Kubernetes Deployment Rolling Update and Rollback Hands-On Lab

Author : Sandeep Kumar Sharma

Introduction: What is a Rolling Update?

A **Rolling Update** in Kubernetes allows you to update the version of an application (or image) without downtime. Kubernetes gradually replaces old Pods with new ones while ensuring that the application remains available throughout the update process.

Similarly, **Rollback** helps you revert to a previous version if something goes wrong during the update.

Objective:

By the end of this lab, participants will learn: - How to perform a rolling update for a Deployment. - How to verify and monitor the rollout process. - How to perform a rollback to a previous version.

Part 1: Create an Initial Deployment

Step 1 — Create Deployment YAML

Create a file named `rolling-deploy.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rolling-deploy-demo
  labels:
    app: rolling-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: rolling-demo
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
```

```
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: rolling-demo
    spec:
      containers:
      - name: nginx
        image: nginx:1.19
        ports:
        - containerPort: 80
```

Step 2 — Apply the Deployment

```
kubectl apply -f rolling-deploy.yaml
```

Step 3 — Verify the Deployment

```
kubectl get deployments
kubectl get pods -l app=rolling-demo
```

All Pods should be running with `nginx:1.19`.

Part 2: Perform a Rolling Update

Step 1 — Update the Deployment Image

Use the `set image` command to update the container image version:

```
kubectl set image deployment/rolling-deploy-demo nginx=nginx:1.25 --record
```

Step 2 — Check the Rollout Status

```
kubectl rollout status deployment/rolling-deploy-demo
```

This command shows the progress of the update.

Step 3 — Verify New Pods

```
kubectl get pods -o wide
```

You should see new Pods being created with the updated `nginx:1.25` image while old ones terminate gradually.

Step 4 — View Deployment History

```
kubectl rollout history deployment/rolling-deploy-demo
```

This will display the revision numbers and change causes (if `--record` was used).

Expected output example:

```
deployment.apps/rolling-deploy-demo
REVISION  CHANGE-CAUSE
1         kubectl apply -f rolling-deploy.yaml
2         kubectl set image deployment/rolling-deploy-demo nginx=nginx:1.25 --
record
```

Part 3: Simulate a Bad Update and Rollback

Step 1 — Perform a Faulty Update

Let's assume a wrong image version is applied:

```
kubectl set image deployment/rolling-deploy-demo nginx=nginx:broken-version --
record
```

Step 2 — Monitor the Rollout

```
kubectl rollout status deployment/rolling-deploy-demo
```

The rollout may hang or show errors because the image does not exist.

Step 3 — Verify Failed Pods

```
kubectl get pods
kubectl describe pod <pod-name>
```

You'll notice image pull errors.

Step 4 — Roll Back to Previous Version

```
kubectl rollout undo deployment/rolling-deploy-demo
```

This restores the Deployment to the previous working version.

Step 5 — Confirm Rollback

```
kubectl rollout history deployment/rolling-deploy-demo
kubectl get pods -o wide
```

You should again see `nginx:1.25` running successfully.

Part 4: Scale the Deployment (Post Rollback)

Step 1 — Scale the Deployment to 5 Replicas

```
kubectl scale deployment rolling-deploy-demo --replicas=5
```

Step 2 — Verify Scaling

```
kubectl get pods -l app=rolling-demo
```

You'll now have 5 running Pods with the stable image version.

Part 5: Clean Up Resources

```
kubectl delete deployment rolling-deploy-demo
```

Verify:

```
kubectl get pods
```

All Pods should be terminated.

Verification Summary

Task	Command	Description
Create Deployment	`kubectl apply -f rolling-deploy.yaml`	Creates base Deployment
Update Image	`kubectl set image deployment/...`	Starts rolling update
Monitor Rollout	`kubectl rollout status deployment/...`	Shows update progress
View History	`kubectl rollout history deployment/...`	Lists revisions
Rollback	`kubectl rollout undo deployment/...`	Reverts to previous version
Scale	`kubectl scale deployment/... --replicas=n`	Changes replica count

✅**Outcome:** After completing this lab, participants will understand the end-to-end process of performing rolling updates and rollbacks in Kubernetes Deployments. They'll be confident in maintaining zero-downtime application upgrades while ensuring rollback safety.

Next Step: Move on to the **Kubernetes Deployment Strategies Lab** to explore Blue-Green and Canary Deployment methods for production environments.