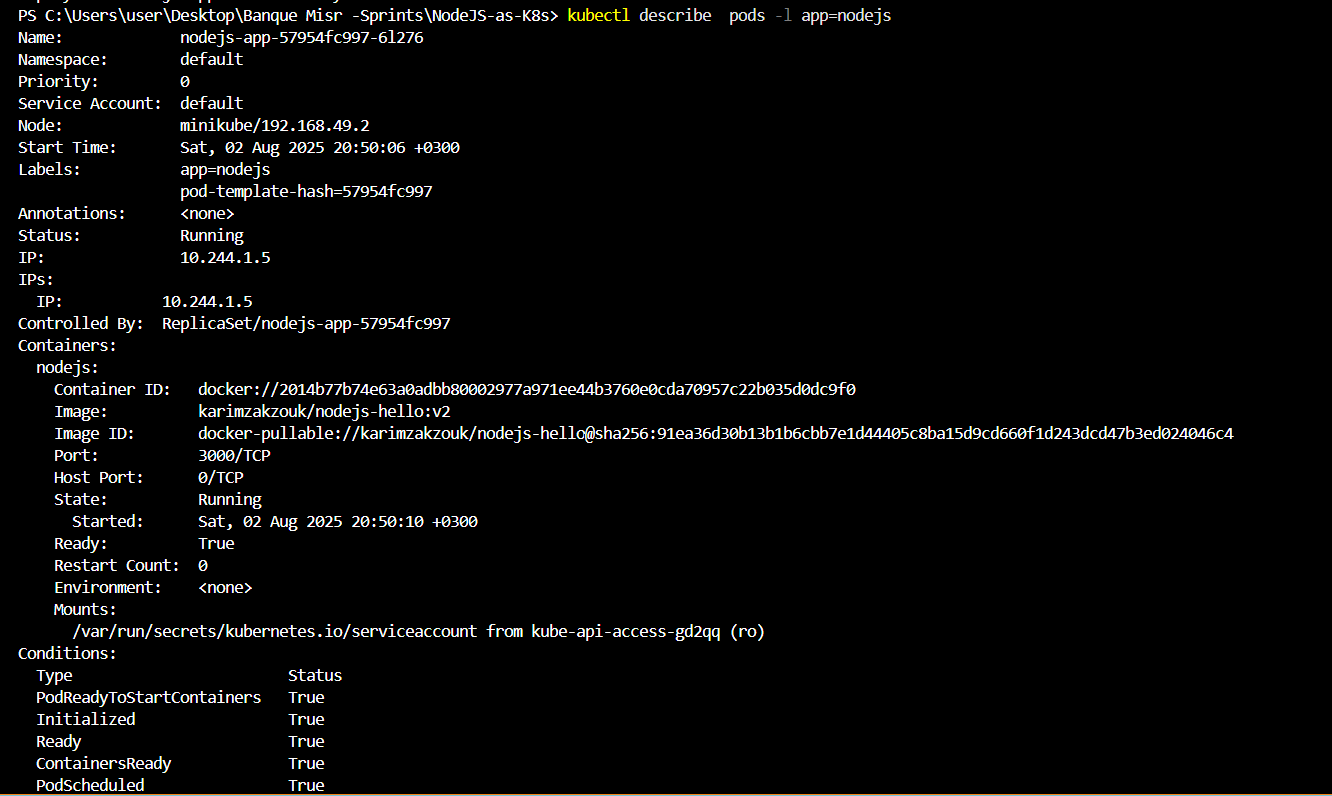
kubectl apply -f updated-deployment.yaml

kubectl rollout status deployment/my-app



• **Screenshot 3:**

kubectl describe pods -l app=nodejs



• **Screenshot 4:**

kubectl rollout undo deployment/nodejs-app



• **Screenshot 5:**

kubectl describe deploy nodejs-app

A screen shot of a computer

AI-generated content may be incorrect.

### ✅ Outcome

The Deployment used a **RollingUpdate** strategy successfully. The application was updated incrementally with zero downtime, and all pods transitioned to the new version while maintaining availability.