Kubernetes Deployment and Rollout Practice

Part 1: Deployment Creation and Update

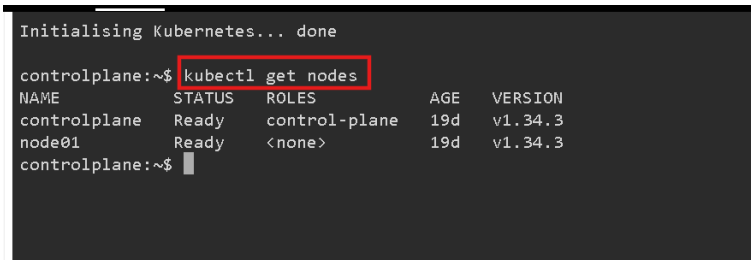
Step 1: Check Kubernetes Cluster

Command:

kubectl get nodes

Explanation:
We are checking whether the Kubernetes cluster is running properly.
All nodes should show **Ready** status before we continue.

Image:



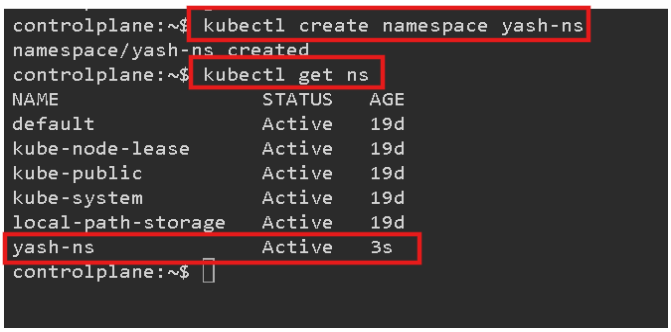
Step 2: Create Namespace

Command:

kubectl create namespace yash-ns
kubectl get ns

Explanation:
We are creating a namespace called **yash-ns** to keep our deployment resources separate and organized.
After creating it, we verify that the namespace exists.

Image:



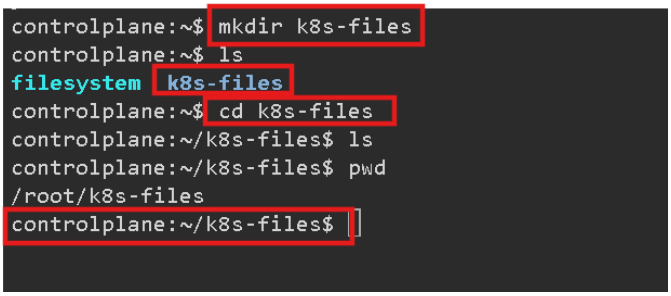
Step 3: Create Working Directory

Command:

mkdir k8s-files
cd k8s-files

Explanation:
We are creating a folder to store our Kubernetes YAML files.
This helps keep the project clean and structured.

Image:



Step 4: Create Deployment YAML File

Command:

vi deployment.yaml

YAML Content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: t20-deployment
  namespace: yash-ns
spec:
  replicas: 7
  selector:
    matchLabels:
      app: t20-wc
  template:
    metadata:
      labels:
        app: t20-wc
    spec:
```

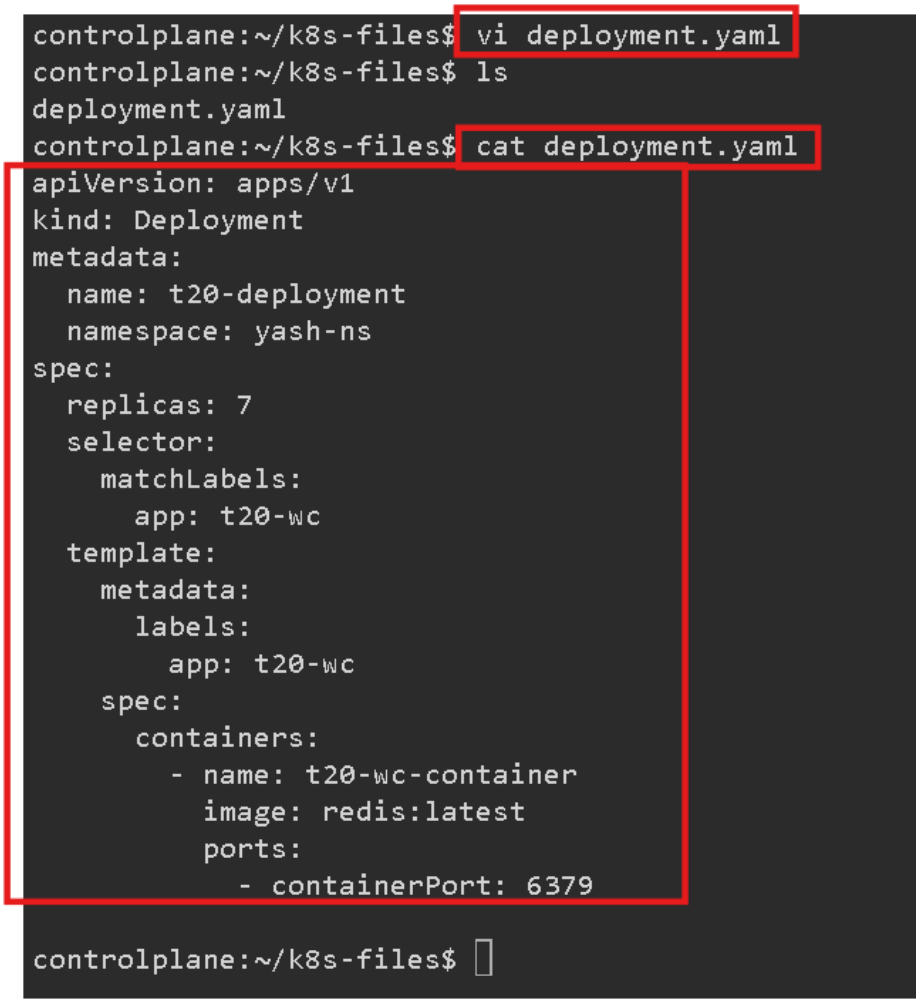
```
    containers:
      - name: t20-wc-container
        image: redis:latest
        ports:
          - containerPort: 6379
```

After saving the file:

cat deployment.yaml

Explanation:
We are creating a deployment configuration file.
This file tells Kubernetes to create 7 pods using the redis image inside the yash-ns namespace.
After saving, we check the file to make sure everything is correct.

Image:



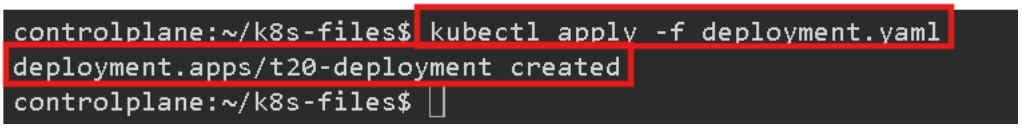
Step 5: Apply Deployment

Command:

kubectl apply -f deployment.yaml

Explanation:
We are applying the YAML file so Kubernetes creates the deployment and starts the pods.

Image:



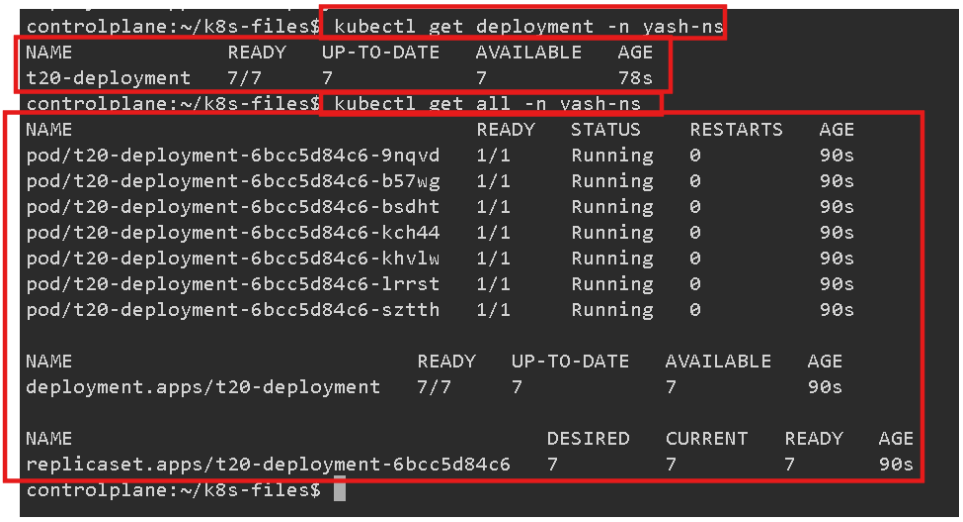
Step 6: Verify Deployment and Resources

Command:

kubectl get deployment -n yash-ns
kubectl get all -n yash-ns

Explanation:
We are checking whether the deployment was created successfully.
This also shows the ReplicaSet and pods that Kubernetes created automatically.

Image:



Step 7: Describe Deployment

Command:

kubectl describe deployment t20-deployment -n yash-ns

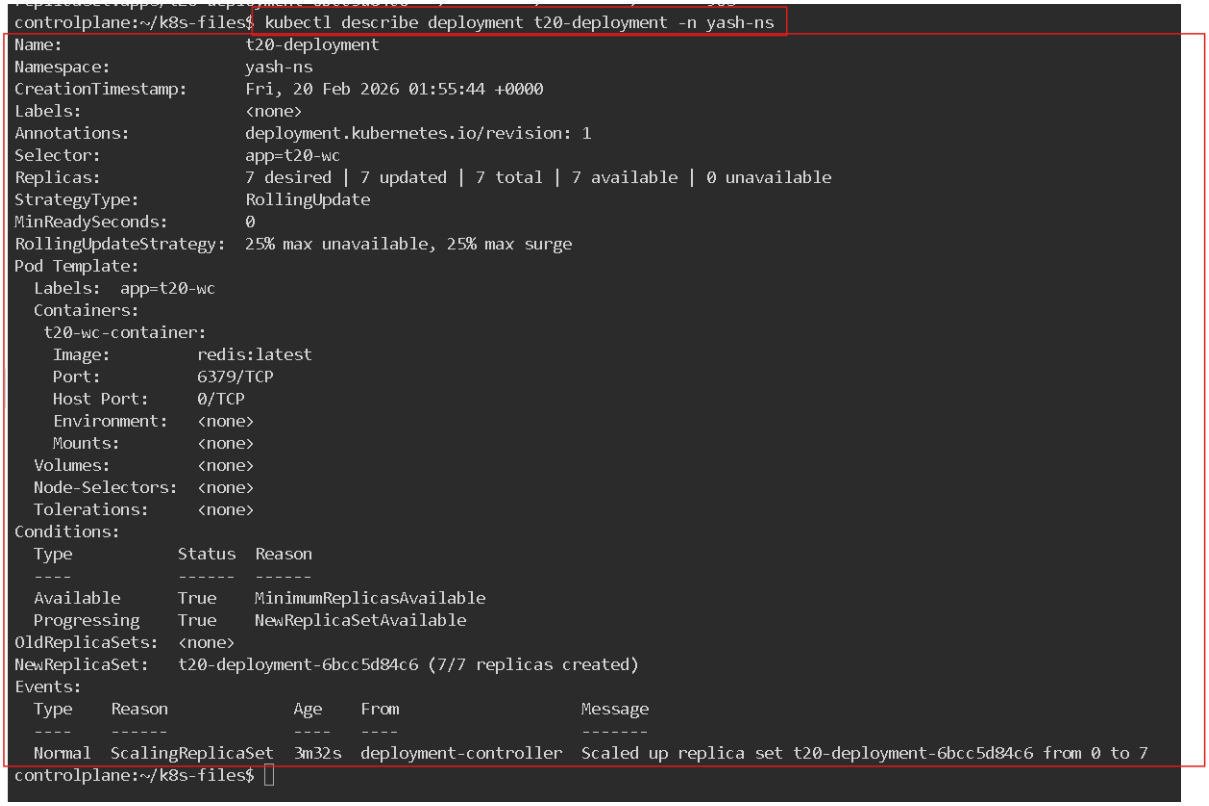
Explanation:

Here we are viewing detailed information about the deployment.
We can see:

- Number of replicas
- Container image used
- Events
- Scaling activity

This helps us understand how Kubernetes is managing the deployment.

Image:



Part 2: Deployment Update and Rollout Management

Step 8: Update Deployment (Change Image and Port)

Command:

vi deployment.yaml

Change image from:

redis:latest

To:

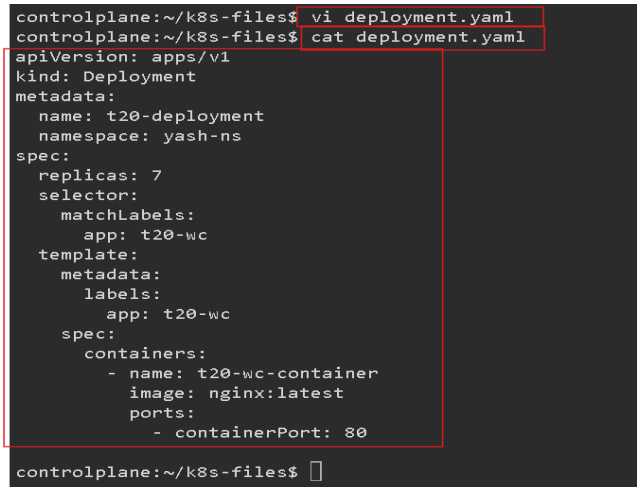
nginx:latest

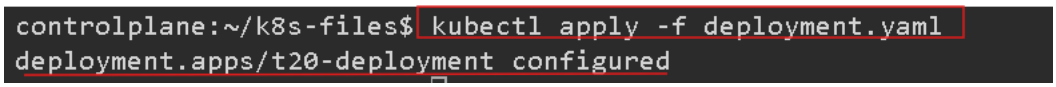
Then run:

cat deployment.yaml
kubectl apply -f deployment.yaml

Explanation:
We are changing the container image from Redis to Nginx and updating the port accordingly.
After saving the changes, we apply the file again so Kubernetes updates the running pods.

Image:





Step 9: Verify Rollout Status and History

Command:

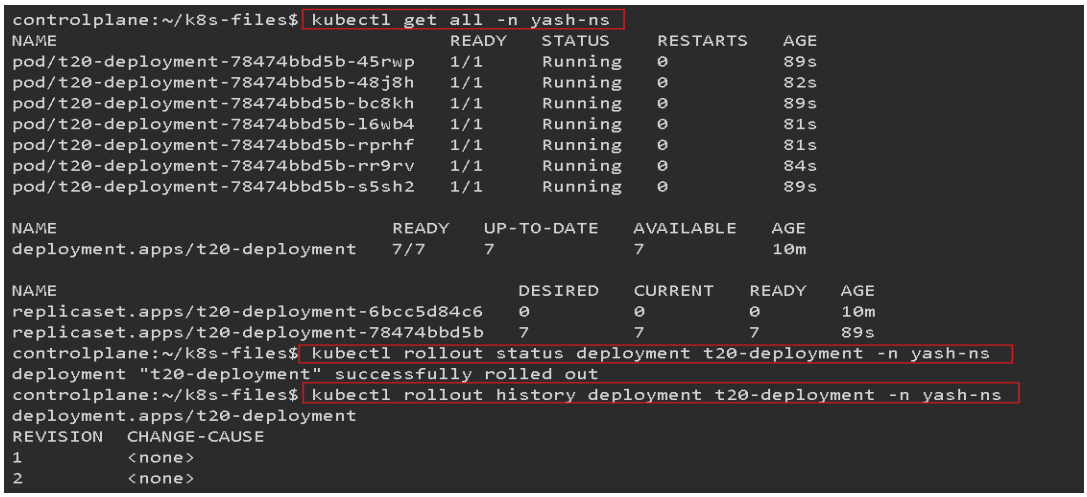
kubectl get all -n yash-ns
kubectl rollout status deployment t20-deployment -n yash-ns
kubectl rollout history deployment t20-deployment -n yash-ns

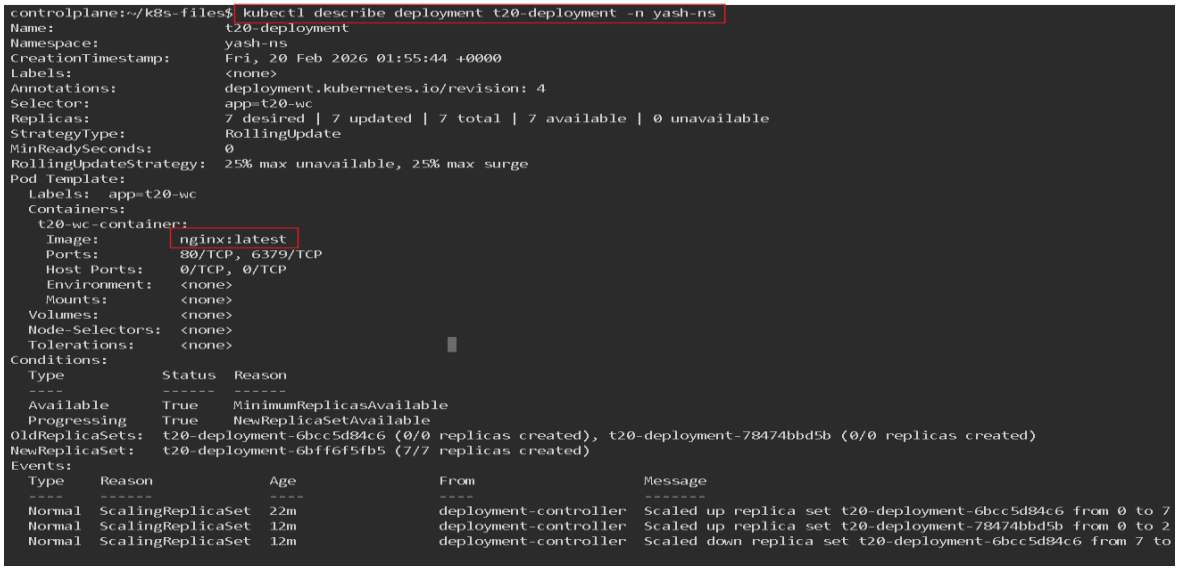
kubectl describe deployment t20-deployment -n yash-ns

Explanation:

First, we check if new pods are created and running.
Then, we verify that the rollout is completed successfully.
The rollout history shows different versions of the deployment.
Finally, we describe the deployment to confirm that the image has changed to **nginx:latest** and the update was applied correctly.

Image:





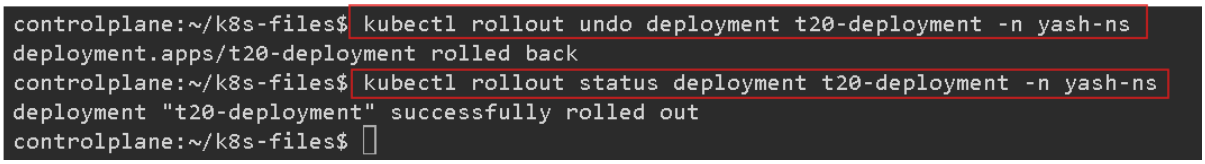
Step 10: Rollback Deployment

Command:

kubectl rollout undo deployment t20-deployment -n yash-ns
kubectl rollout status deployment t20-deployment -n yash-ns

Explanation:
We are rolling back the deployment to the previous version.
This is useful if the new version has any issues.
After rollback, we check the status to ensure it is successful.

Image:



Step 11: Final Verification

Command:

kubectl get all -n yash-ns
kubectl describe deployment t20-deployment -n yash-ns

Explanation:
We are confirming that everything is running correctly after the rollback.
We verify the pods, replicas, and container image to make sure the deployment is stable.

Image:

```
controlplane:~/k8s-files$ kubectl get all -n yash-ns
NAME                                    READY   STATUS    RESTARTS   AGE
pod/t20-deployment-6bcc5d84c6-42hz2     1/1     Running   0          26s
pod/t20-deployment-6bcc5d84c6-75522     1/1     Running   0          30s
pod/t20-deployment-6bcc5d84c6-bf5tv     1/1     Running   0          28s
pod/t20-deployment-6bcc5d84c6-d87kk     1/1     Running   0          30s
pod/t20-deployment-6bcc5d84c6-hqmtv     1/1     Running   0          30s
pod/t20-deployment-6bcc5d84c6-tg4lc     1/1     Running   0          28s
pod/t20-deployment-6bcc5d84c6-z7nq9     1/1     Running   0          29s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/t20-deployment      7/7     7            7           26m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/t20-deployment-6bcc5d84c6      7         7         7       26m
replicaset.apps/t20-deployment-6bff6f5fb5      0         0         0       5m
replicaset.apps/t20-deployment-78474bbd5b      0         0         0       17m
controlplane:~/k8s-files$ kubectl describe deployment t20-deployment -n yash-ns
Name:                   t20-deployment
Namespace:              yash-ns
CreationTimestamp:      Fri, 20 Feb 2026 01:55:44 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 5
Selector:               app=t20-wc
Replicas:               7 desired | 7 updated | 7 total | 7 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=t20-wc
  Containers:
   t20-wc-container:
    Image:          redis:latest
    Port:           6379/TCP
    Host Port:      0/TCP
    Environment:    <none>
    Mounts:         <none>
```

```
    Host Port:      0/TCP
    Environment:    <none>
    Mounts:         <none>
  Volumes:          <none>
  Node-Selectors:   <none>
  Tolerations:      <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  t20-deployment-78474bbd5b (0/0 replicas created), t20-deployment-6bff6f5fb5 (0/0 replicas created)
NewReplicaSet:   t20-deployment-6bcc5d84c6 (7/7 replicas created)
Events:
  Type    Reason             Age                From                   Message
  ----    ------             ----               ----                   -------
  Normal  ScalingReplicaSet  27m                deployment-controller  Scaled up replica set t20-deployment-6bcc5d84c6 from 0 to
7
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled up replica set t20-deployment-78474bbd5b from 0 to
2
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled down replica set t20-deployment-6bcc5d84c6 from 7 t
o 6
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled up replica set t20-deployment-78474bbd5b from 2 to
3
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled down replica set t20-deployment-6bcc5d84c6 from 6 t
o 5
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled up replica set t20-deployment-78474bbd5b from 3 to
4
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled down replica set t20-deployment-6bcc5d84c6 from 5 t
o 4
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled up replica set t20-deployment-78474bbd5b from 4 to
5
  Normal  ScalingReplicaSet  17m                deployment-controller  Scaled down replica set t20-deployment-6bcc5d84c6 from 4 t
o 3
  Normal  ScalingReplicaSet  35s (x32 over 17m)  deployment-controller  (combined from similar events): Scaled up replica set t20-
deployment-6bcc5d84c6 from 0 to 2
```