ROLLBACK IN KUBERNETES

1. Introduction to Kubernetes Rollbacks

When you deploy an application in Kubernetes using a **Deployment**, Kubernetes maintains a **revision history** of the Deployment's states. Each time you update a Deployment, Kubernetes creates a new revision. If something goes wrong after an update, you can easily roll back to a previous revision.

Key Concepts:

- **Revisions**: Each change to a Deployment is stored as a revision. Kubernetes keeps track of these revisions, and you can roll back to any previous one.
- **Rollback**: The process of reverting to a previous state of the Deployment.
- **Rolling Updates**: Kubernetes updates the deployment in a controlled manner, one pod at a time, to minimize downtime.

- **Initial Deployment: Nginx v1.14.2:** The initial deployment of the application running with Nginx version 1.14.2, marking a stable baseline for operations.
- **Update Deployment: Nginx v1.16.1:** An update is deployed with Nginx version 1.16.1. This stage involves replacing the current version to introduce new features and improvements.
- **Issue Detected: Problem Found:** Post-deployment issues are detected in the new version, prompting the need for immediate investigation and troubleshooting.
- **Rollback: Revert to v1.14.2:** Due to critical issues, a rollback is performed to restore the system to the previously stable version (v1.14.2).
- **Stable Deployment: Nginx v1.14.2:** After the rollback, the system returns to a stable state running on the older but reliable Nginx version.

2. Example: Rolling Back a Deployment

Let's walk through an example where we create a deployment, update it, and then roll back to the previous version.

Step 1: Create an Initial Deployment

We will first create a simple Deployment using **Nginx** as an example.

```
# nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2  # Initial Version of Nginx
        ports:
        - containerPort: 80
```

```
# nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer  # Exposes the service externally with a public IP
  selector:
    app: nginx  # The label selector to match the pods
  ports:
    - protocol: TCP
      port: 80        # The port the service will expose
      targetPort: 80   # The port on the pod where the Nginx container is running
```

Deploy this file to your cluster:

```
kubectl apply -f nginx-deployment.yaml
```

Check the status of your Deployment:

```
kubectl get deployments
```

You should see that the Deployment is running with 3 replicas of Nginx version 1.14.2.

Step 2: Update the Deployment

Next, update the Deployment by changing the Nginx image to a newer version (1.16.1). Modify the deployment YAML to:

nginx-deployment-update.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16.1  # Updated Version of Nginx
        ports:
        - containerPort: 80
```

Apply the update:

```
kubectl apply -f nginx-deployment-update.yaml
```

Step 3: Check Deployment History

You can check the rollout history of the Deployment using the following command:

kubectl rollout history deployment/nginx-deployment

The output will show the revisions:

deployment.apps/nginx-deployment

REVISION CHANGE-CAUSE

1 <none>

2 <none>

This shows that the Deployment has two revisions: 1 for the initial Nginx version 1.14.2 and 2 for the updated Nginx version 1.16.1.

Step 4: Rollback to Previous Version

Suppose you discover that version 1.16.1 of Nginx has issues, and you want to roll back to version 1.14.2. To roll back to the previous revision (in this case, revision 1), run the following command:

kubectl rollout undo deployment/nginx-deployment

This will revert the Deployment to the previous working state.

Step 5: Verify the Rollback

You can verify that the rollback was successful by checking the Deployment status and the image version running in the pods:

kubectl get deployments

kubectl describe deployment nginx-deployment

You should see that the Deployment has reverted to using the Nginx image 1.14.2. Additionally, you can inspect the rollout history again:

kubectl rollout history deployment/nginx-deployment

You should see the following:

deployment.apps/nginx-deployment

REVISION CHANGE-CAUSE

1 <none>

2 <none>

3 <none>

This indicates that the Deployment has now reverted to the original revision (revision 1), but a new revision (revision 3) has been created as a result of the rollback.

3. Rollback to a Specific Revision

If you want to roll back to a specific revision (other than the immediate previous one), you can specify the revision number:

```
kubectl rollout undo deployment/nginx-deployment --to-revision=1
```

This command explicitly rolls back the Deployment to revision 1.

4. Rollback with Annotations (Tracking Changes)

When deploying new versions, you can add annotations to track the reason for the change. This helps in understanding why each revision was made.

To add an annotation, update the deployment as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  annotations:
    kubernetes.io/change-cause: "Updated to Nginx 1.16.1"
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16.1
        ports:
        - containerPort: 80
```

Apply this updated deployment:

kubectl apply -f nginx-deployment.yaml

When you check the rollout history, the annotation will be visible:

kubectl rollout history deployment/nginx-deployment

The output will show the change cause:

deployment.apps/nginx-deployment

REVISION CHANGE-CAUSE

1 <none>

2 Updated to Nginx 1.16.1

5. Managing Rollbacks in Production

Here are some best practices to manage rollbacks effectively in production:

5.1. Watch Rollout Status

After updating or rolling back a Deployment, you should monitor its status to ensure that all replicas are updated or rolled back successfully. Use the following command:

kubectl rollout status deployment/nginx-deployment

This will continuously display the status of the rollout until it is complete.

5.2. Pause a Deployment Rollout

If you want to pause a Deployment during the update process (to check if the new version is working as expected), you can pause the rollout:

kubectl rollout pause deployment/nginx-deployment

To resume the rollout, use:

kubectl rollout resume deployment/nginx-deployment

6. Summary of Rollback Commands

Command	Description
kubectl rollout undo deployment/<deployment-name>	Roll back to the previous revision.
kubectl rollout undo deployment/<deployment-name> --to-revision=<revision>	Roll back to a specific revision.
kubectl rollout history deployment/<deployment-name>	View the history of revisions.
kubectl rollout pause deployment/<deployment-name>	Pause the current rollout.
kubectl rollout resume deployment/<deployment-name>	Resume a paused rollout.
kubectl rollout status deployment/<deployment-name>	Check the status of the rollout.