Lesson 03 Demo 08

Managing Kubernetes Deployments and Version Control

Objective: To create a Kubernetes deployment, upgrade the image version, and revert to the previous version, demonstrating effective version control within a Kubernetes cluster

Tools required: kubeadm, kubectl, kubelet, and containerd

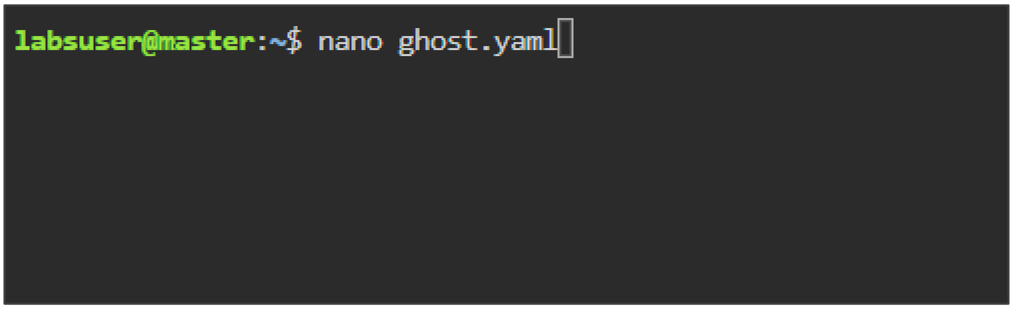
Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:

1. Create a Kubernetes deployment
2. Upgrade the image version
3. Switch back to the previous version

Step 1: Create a Kubernetes deployment

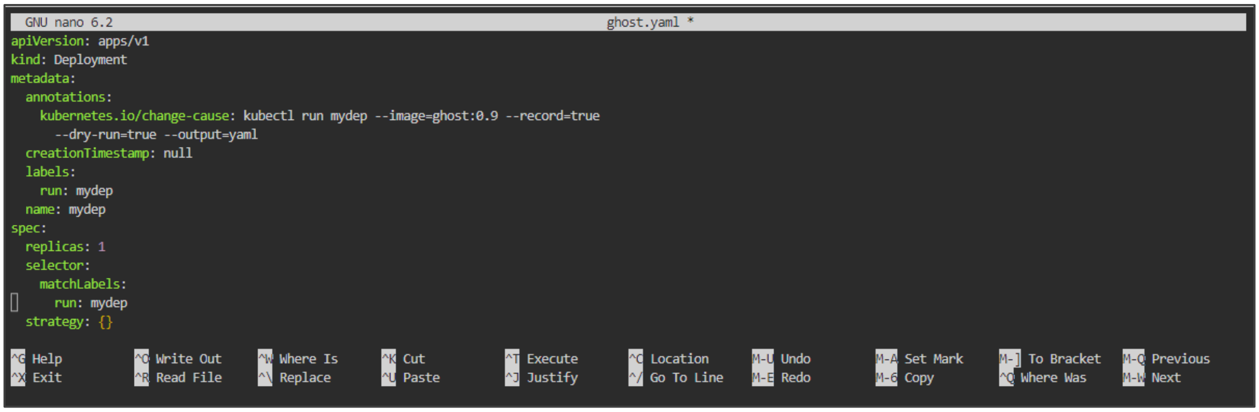
1.1 Create a YAML file using the following command:
nano ghost.yaml



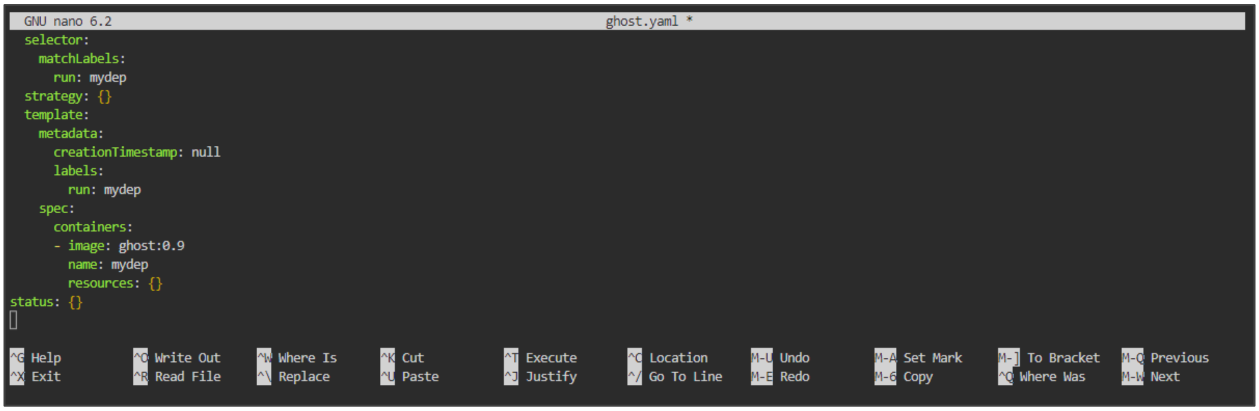
```
labsuser@master:~$ nano ghost.yaml
```

1.2 Add the following code to the **ghost.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    kubernetes.io/change-cause: kubectl run mydep --image=ghost:0.9 --record=true
      --dry-run=true --output=yaml
  creationTimestamp: null
  labels:
    run: mydep
  name: mydep
spec:
  replicas: 1
  selector:
    matchLabels:
      run: mydep
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        run: mydep
    spec:
      containers:
      - image: ghost:0.9
        name: mydep
        resources: {}
status: {}
```



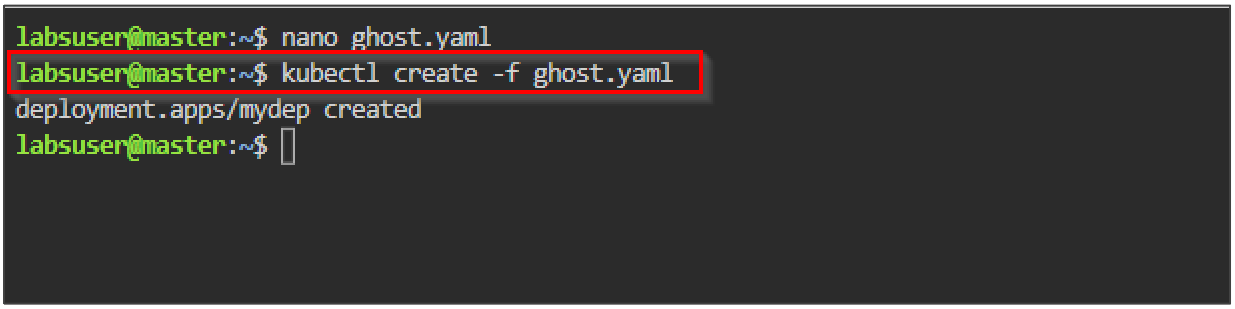
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    kubernetes.io/change-cause: kubectl run mydep --image=ghost:0.9 --record=true
      --dry-run=true --output=yaml
  creationTimestamp: null
  labels:
    run: mydep
  name: mydep
spec:
  replicas: 1
  selector:
    matchLabels:
      run: mydep
  strategy: {}
```



```yaml
  selector:
    matchLabels:
      run: mydep
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        run: mydep
    spec:
      containers:
      - image: ghost:0.9
        name: mydep
        resources: {}
status: {}
```

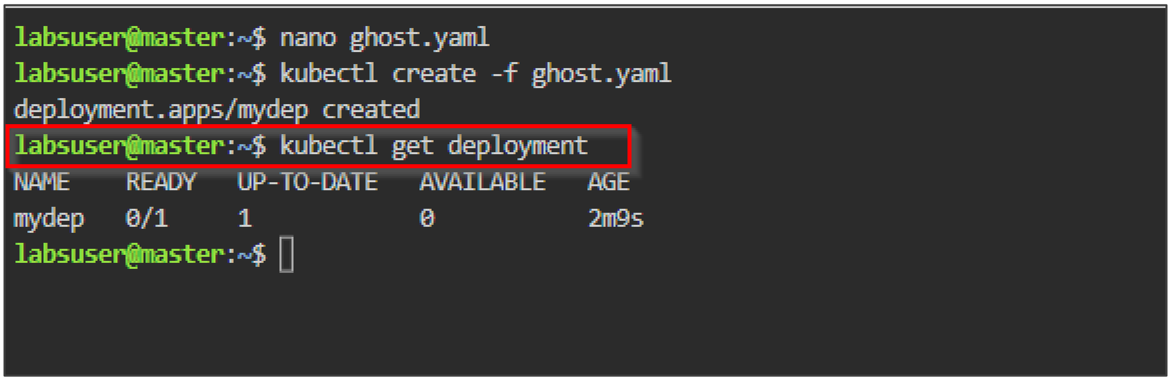
1.3 Create the deployment resource using the following command:

kubectl create -f ghost.yaml



```
labsuser@master:~$ nano ghost.yaml
labsuser@master:~$ kubectl create -f ghost.yaml
deployment.apps/mydep created
labsuser@master:~$ 
```

1.4 Verify the deployment using the following command:
kubectl get deployment

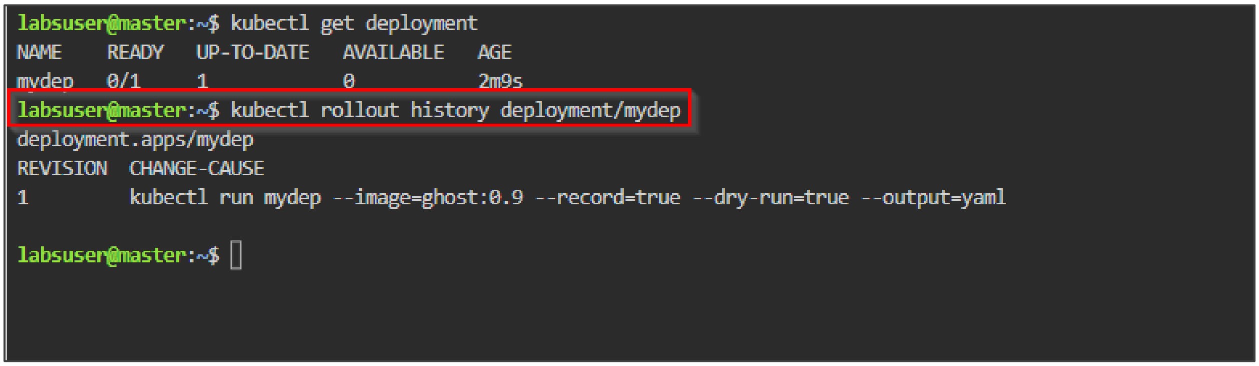


```
labsuser@master:~$ nano ghost.yaml
labsuser@master:~$ kubectl create -f ghost.yaml
deployment.apps/mydep created
labsuser@master:~$ kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mydep   0/1     1            0           2m9s
labsuser@master:~$ 
```

The deployment is successfully created.

Step 2: Upgrade the image version

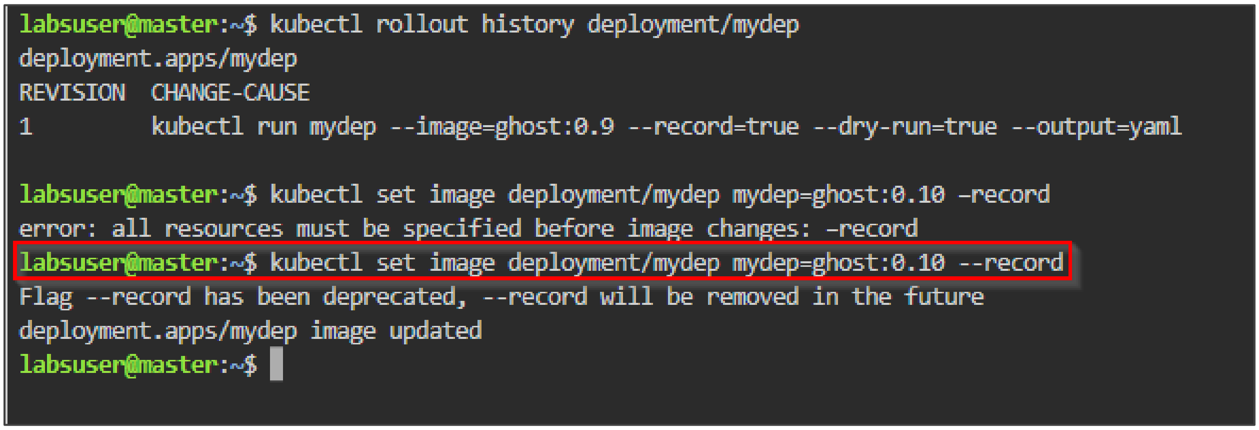
2.1 Verify the deployment rollout history using the following command:
kubectl rollout history deployment/mydep



```
labsuser@master:~$ kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mydep   0/1     1            0           2m9s
labsuser@master:~$ kubectl rollout history deployment/mydep
deployment.apps/mydep
REVISION  CHANGE-CAUSE
1         kubectl run mydep --image=ghost:0.9 --record=true --dry-run=true --output=yaml

labsuser@master:~$ 
```

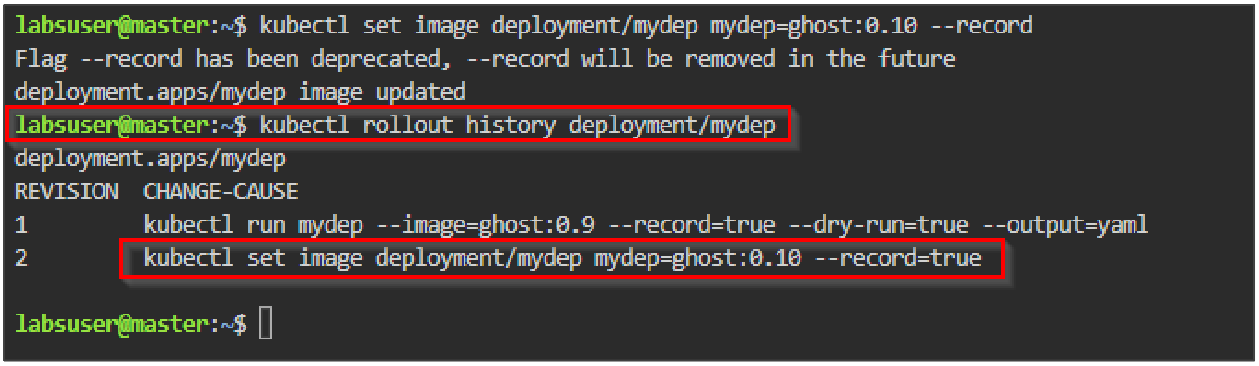
2.2 Upgrade the deployment image version to **0.10** using the following command:
kubectl set image deployment/mydep mydep=ghost:0.10 --record



```
labsuser@master:~$ kubectl rollout history deployment/mydep
deployment.apps/mydep
REVISION  CHANGE-CAUSE
1         kubectl run mydep --image=ghost:0.9 --record=true --dry-run=true --output=yaml

labsuser@master:~$ kubectl set image deployment/mydep mydep=ghost:0.10 -record
error: all resources must be specified before image changes: -record
labsuser@master:~$ kubectl set image deployment/mydep mydep=ghost:0.10 --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/mydep image updated
labsuser@master:~$
```

2.3 Verify the deployment rollout history using the following command:
kubectl rollout history deployment/mydep



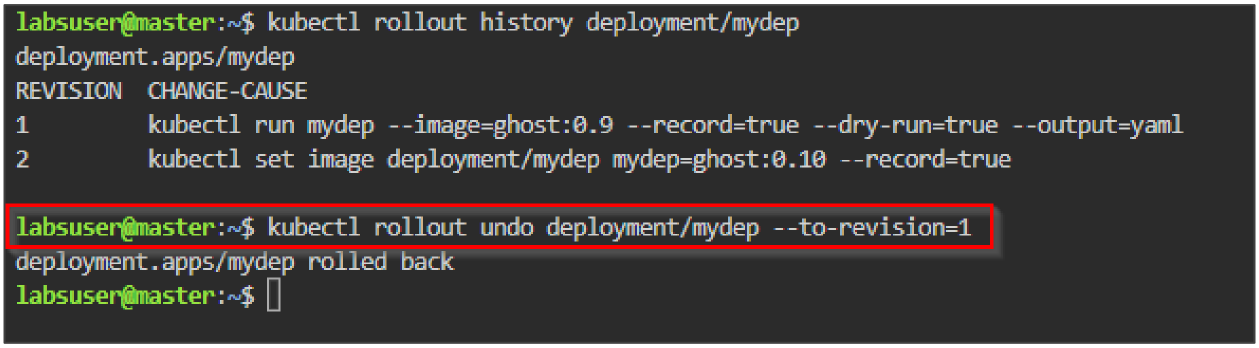
```
labsuser@master:~$ kubectl set image deployment/mydep mydep=ghost:0.10 --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/mydep image updated
labsuser@master:~$ kubectl rollout history deployment/mydep
deployment.apps/mydep
REVISION  CHANGE-CAUSE
1         kubectl run mydep --image=ghost:0.9 --record=true --dry-run=true --output=yaml
2         kubectl set image deployment/mydep mydep=ghost:0.10 --record=true

labsuser@master:~$
```

The image version of the deployment is upgraded to **0.10**.

Step 3: Switch back to the previous version

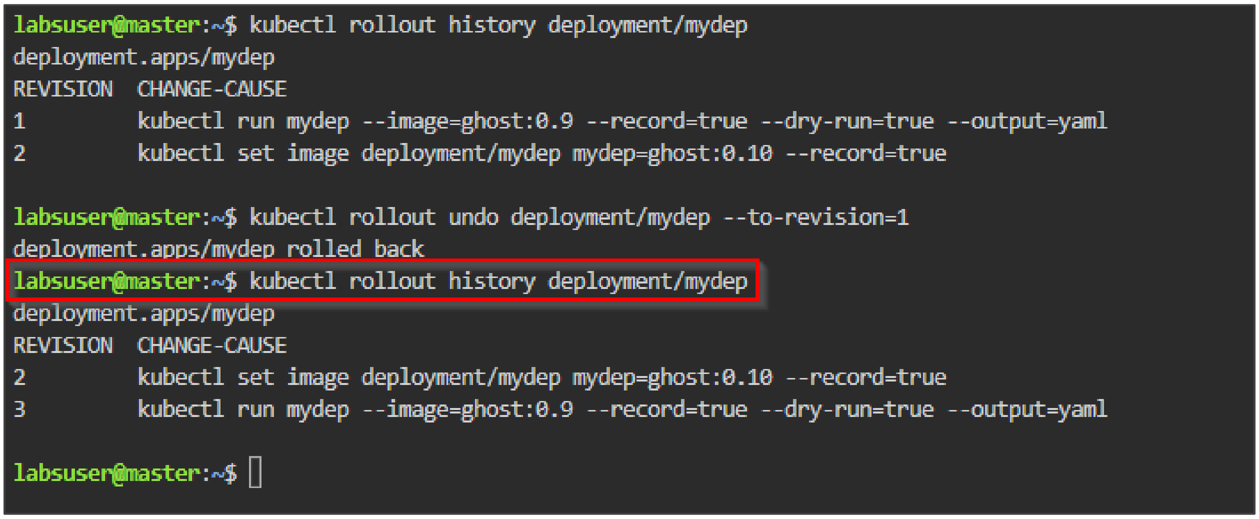
3.1 Execute the following command to revert to the initial version of deployment:
kubectl rollout undo deployment/mydep --to-revision=1



```
labsuser@master:~$ kubectl rollout history deployment/mydep
deployment.apps/mydep
REVISION  CHANGE-CAUSE
1         kubectl run mydep --image=ghost:0.9 --record=true --dry-run=true --output=yaml
2         kubectl set image deployment/mydep mydep=ghost:0.10 --record=true

labsuser@master:~$ kubectl rollout undo deployment/mydep --to-revision=1
deployment.apps/mydep rolled back
labsuser@master:~$ 
```

3.2 Verify the deployment rollout history using the following command:
kubectl rollout history deployment/mydep



```
labsuser@master:~$ kubectl rollout history deployment/mydep
deployment.apps/mydep
REVISION  CHANGE-CAUSE
1         kubectl run mydep --image=ghost:0.9 --record=true --dry-run=true --output=yaml
2         kubectl set image deployment/mydep mydep=ghost:0.10 --record=true

labsuser@master:~$ kubectl rollout undo deployment/mydep --to-revision=1
deployment.apps/mydep rolled back
labsuser@master:~$ kubectl rollout history deployment/mydep
deployment.apps/mydep
REVISION  CHANGE-CAUSE
2         kubectl set image deployment/mydep mydep=ghost:0.10 --record=true
3         kubectl run mydep --image=ghost:0.9 --record=true --dry-run=true --output=yaml

labsuser@master:~$ 
```

The deployment image version is returned to its original state.

By following these steps, you have successfully created a Kubernetes deployment, upgraded the image version, and reverted to the previous version, demonstrating effective version control within a Kubernetes cluster. This ensures seamless updates and rollbacks in Kubernetes, helping maintain stable deployments while minimizing downtime and reducing the impact of issues with new versions.