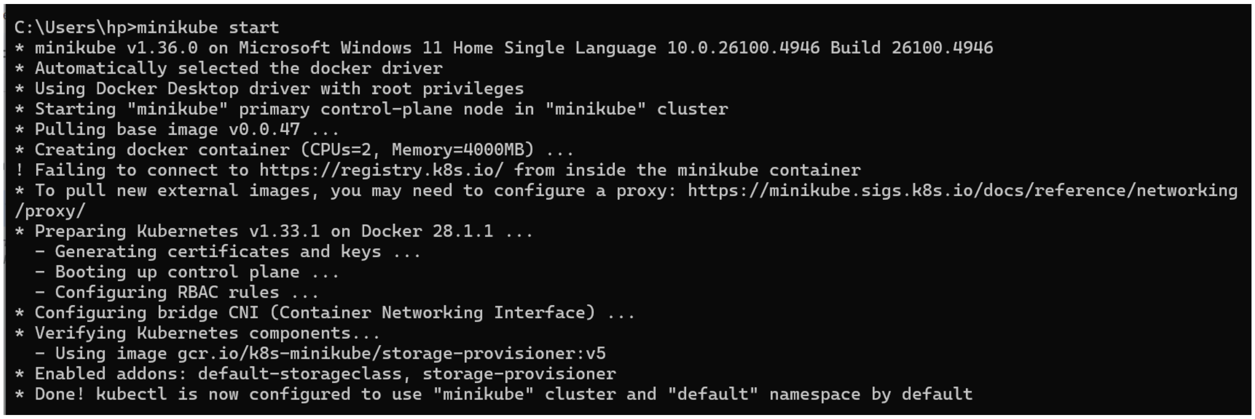
🔹 Exercise 1: Setting Up Your Kubernetes Cluster

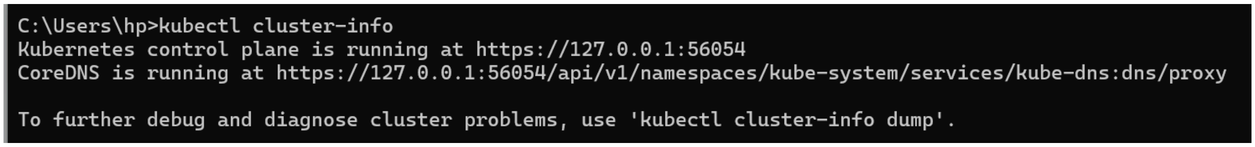
Objective: Set up a local Kubernetes environment using Minikube and kubectl.

1. Install Minikube and kubectl.
2. Start a Minikube cluster with minikube start.
3. Use kubectl cluster-info to verify your cluster is running.
4. List all nodes using kubectl get nodes.

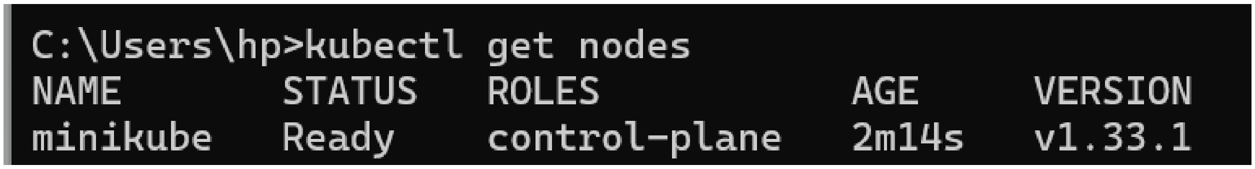
- minikube start
- kubectl cluster-info
- kubectl get nodes



```
C:\Users\hp>minikube start
* minikube v1.36.0 on Microsoft Windows 11 Home Single Language 10.0.26100.4946 Build 26100.4946
* Automatically selected the docker driver
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.47 ...
* Creating docker container (CPUs=2, Memory=4000MB) ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking
/proxy/
* Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```



```
C:\Users\hp>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:56054
CoreDNS is running at https://127.0.0.1:56054/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```



```
C:\Users\hp>kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   2m14s   v1.33.1
```

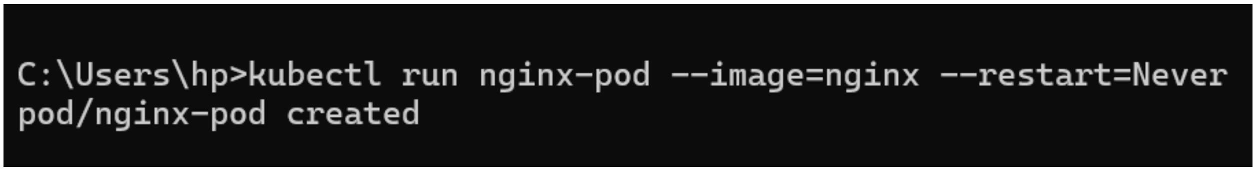
🔹 Exercise 2: Creating and Managing Pods

Objective: Learn how to create and manage Pods.

1. Create a simple pod using a predefined image like nginx
2. Check the pod status.
3. View pod logs
4. Expose the pod via a service

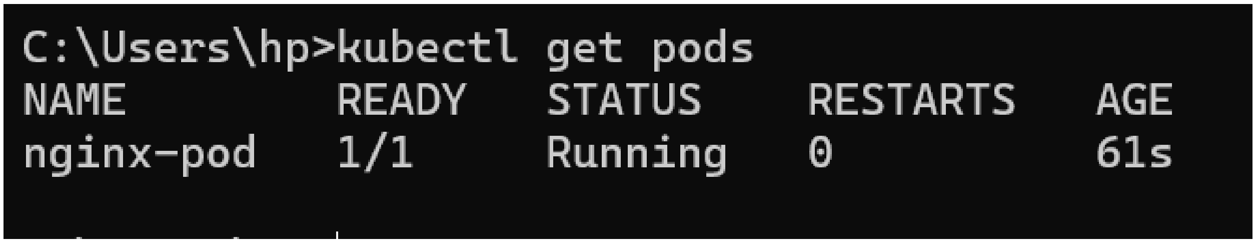
Checkpoint : What happens when you delete a pod? Test it by deleting the nginx pod and observe the behavior of the cluster.

- kubectl run nginx-pod --image=nginx --restart=Never



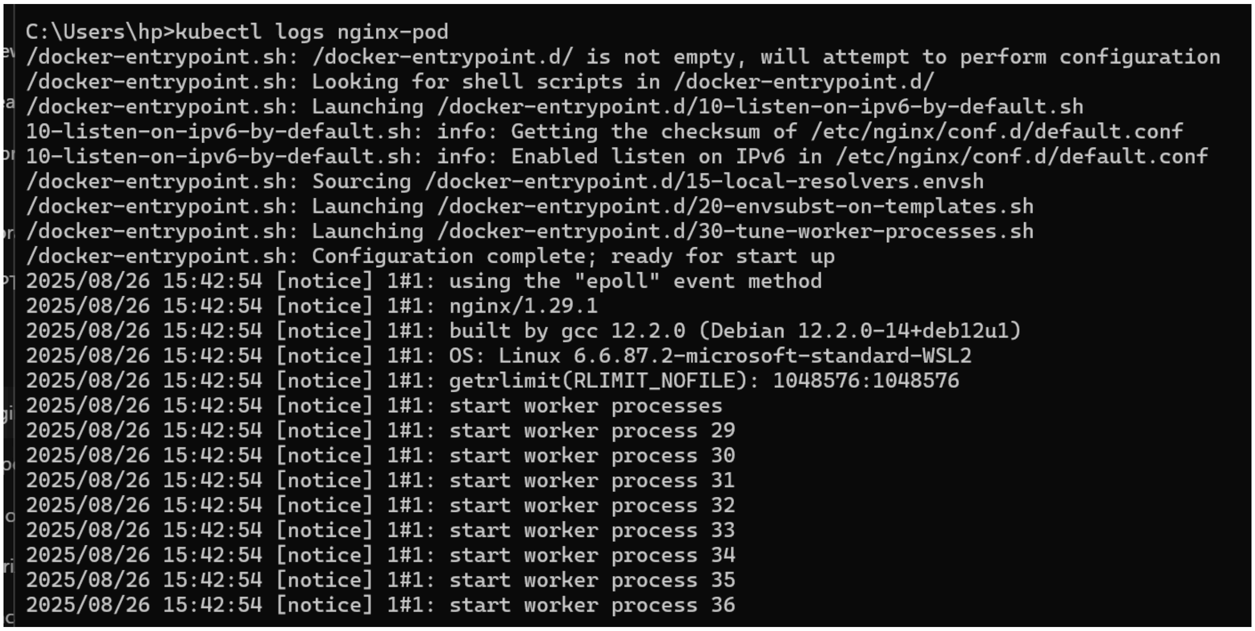
```
C:\Users\hp>kubectl run nginx-pod --image=nginx --restart=Never
pod/nginx-pod created
```

- kubectl get pods



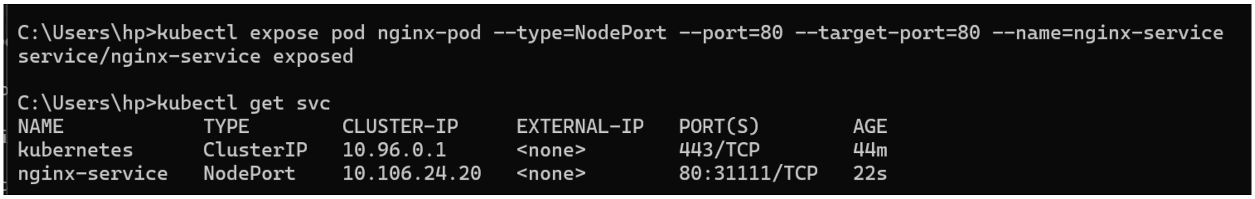
```
C:\Users\hp>kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          61s
```

- kubectl logs nginx-pod



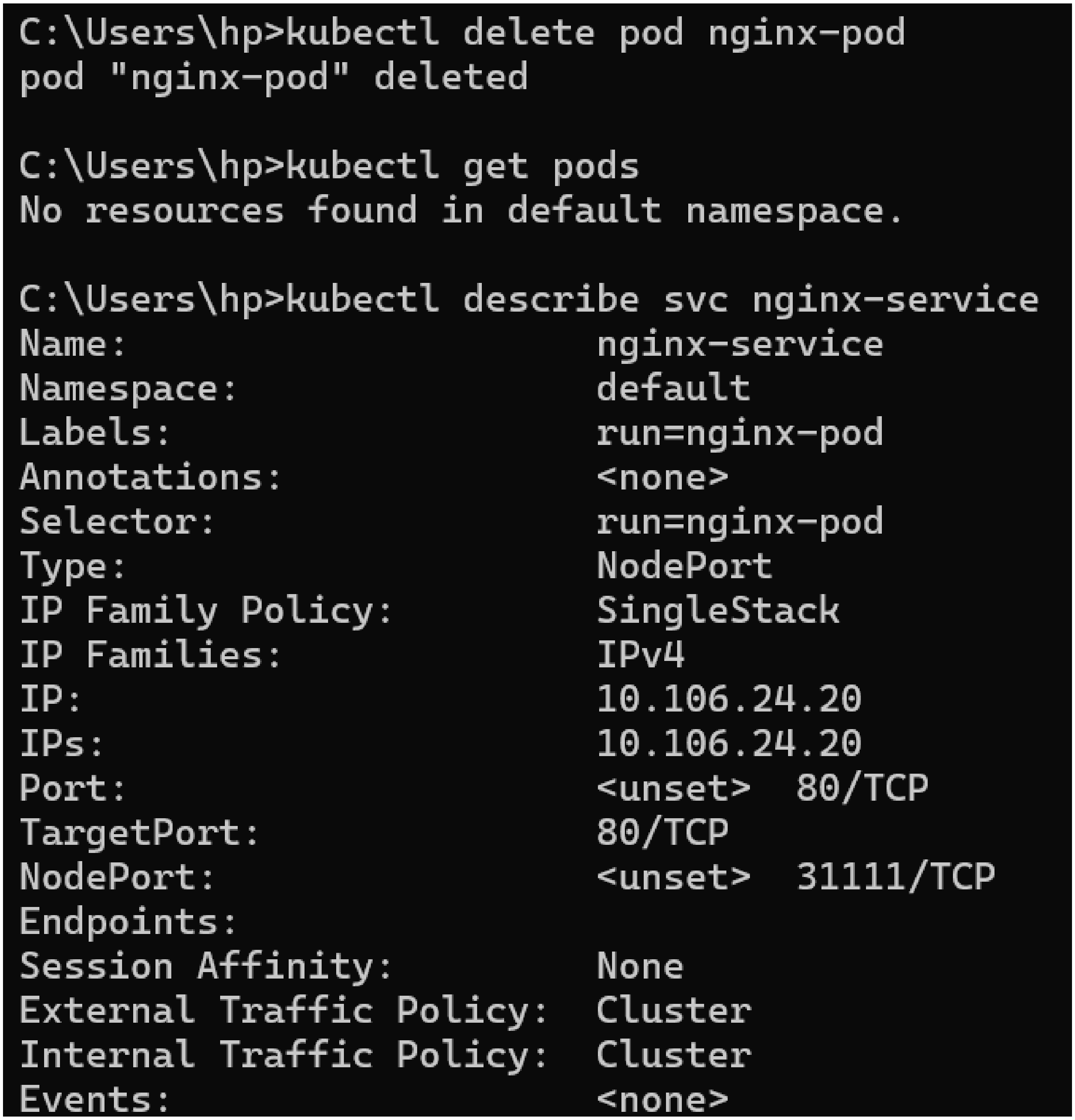
```
C:\Users\hp>kubectl logs nginx-pod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/08/26 15:42:54 [notice] 1#1: using the "epoll" event method
2025/08/26 15:42:54 [notice] 1#1: nginx/1.29.1
2025/08/26 15:42:54 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14+deb12u1)
2025/08/26 15:42:54 [notice] 1#1: OS: Linux 6.6.87.2-microsoft-standard-WSL2
2025/08/26 15:42:54 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/08/26 15:42:54 [notice] 1#1: start worker processes
2025/08/26 15:42:54 [notice] 1#1: start worker process 29
2025/08/26 15:42:54 [notice] 1#1: start worker process 30
2025/08/26 15:42:54 [notice] 1#1: start worker process 31
2025/08/26 15:42:54 [notice] 1#1: start worker process 32
2025/08/26 15:42:54 [notice] 1#1: start worker process 33
2025/08/26 15:42:54 [notice] 1#1: start worker process 34
2025/08/26 15:42:54 [notice] 1#1: start worker process 35
2025/08/26 15:42:54 [notice] 1#1: start worker process 36
```

- kubectl expose pod nginx-pod --type=NodePort --port=80 --target-port=80 --name=nginx-service
- kubectl get svc



```
C:\Users\hp>kubectl expose pod nginx-pod --type=NodePort --port=80 --target-port=80 --name=nginx-service
service/nginx-service exposed

C:\Users\hp>kubectl get svc
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes       ClusterIP   10.96.0.1      <none>        443/TCP        44m
nginx-service    NodePort    10.106.24.20   <none>        80:31111/TCP   22s
```

Since we created a Pod directly, it is not managed by a controller (like Deployment, ReplicaSet).
So it will be gone permanently. The service will still exist, but with no endpoints.



```
C:\Users\hp>kubectl delete pod nginx-pod
pod "nginx-pod" deleted

C:\Users\hp>kubectl get pods
No resources found in default namespace.

C:\Users\hp>kubectl describe svc nginx-service
Name:                     nginx-service
Namespace:                default
Labels:                   run=nginx-pod
Annotations:              <none>
Selector:                 run=nginx-pod
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.106.24.20
IPs:                      10.106.24.20
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31111/TCP
Endpoints:
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
```

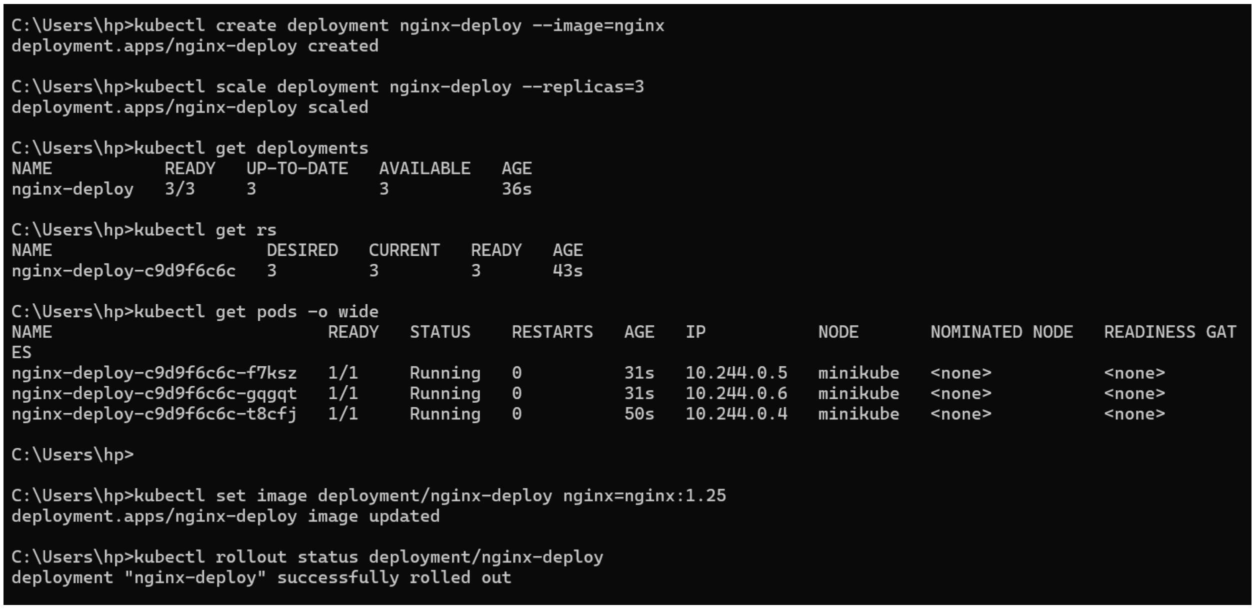
🔹 Exercise 3: Working with Deployments

Objective: Use Deployments for managing replicated Pods.
1. Create a deployment with using nginx image
2. Scale the deployment to 3 replicas
3. Verify the deployment
4. Update the deployment by changing the image(imperative way)

Checkpoint:

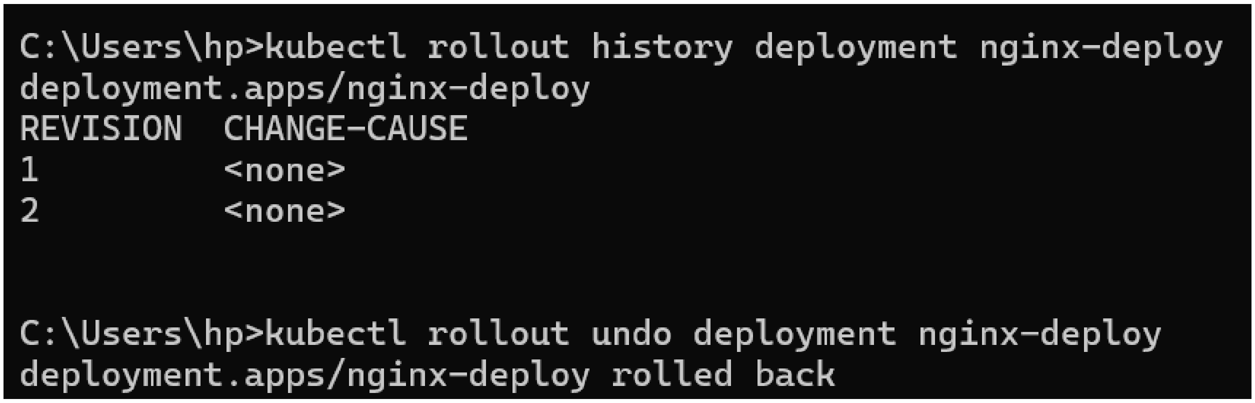
● What does deployment rollout history show? How would you roll back a deployment?

- kubectl create deployment nginx-deploy --image=nginx
- kubectl scale deployment nginx-deploy --replicas=3
- kubectl get deployments
- kubectl get rs
- kubectl get pods -o wide
- kubectl set image deployment/nginx-deploy nginx=nginx:1.25
- kubectl rollout status deployment/nginx-deploy



```
C:\Users\hp>kubectl create deployment nginx-deploy --image=nginx
deployment.apps/nginx-deploy created

C:\Users\hp>kubectl scale deployment nginx-deploy --replicas=3
deployment.apps/nginx-deploy scaled

C:\Users\hp>kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deploy   3/3     3            3           36s

C:\Users\hp>kubectl get rs
NAME                     DESIRED   CURRENT   READY   AGE
nginx-deploy-c9d9f6c6c   3         3         3       43s

C:\Users\hp>kubectl get pods -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GAT
ES
nginx-deploy-c9d9f6c6c-f7ksz   1/1     Running   0          31s   10.244.0.5   minikube   <none>           <none>
nginx-deploy-c9d9f6c6c-gqgqt   1/1     Running   0          31s   10.244.0.6   minikube   <none>           <none>
nginx-deploy-c9d9f6c6c-t8cfj   1/1     Running   0          50s   10.244.0.4   minikube   <none>           <none>

C:\Users\hp>

C:\Users\hp>kubectl set image deployment/nginx-deploy nginx=nginx:1.25
deployment.apps/nginx-deploy image updated

C:\Users\hp>kubectl rollout status deployment/nginx-deploy
deployment "nginx-deploy" successfully rolled out
```

- kubectl rollout history deployment nginx-deploy
- kubectl rollout undo deployment nginx-deploy



```
C:\Users\hp>kubectl rollout history deployment nginx-deploy
deployment.apps/nginx-deploy
REVISION  CHANGE-CAUSE
1         <none>
2         <none>


C:\Users\hp>kubectl rollout undo deployment nginx-deploy
deployment.apps/nginx-deploy rolled back
```

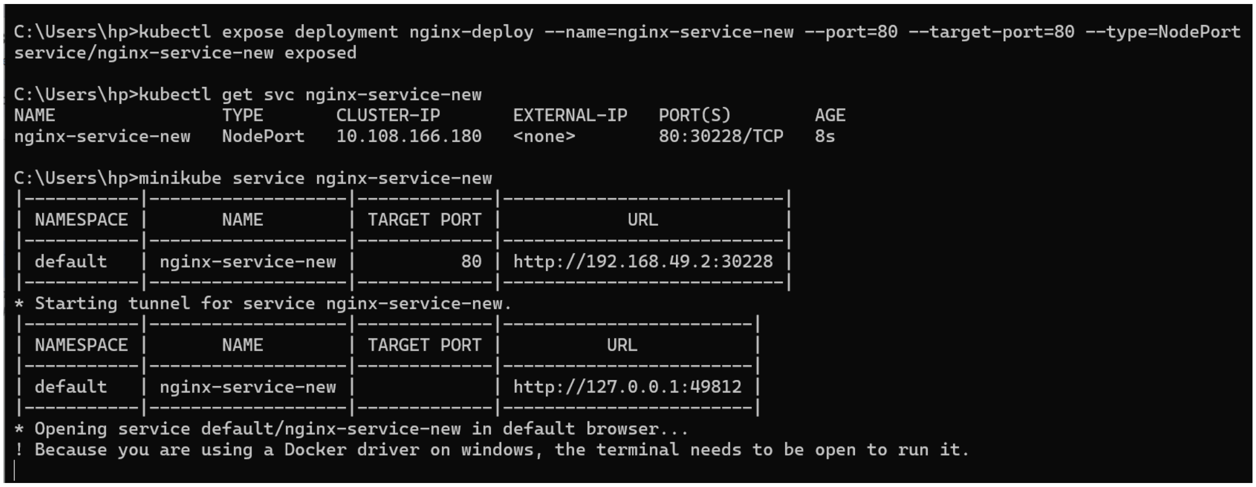
Exercise 4: Services and Networking

Objective: Expose your app using Kubernetes services.

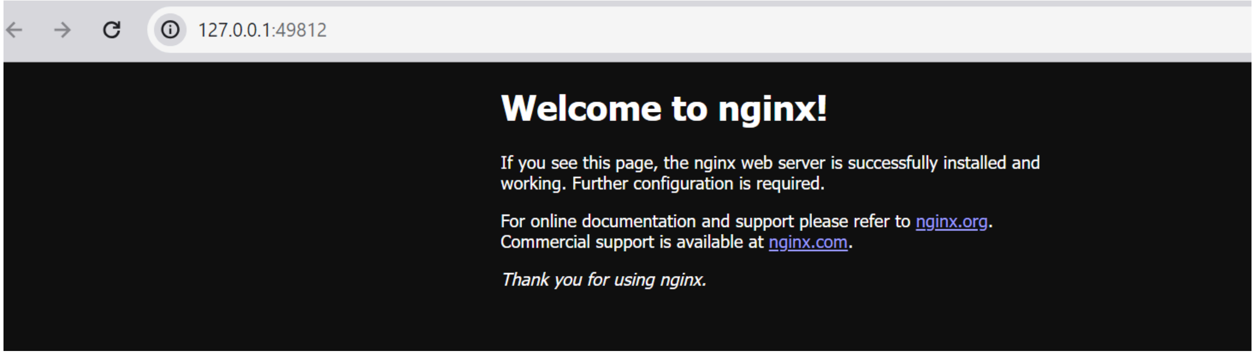
1. Expose your nginx deployment using a Service:
2. Create a service of type NodePort to make it accessible externally.
3. View the service details.
4. Test access to the nginx service through the browser.

Checkpoint:

● What is the difference between ClusterIP, NodePort, and LoadBalancer services?
When should you use each?



```
C:\Users\hp>kubectl expose deployment nginx-deploy --name=nginx-service-new --port=80 --target-port=80 --type=NodePort
service/nginx-service-new exposed

C:\Users\hp>kubectl get svc nginx-service-new
NAME                TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
nginx-service-new   NodePort   10.108.166.180   <none>        80:30228/TCP   8s

C:\Users\hp>minikube service nginx-service-new
|-----------|-------------------|-------------|---------------------------|
| NAMESPACE |       NAME        | TARGET PORT |            URL            |
|-----------|-------------------|-------------|---------------------------|
| default   | nginx-service-new |          80 | http://192.168.49.2:30228 |
|-----------|-------------------|-------------|---------------------------|
* Starting tunnel for service nginx-service-new.
|-----------|-------------------|-------------|------------------------|
| NAMESPACE |       NAME        | TARGET PORT |          URL           |
|-----------|-------------------|-------------|------------------------|
| default   | nginx-service-new |             | http://127.0.0.1:49812 |
|-----------|-------------------|-------------|------------------------|
* Opening service default/nginx-service-new in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

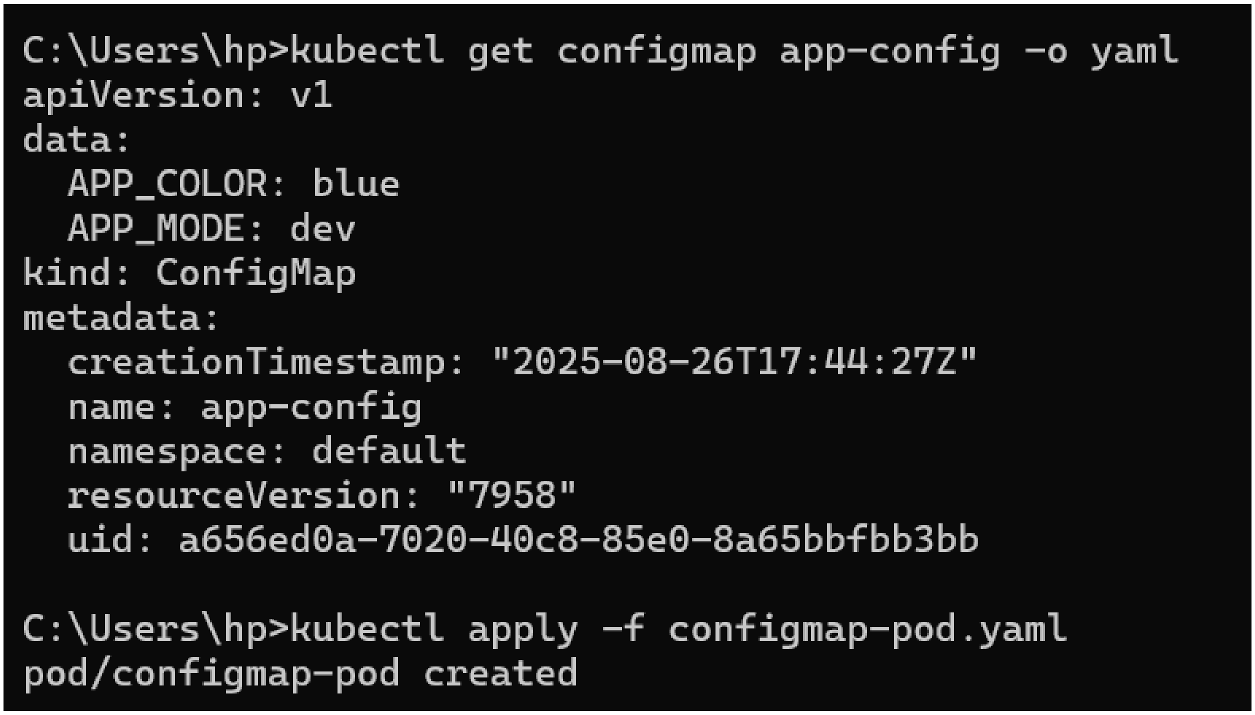
ClusterIP (default): Exposes the service inside the cluster only. Used for internal service-to-service communication (e.g., backend, DB).
NodePort: Exposes the service on each Node’s IP at a static port (30000–32767). Good for dev/testing or bare-metal clusters.
LoadBalancer: Creates an external IP via cloud provider’s load balancer. Best for production apps that need public access.

🔹 Exercise 5: ConfigMaps and Secrets

Objective: Manage configurations using ConfigMaps and Secrets.
1. Create a ConfigMap using a key-value pair:
2. Mount the ConfigMap as environment variables in a pod.
3. Create a Secret:
4. Access the Secret in the pod via environment variables.

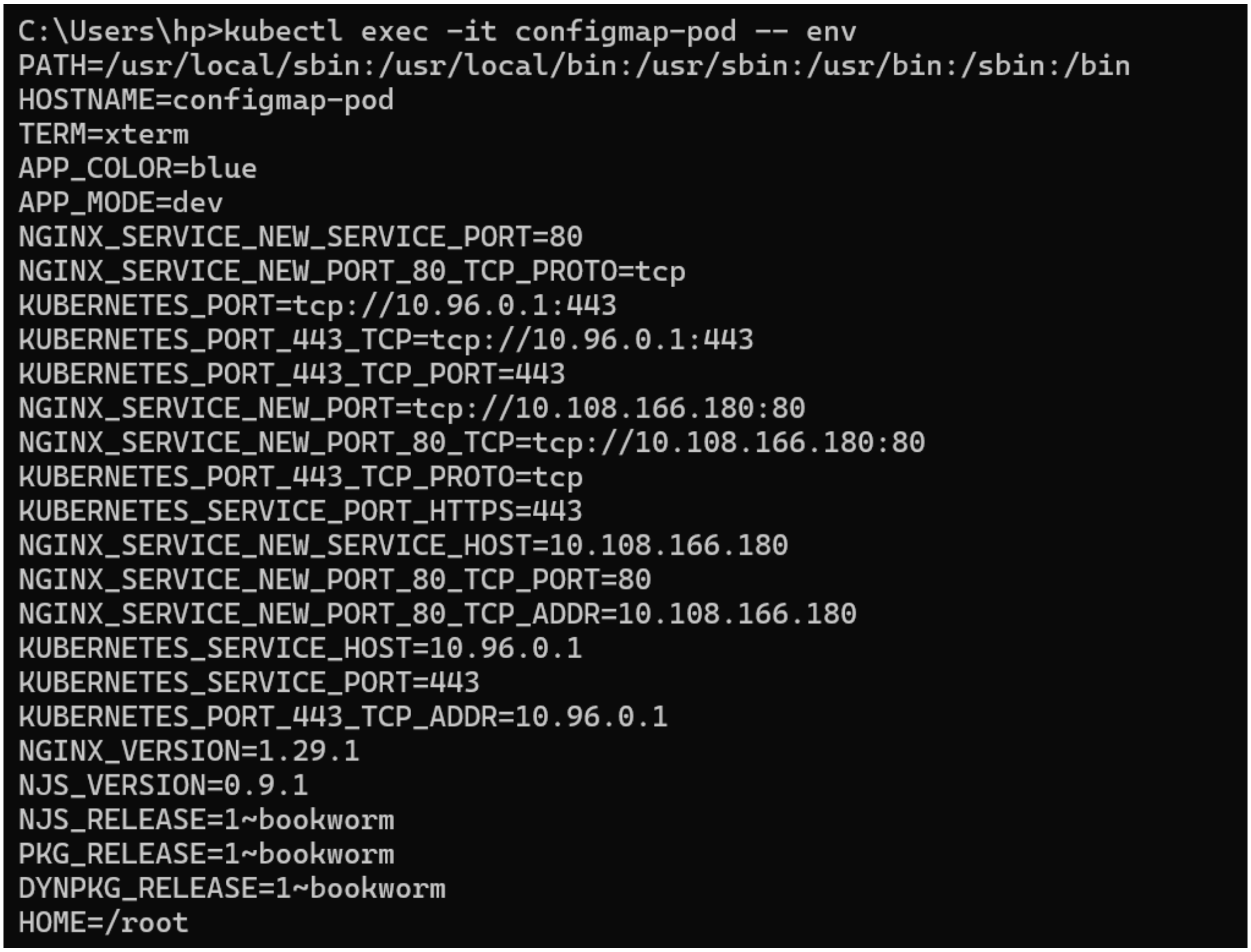
Checkpoint:
● How would you access the value of a ConfigMap or Secret within your application?

- kubectl create configmap app-config --from-literal=APP_COLOR=blue --from-literal=APP_MODE=dev
- kubectl get configmap app-config -o yaml
- kubectl apply -f configmap-pod.yaml

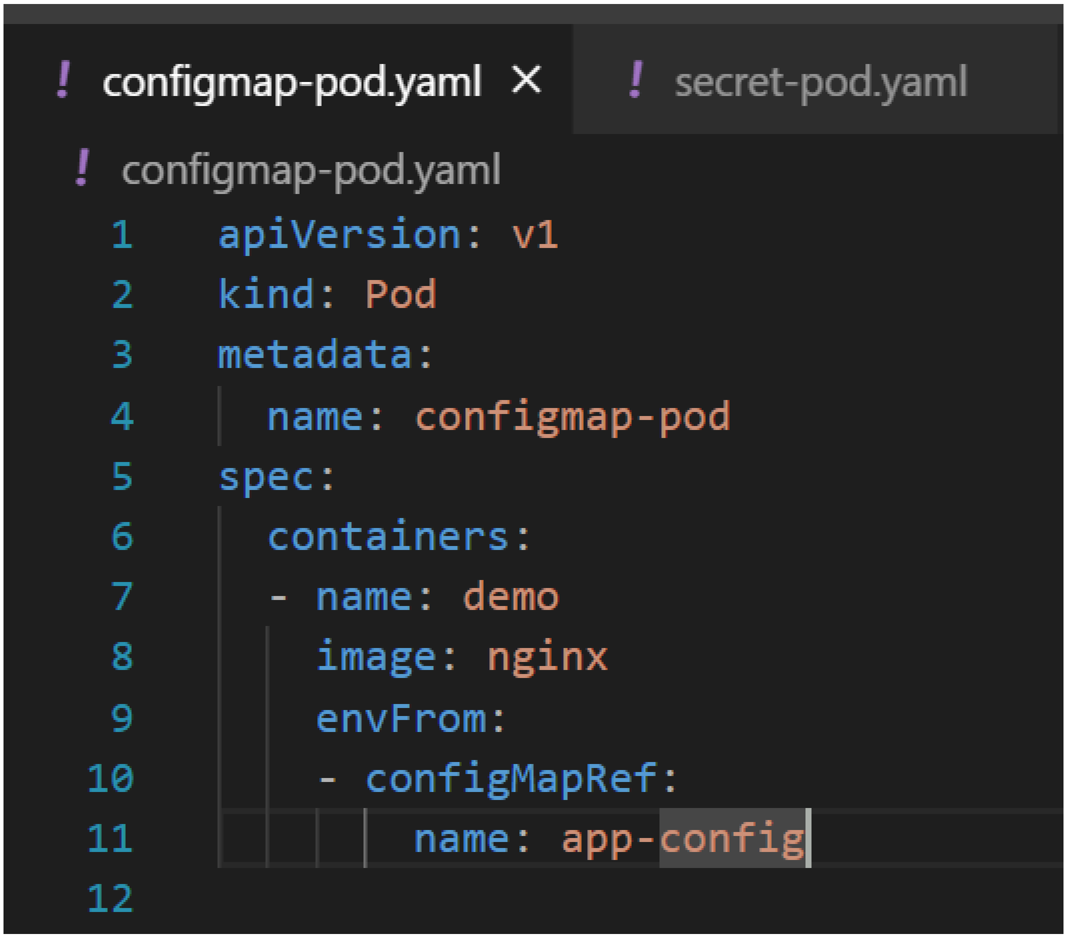


```
C:\Users\hp>kubectl get configmap app-config -o yaml
apiVersion: v1
data:
  APP_COLOR: blue
  APP_MODE: dev
kind: ConfigMap
metadata:
  creationTimestamp: "2025-08-26T17:44:27Z"
  name: app-config
  namespace: default
  resourceVersion: "7958"
  uid: a656ed0a-7020-40c8-85e0-8a65bbfbb3bb

C:\Users\hp>kubectl apply -f configmap-pod.yaml
pod/configmap-pod created
```

- kubectl exec -it configmap-pod -- env



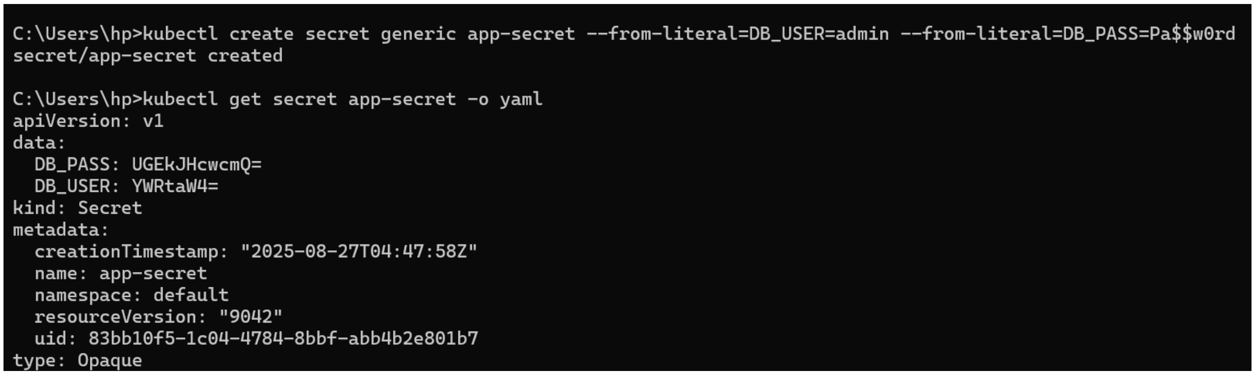
```
C:\Users\hp>kubectl exec -it configmap-pod -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=configmap-pod
TERM=xterm
APP_COLOR=blue
APP_MODE=dev
NGINX_SERVICE_NEW_SERVICE_PORT=80
NGINX_SERVICE_NEW_PORT_80_TCP_PROTO=tcp
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PORT=443
NGINX_SERVICE_NEW_PORT=tcp://10.108.166.180:80
NGINX_SERVICE_NEW_PORT_80_TCP=tcp://10.108.166.180:80
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
NGINX_SERVICE_NEW_SERVICE_HOST=10.108.166.180
NGINX_SERVICE_NEW_PORT_80_TCP_PORT=80
NGINX_SERVICE_NEW_PORT_80_TCP_ADDR=10.108.166.180
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
NGINX_VERSION=1.29.1
NJS_VERSION=0.9.1
NJS_RELEASE=1~bookworm
PKG_RELEASE=1~bookworm
DYNPKG_RELEASE=1~bookworm
HOME=/root
```



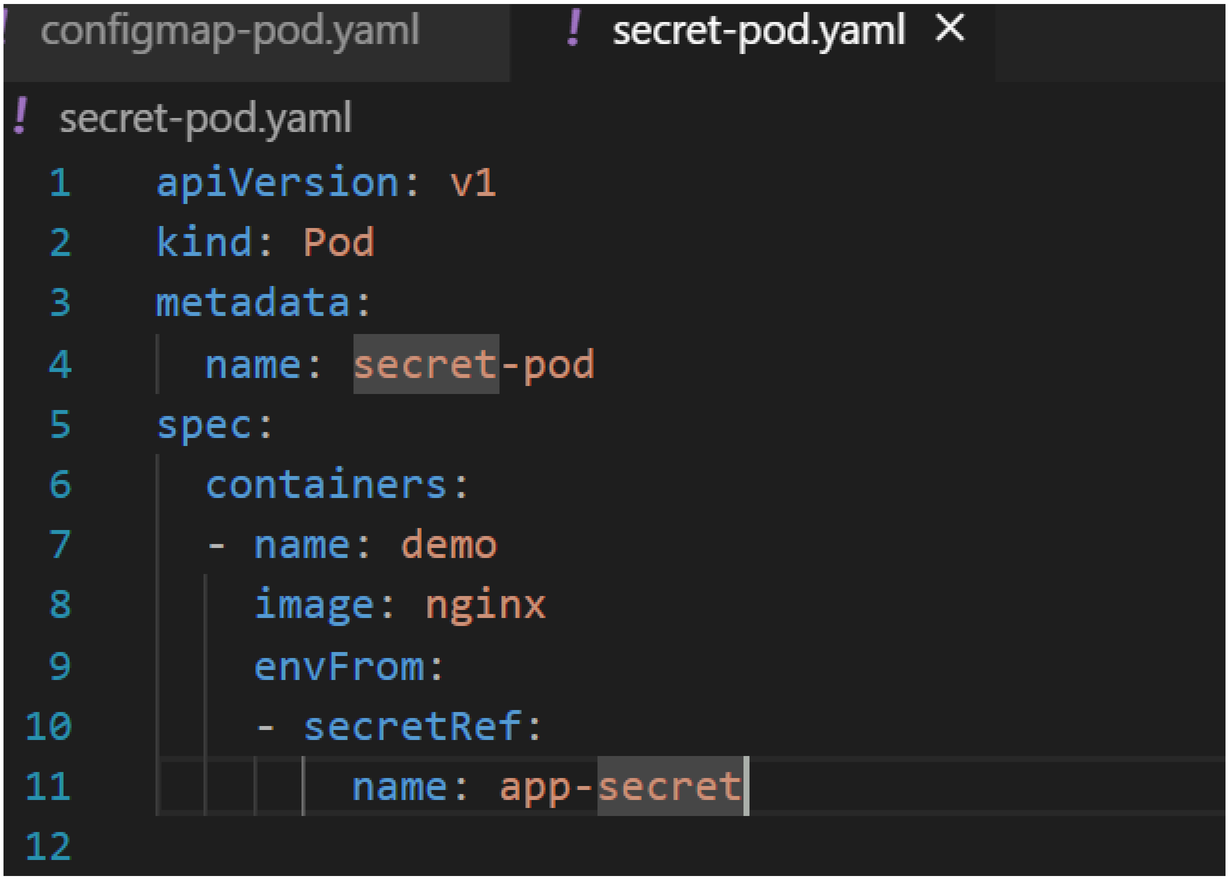
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod
spec:
  containers:
  - name: demo
    image: nginx
    envFrom:
    - configMapRef:
        name: app-config

```

- kubectl create secret generic app-secret --from-literal=DB_USER=admin --from-literal=DB_PASS=Pa$$w0rd

- kubectl get secret app-secret -o yaml

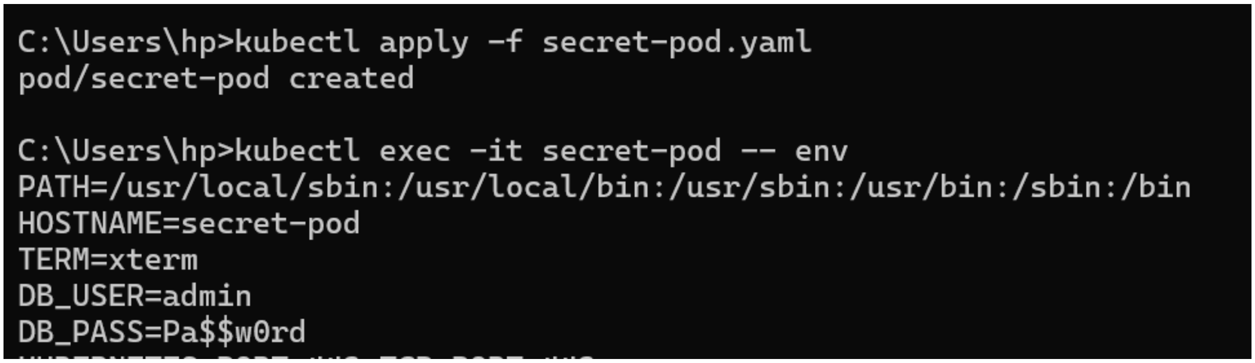


```
C:\Users\hp>kubectl create secret generic app-secret --from-literal=DB_USER=admin --from-literal=DB_PASS=Pa$$w0rd
secret/app-secret created

C:\Users\hp>kubectl get secret app-secret -o yaml
apiVersion: v1
data:
  DB_PASS: UGEkJHcwcmQ=
  DB_USER: YWRtaW4=
kind: Secret
metadata:
  creationTimestamp: "2025-08-27T04:47:58Z"
  name: app-secret
  namespace: default
  resourceVersion: "9042"
  uid: 83bb10f5-1c04-4784-8bbf-abb4b2e801b7
type: Opaque
```



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-pod
spec:
  containers:
  - name: demo
    image: nginx
    envFrom:
    - secretRef:
        name: app-secret

```

- kubectl apply -f secret-pod.yaml
- kubectl exec -it secret-pod -- env



```
C:\Users\hp>kubectl apply -f secret-pod.yaml
pod/secret-pod created

C:\Users\hp>kubectl exec -it secret-pod -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=secret-pod
TERM=xterm
DB_USER=admin
DB_PASS=Pa$$w0rd
```

For accessing the values of configmap or secret, we can access them using normal environment variables.

import os
user = os.getenv("DB_USER")
password = os.getenv("DB_PASS")

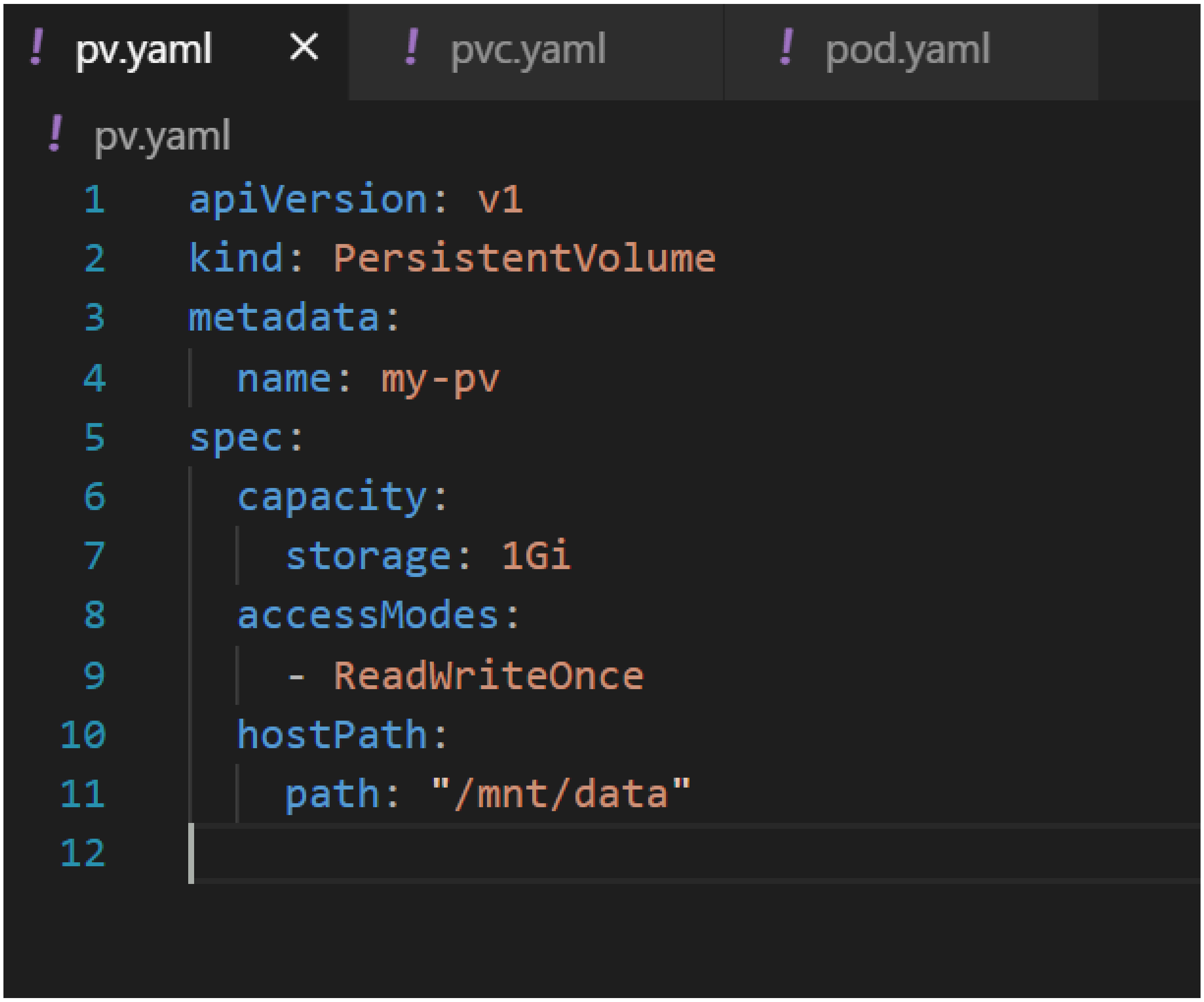
🔹 Exercise 6: Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)

Objective: Use PVs and PVCs for persistent data storage.

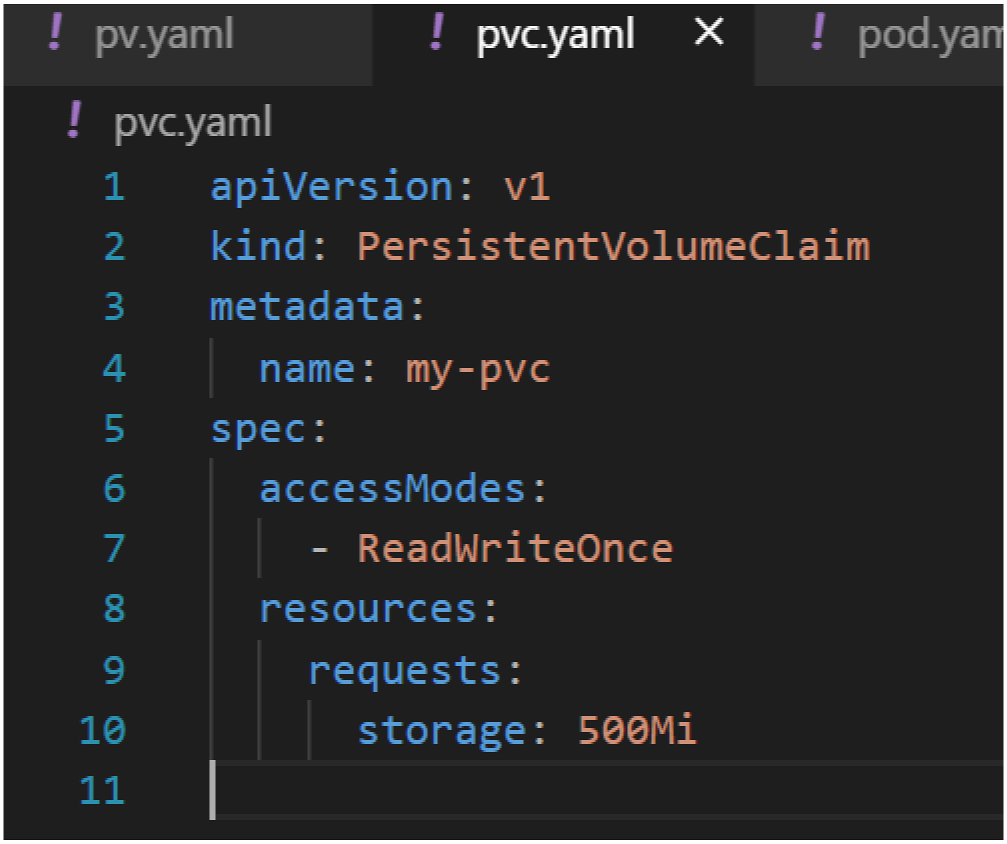
1. Create a Persistent Volume (PV) and a Persistent Volume Claim (PVC) in YAML.
2. Apply the YAML files to create the PV and PVC.
3. Create a pod that uses the PVC to mount the volume.
4. Write data to the volume and verify its persistence by restarting the pod.

Checkpoint:

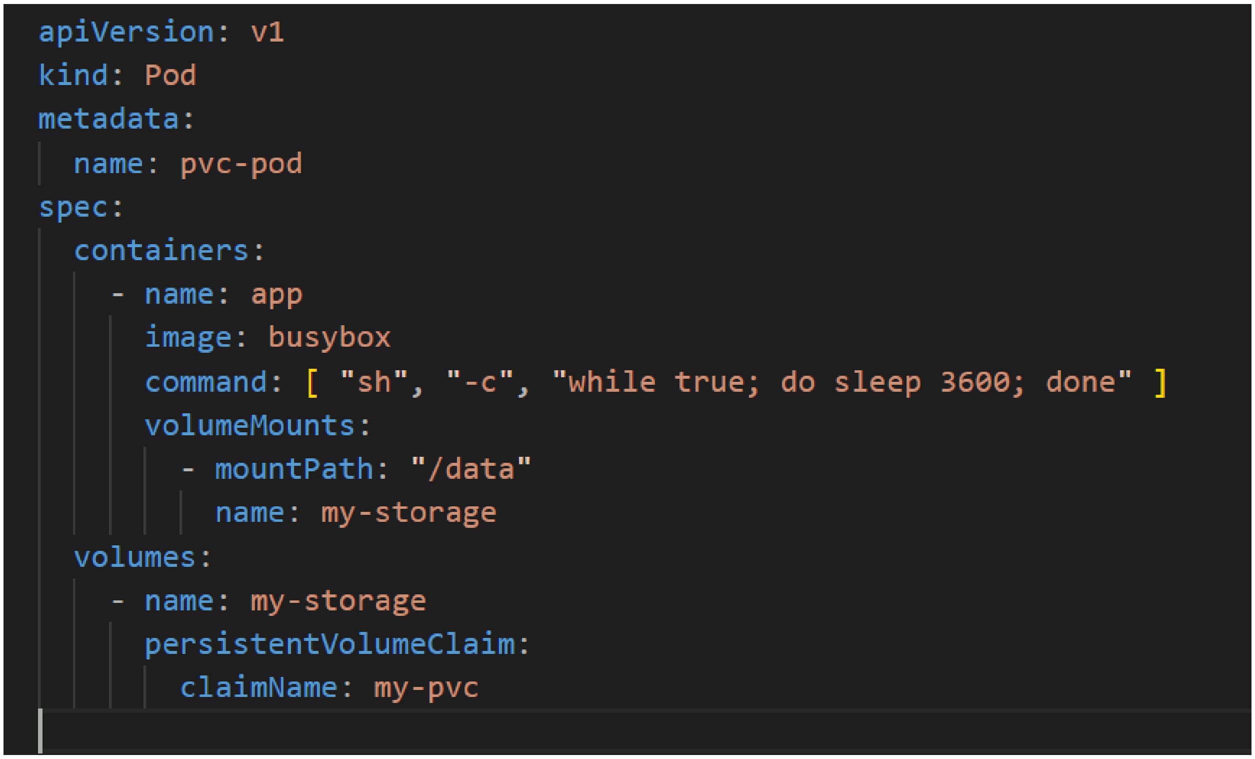
- What happens if the PVC is deleted? Does the underlying Persistent Volume get deleted as well?



```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"

```



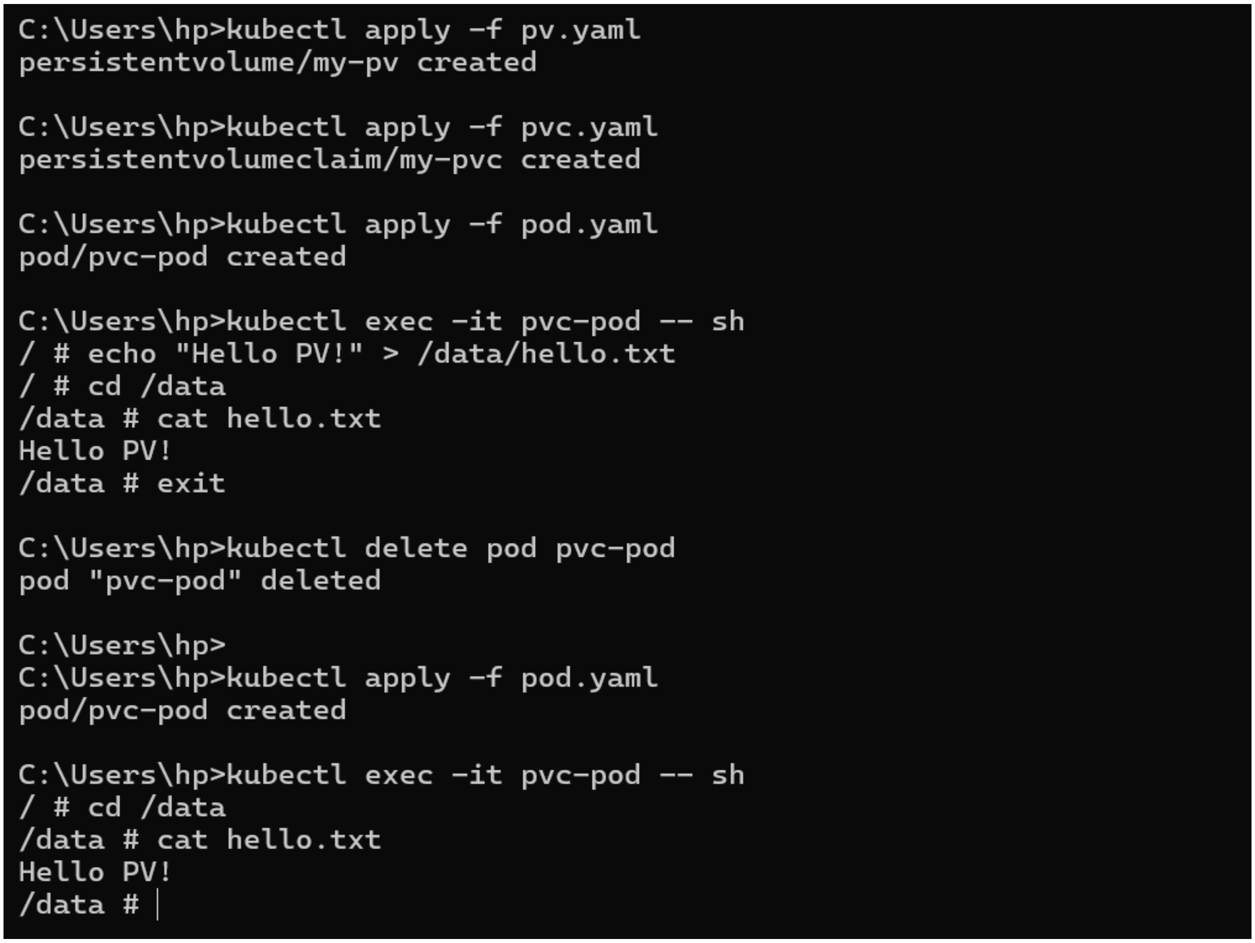
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi

```



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pvc-pod
spec:
  containers:
    - name: app
      image: busybox
      command: [ "sh", "-c", "while true; do sleep 3600; done" ]
      volumeMounts:
        - mountPath: "/data"
          name: my-storage
  volumes:
    - name: my-storage
      persistentVolumeClaim:
        claimName: my-pvc

```

- **kubectl apply -f pv.yaml**
- **kubectl apply -f pvc.yaml**
- **kubectl apply -f pod.yaml**

- kubectl exec -it pvc-pod -- sh
- / # echo "Hello PV!" > /data/hello.txt
- / # cd /data
- /data # cat hello.txt
- Hello PV!
- /data # exit
- kubectl delete pod pvc-pod
- kubectl apply -f pod.yaml
- kubectl exec -it pvc-pod -- sh
- / # cd /data
- /data # cat hello.txt
- Hello PV!



```
C:\Users\hp>kubectl apply -f pv.yaml
persistentvolume/my-pv created

C:\Users\hp>kubectl apply -f pvc.yaml
persistentvolumeclaim/my-pvc created

C:\Users\hp>kubectl apply -f pod.yaml
pod/pvc-pod created

C:\Users\hp>kubectl exec -it pvc-pod -- sh
/ # echo "Hello PV!" > /data/hello.txt
/ # cd /data
/data # cat hello.txt
Hello PV!
/data # exit

C:\Users\hp>kubectl delete pod pvc-pod
pod "pvc-pod" deleted

C:\Users\hp>
C:\Users\hp>kubectl apply -f pod.yaml
pod/pvc-pod created

C:\Users\hp>kubectl exec -it pvc-pod -- sh
/ # cd /data
/data # cat hello.txt
Hello PV!
/data #
```

When a PVC is deleted, the bound PV enters a Released state but is not deleted automatically.
The fate of the underlying storage depends on the PV’s reclaimPolicy, it may be retained, recycled, or deleted.

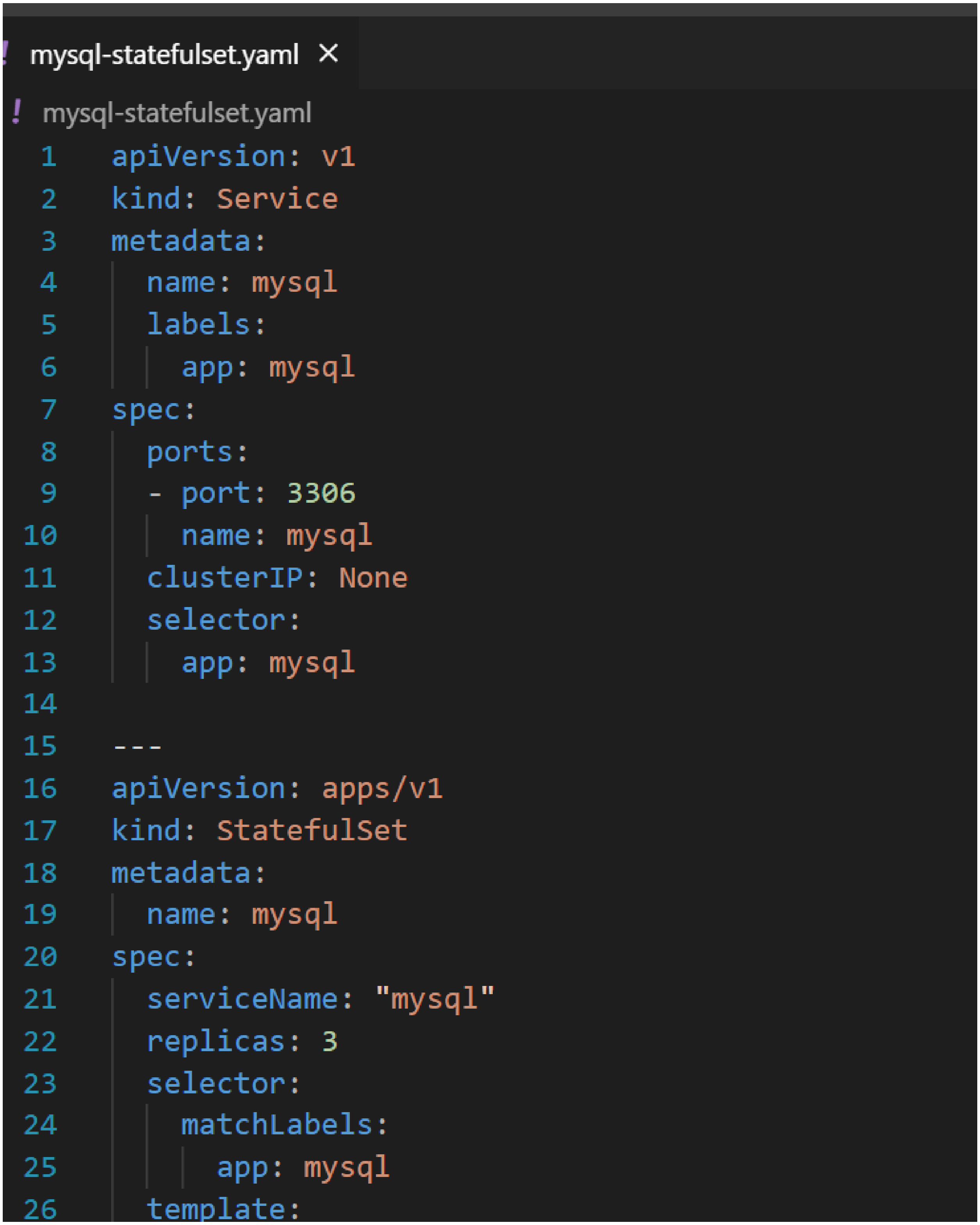
Exercise 7: StatefulSets

Objective: Use StatefulSets for managing stateful applications.

1. Deploy a StatefulSet with (create YAML for an app like MySQL).
2. View the StatefulSet.
3. Create a headless service for the StatefulSet and access the pod by its stable network identity.

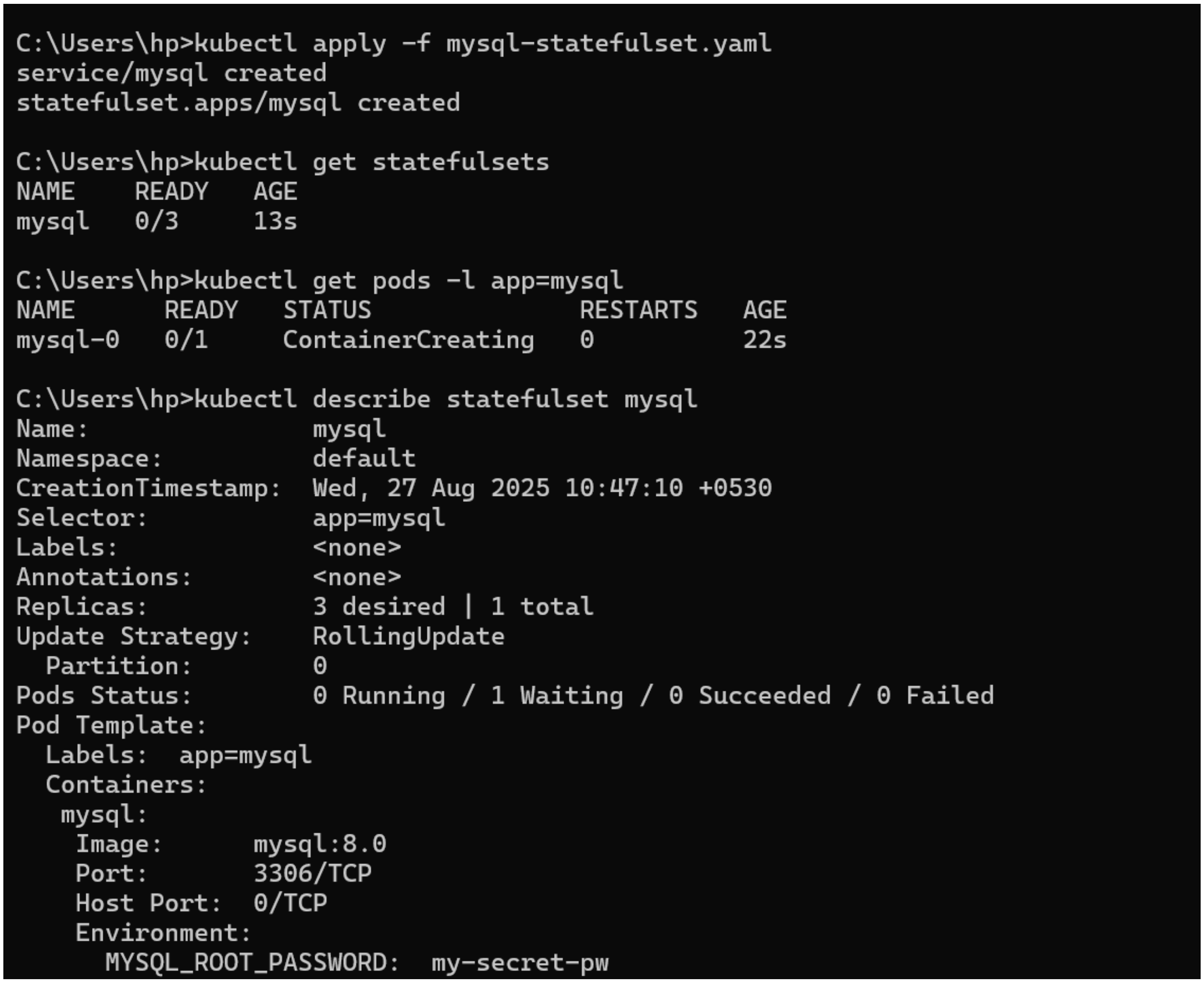
Checkpoint:

- What are the key differences between StatefulSets and Deployments? When would you use a StatefulSet instead of a Deployment?



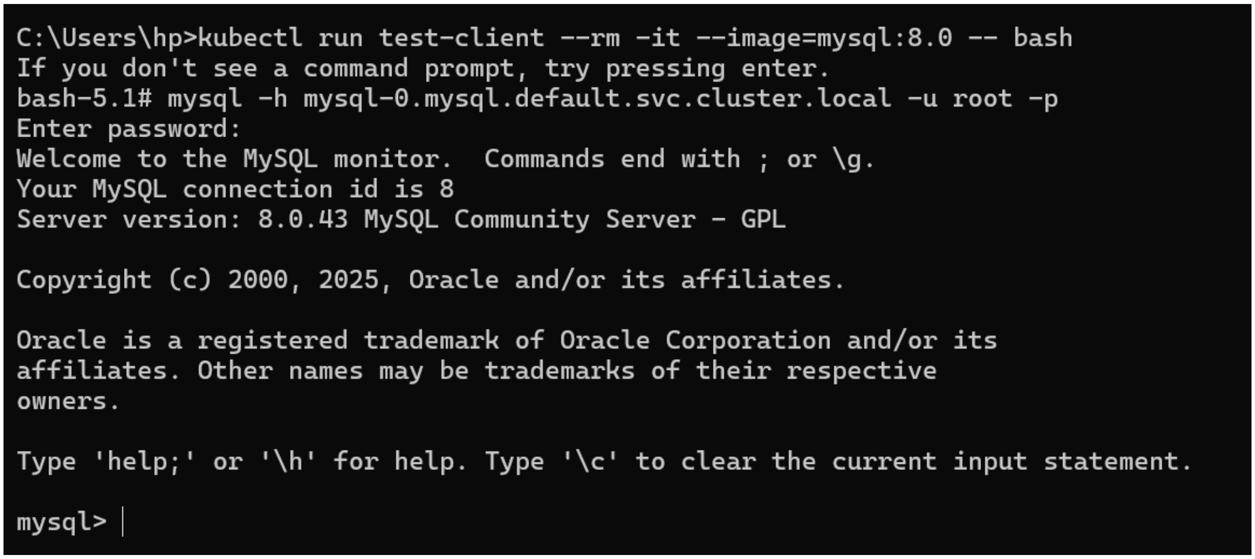
```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  ports:
  - port: 3306
    name: mysql
  clusterIP: None
  selector:
    app: mysql

---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  serviceName: "mysql"
  replicas: 3
  selector:
    matchLabels:
      app: mysql
  template:
```

- kubectl apply -f mysql-statefulset.yaml
- kubectl get statefulsets
- kubectl get pods -l app=mysql
- kubectl describe statefulset mysql



```
C:\Users\hp>kubectl apply -f mysql-statefulset.yaml
service/mysql created
statefulset.apps/mysql created

C:\Users\hp>kubectl get statefulsets
NAME    READY   AGE
mysql   0/3     13s

C:\Users\hp>kubectl get pods -l app=mysql
NAME      READY   STATUS              RESTARTS   AGE
mysql-0   0/1     ContainerCreating   0          22s

C:\Users\hp>kubectl describe statefulset mysql
Name:               mysql
Namespace:          default
CreationTimestamp:  Wed, 27 Aug 2025 10:47:10 +0530
Selector:           app=mysql
Labels:             <none>
Annotations:        <none>
Replicas:           3 desired | 1 total
Update Strategy:    RollingUpdate
  Partition:        0
Pods Status:        0 Running / 1 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=mysql
  Containers:
   mysql:
    Image:      mysql:8.0
    Port:       3306/TCP
    Host Port:  0/TCP
    Environment:
      MYSQL_ROOT_PASSWORD:  my-secret-pw
```

kubectl run test-client --rm -it --image=mysql:8.0 -- bash



```
C:\Users\hp>kubectl run test-client --rm -it --image=mysql:8.0 -- bash
If you don't see a command prompt, try pressing enter.
bash-5.1# mysql -h mysql-0.mysql.default.svc.cluster.local -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.43 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Use a StatefulSet when the app needs a fixed name, its own storage, and must start/stop in order like databases.

Use a Deployment when the app is stateless, and any pod can be replaced anytime like web servers.

🔹 Exercise 8: Horizontal Pod Autoscaling (HPA)

Objective: Scale your application automatically based on metrics.

1. Create a deployment (e.g., a simple HTTP server).

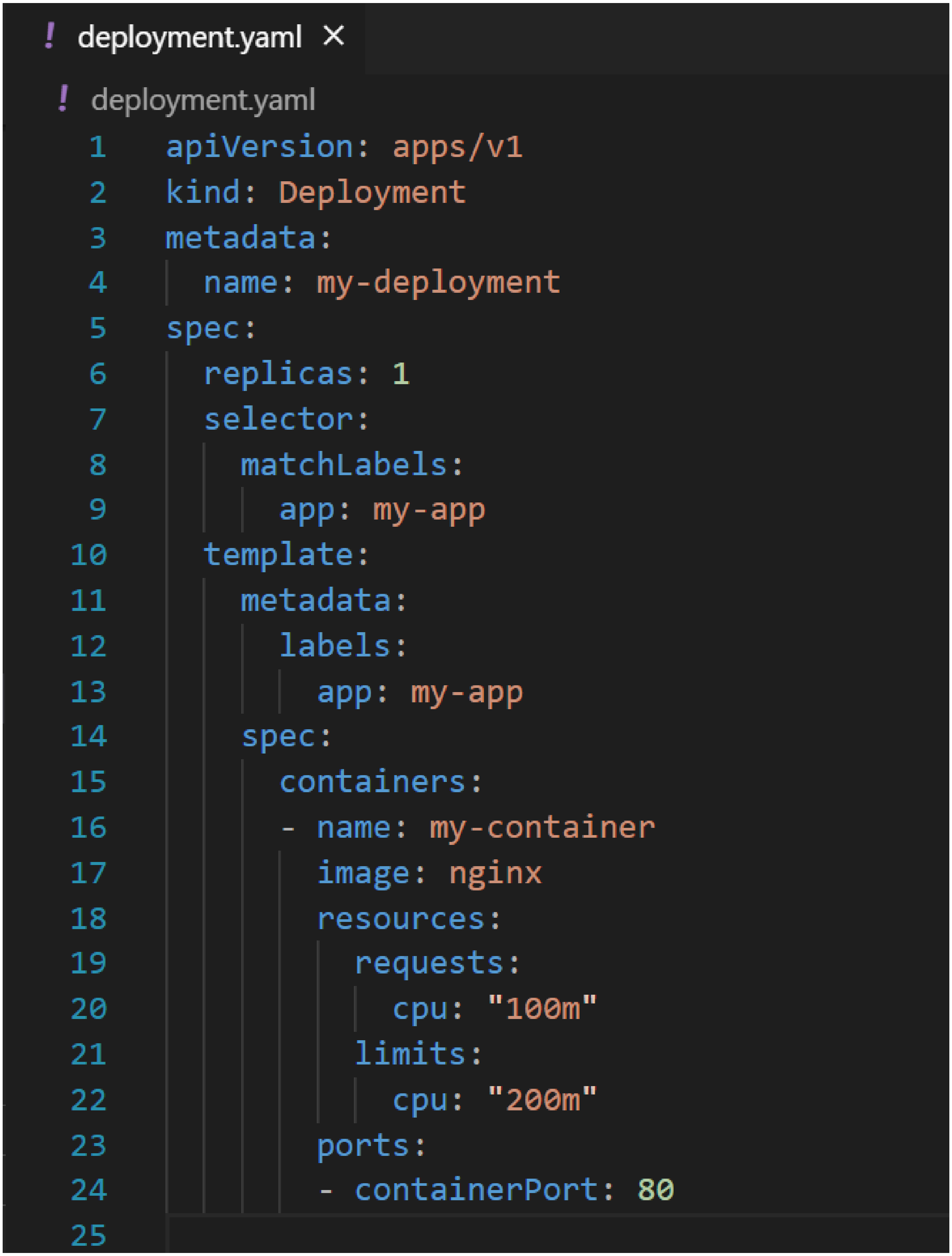
2. Enable metrics server for autoscaling (e.g., Minikube).

3. Create an HPA to scale the deployment based on CPU utilization:

4. Test autoscaling by generating load on the deployment .

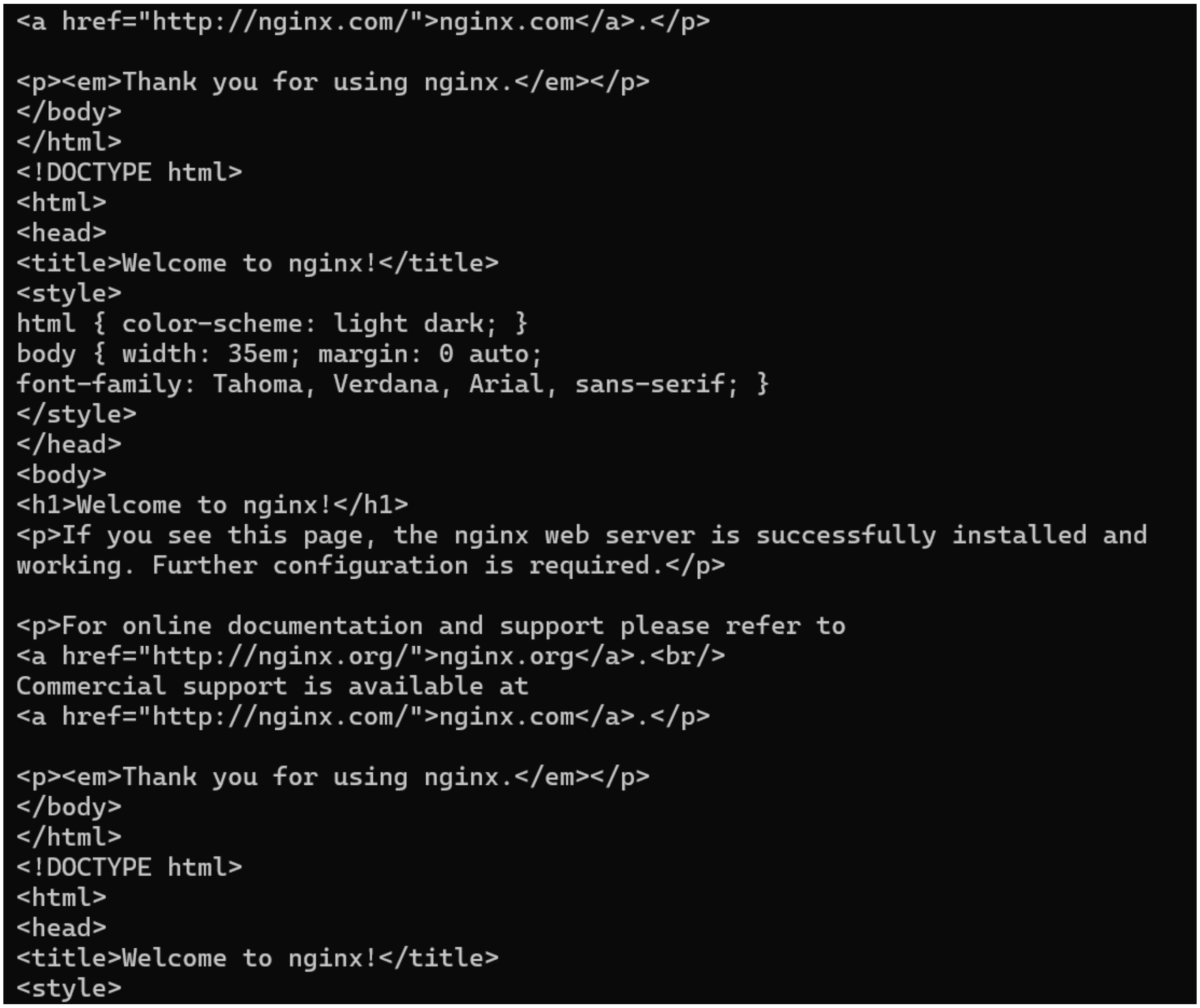
Checkpoint:

● How does the HPA decide when to scale? What metrics are used for scaling?

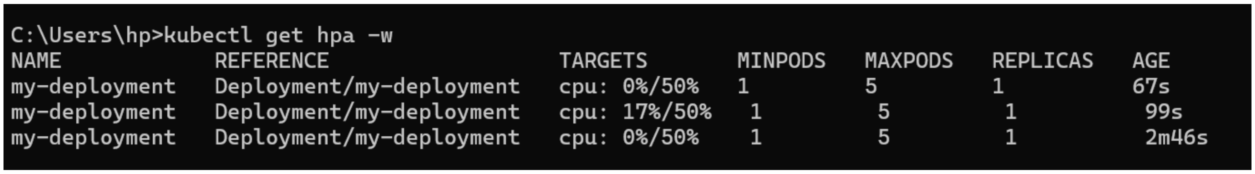


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: nginx
        resources:
          requests:
            cpu: "100m"
          limits:
            cpu: "200m"
        ports:
        - containerPort: 80

```

- kubectl apply -f deployment.yaml
- minikube addons enable metrics-server
- kubectl autoscale deployment my-deployment --cpu-percent=50 --min=1 --max=5
- kubectl expose deployment my-deployment --type=NodePort --port=80
- kubectl run -i --tty load-generator --image=busybox /bin/sh
- #/ while true; do wget -q -O- http://my-deployment.default.svc.cluster.local; done
- kubectl get hpa -w



```
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```



```
C:\Users\hp>kubectl get hpa -w
NAME            REFERENCE                  TARGETS       MINPODS   MAXPODS   REPLICAS   AGE
my-deployment   Deployment/my-deployment   cpu: 0%/50%   1         5         1          67s
my-deployment   Deployment/my-deployment   cpu: 17%/50%   1         5         1          99s
my-deployment   Deployment/my-deployment   cpu: 0%/50%    1         5         1          2m46s
```

- The Horizontal Pod Autoscaler (HPA) monitors metrics (primarily CPU utilization or memory usage) collected by the metrics-server.

- It compares the current usage of the pods against the target threshold you set (e.g., 50% CPU).

- If the average usage across pods is higher than the target, HPA adds more pods (scales up).

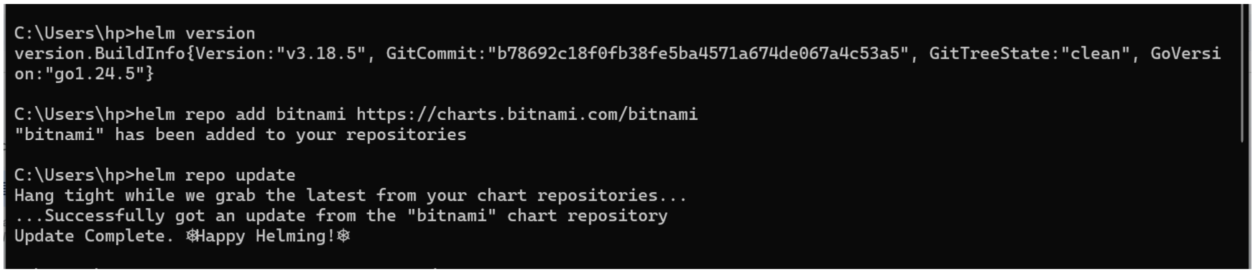
- If the average usage is lower than the target, HPA reduces pods (scales down).

- This ensures applications automatically adapt to varying load without manual intervention.

🔹 Exercise 9: Helm Basics

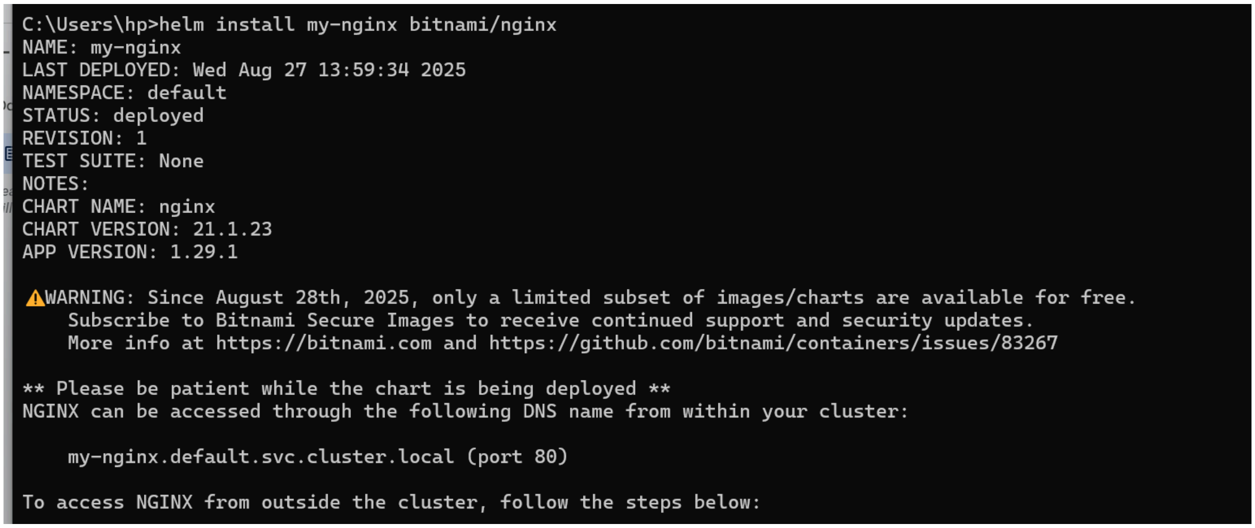
Objective: Use Helm to manage Kubernetes applications.
1. Install Helm on your local machine.
2. Add a Helm chart repository:
3. Install a package from the Helm chart repository, e.g., Nginx:
4. Verify the installation using
Checkpoint:
● What advantages does using Helm offer over manually managing Kubernetes resources with kubectl?

- choco install kubernetes-helm
- helm version
- helm repo add bitnami https://charts.bitnami.com/bitnami
- helm repo update

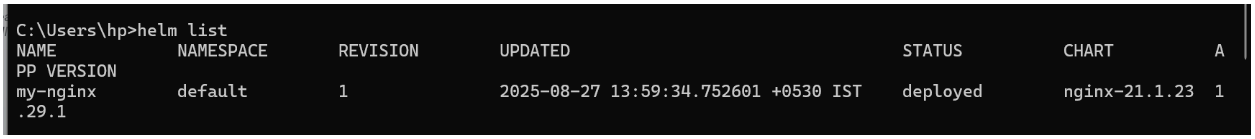


```
C:\Users\hp>helm version
version.BuildInfo{Version:"v3.18.5", GitCommit:"b78692c18f0fb38fe5ba4571a674de067a4c53a5", GitTreeState:"clean", GoVersi
on:"go1.24.5"}

C:\Users\hp>helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories

C:\Users\hp>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "bitnami" chart repository
Update Complete. ⎈Happy Helming!⎈
```

- helm install my-nginx bitnami/nginx
- helm list
- helm status my-nginx



```
C:\Users\hp>helm install my-nginx bitnami/nginx
NAME: my-nginx
LAST DEPLOYED: Wed Aug 27 13:59:34 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 21.1.23
APP VERSION: 1.29.1

⚠WARNING: Since August 28th, 2025, only a limited subset of images/charts are available for free.
    Subscribe to Bitnami Secure Images to receive continued support and security updates.
    More info at https://bitnami.com and https://github.com/bitnami/containers/issues/83267

** Please be patient while the chart is being deployed **
NGINX can be accessed through the following DNS name from within your cluster:

    my-nginx.default.svc.cluster.local (port 80)

To access NGINX from outside the cluster, follow the steps below:
```



```
C:\Users\hp>helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART          A
PP VERSION
my-nginx        default         1               2025-08-27 13:59:34.752601 +0530 IST    deployed        nginx-21.1.23  1
.29.1
```

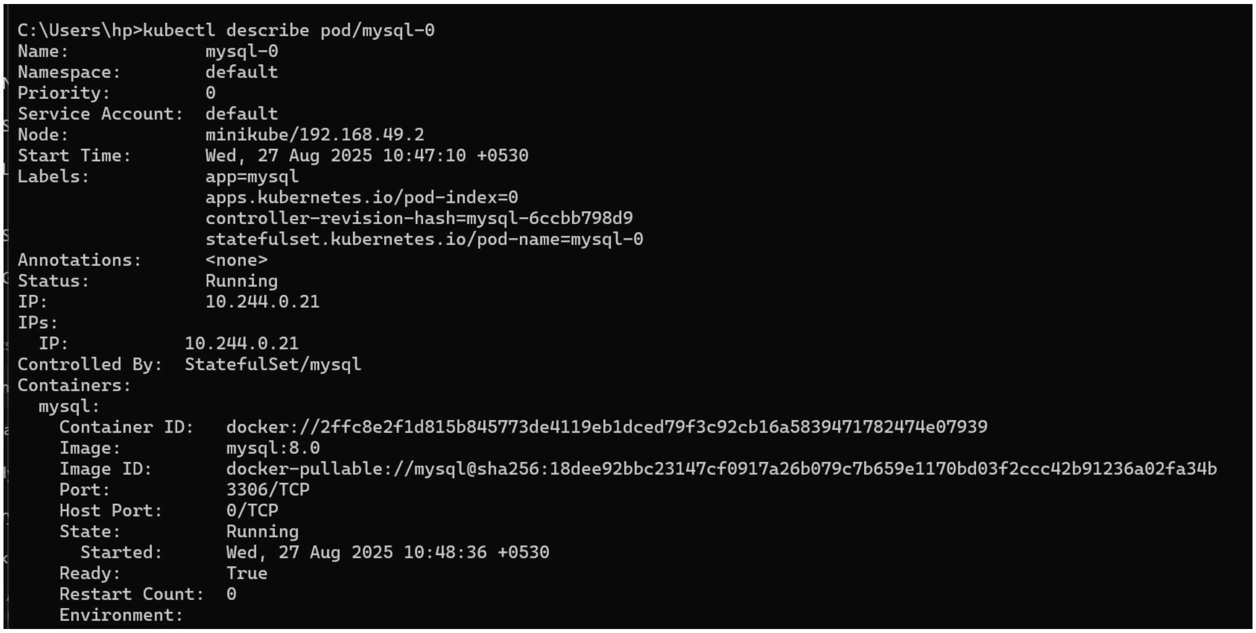
Helm makes deploying apps easier by packaging all Kubernetes resources (YAML files) into one chart, so we don’t have to manage them manually. It also allows quick upgrades, rollbacks, and reusability across environments, which is harder with plain kubectl.

🔹 Exercise 10: Debugging and Troubleshooting

Objective: Learn how to troubleshoot issues in Kubernetes.
1. Identify pod issues using describe command
2. Check the status of nodes and pods
3. View events related to the pod
4. View logs for troubleshooting(pods and deployment)

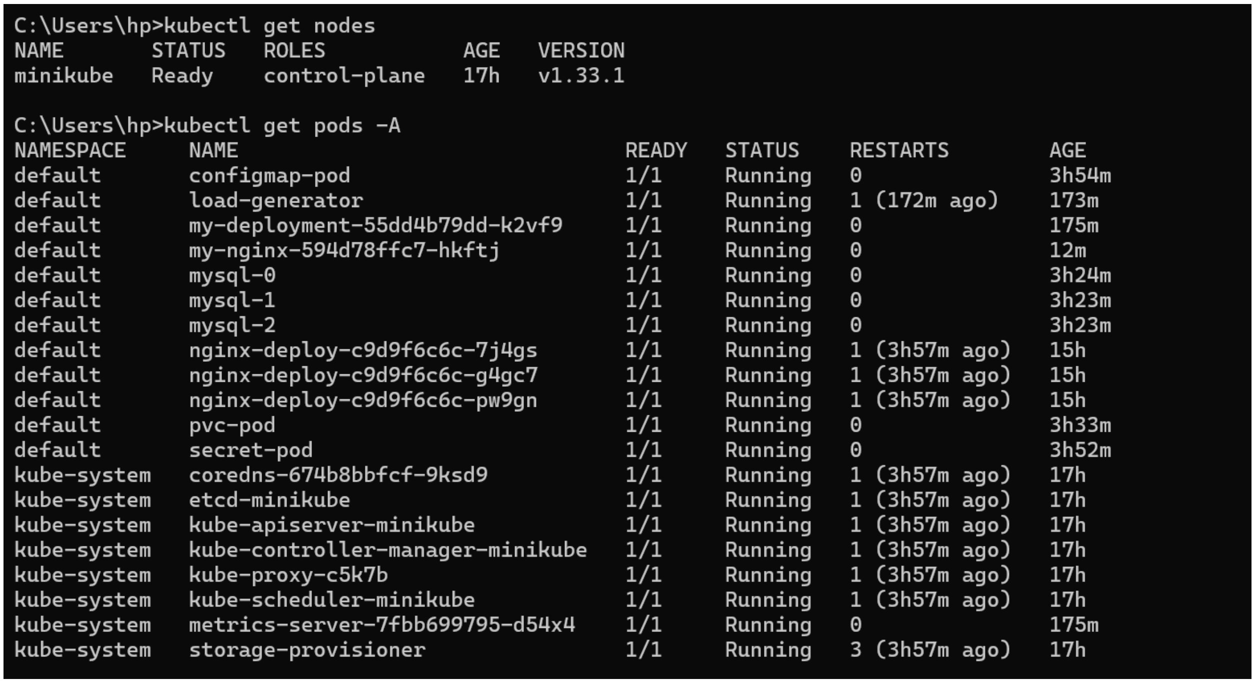
Checkpoint:
● What are some common reasons for a pod being in a CrashLoopBackOff state?

- kubectl describe pod/mysql-0



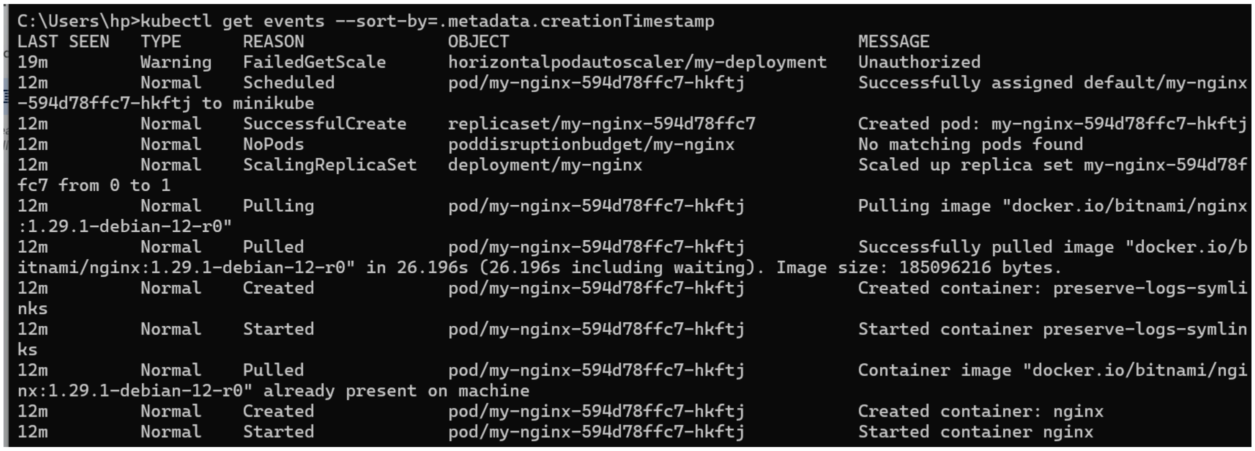
```
C:\Users\hp>kubectl describe pod/mysql-0
Name:             mysql-0
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Wed, 27 Aug 2025 10:47:10 +0530
Labels:           app=mysql
                  apps.kubernetes.io/pod-index=0
                  controller-revision-hash=mysql-6ccbb798d9
                  statefulset.kubernetes.io/pod-name=mysql-0
Annotations:      <none>
Status:           Running
IP:               10.244.0.21
IPs:
  IP:           10.244.0.21
Controlled By:  StatefulSet/mysql
Containers:
  mysql:
    Container ID:   docker://2ffc8e2f1d815b845773de4119eb1dced79f3c92cb16a5839471782474e07939
    Image:          mysql:8.0
    Image ID:       docker-pullable://mysql@sha256:18dee92bbc23147cf0917a26b079c7b659e1170bd03f2ccc42b91236a02fa34b
    Port:           3306/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 27 Aug 2025 10:48:36 +0530
    Ready:          True
    Restart Count:  0
    Environment:
```

- kubectl get nodes
- kubectl get pods -A



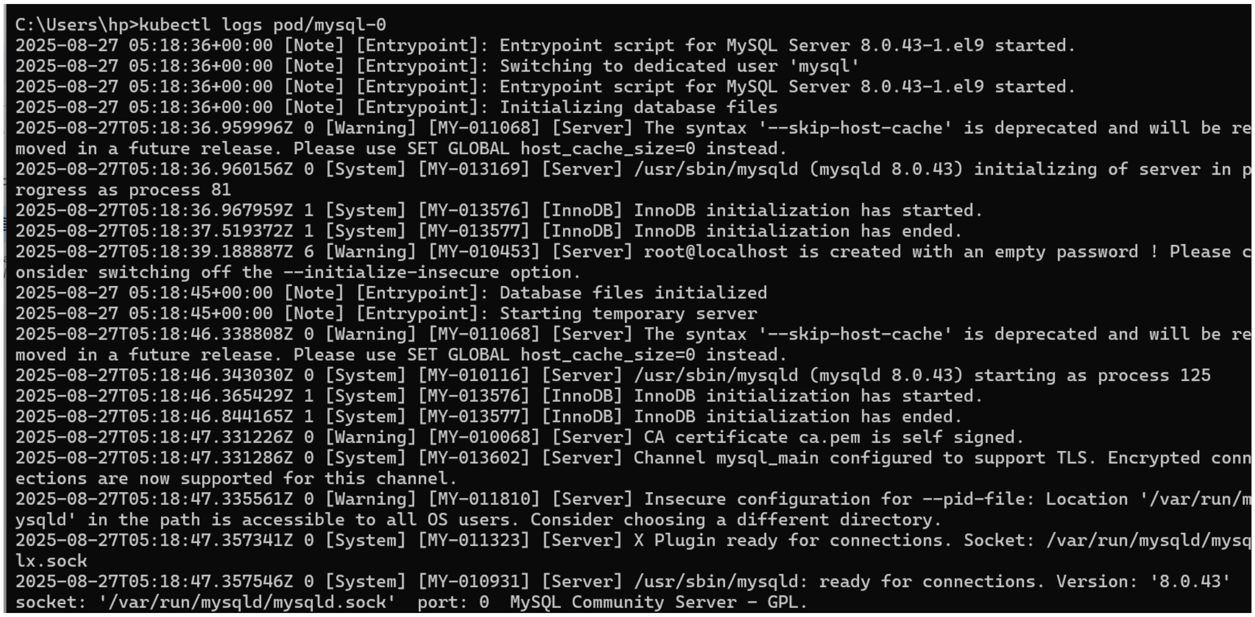
```
C:\Users\hp>kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   17h   v1.33.1

C:\Users\hp>kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS        AGE
default       configmap-pod                      1/1     Running   0               3h54m
default       load-generator                     1/1     Running   1 (172m ago)    173m
default       my-deployment-55dd4b79dd-k2vf9     1/1     Running   0               175m
default       my-nginx-594d78ffc7-hkftj          1/1     Running   0               12m
default       mysql-0                            1/1     Running   0               3h24m
default       mysql-1                            1/1     Running   0               3h23m
default       mysql-2                            1/1     Running   0               3h23m
default       nginx-deploy-c9d9f6c6c-7j4gs       1/1     Running   1 (3h57m ago)   15h
default       nginx-deploy-c9d9f6c6c-g4gc7       1/1     Running   1 (3h57m ago)   15h
default       nginx-deploy-c9d9f6c6c-pw9gn       1/1     Running   1 (3h57m ago)   15h
default       pvc-pod                            1/1     Running   0               3h33m
default       secret-pod                         1/1     Running   0               3h52m
kube-system   coredns-674b8bbfcf-9ksd9           1/1     Running   1 (3h57m ago)   17h
kube-system   etcd-minikube                      1/1     Running   1 (3h57m ago)   17h
kube-system   kube-apiserver-minikube            1/1     Running   1 (3h57m ago)   17h
kube-system   kube-controller-manager-minikube   1/1     Running   1 (3h57m ago)   17h
kube-system   kube-proxy-c5k7b                   1/1     Running   1 (3h57m ago)   17h
kube-system   kube-scheduler-minikube            1/1     Running   1 (3h57m ago)   17h
kube-system   metrics-server-7fbb699795-d54x4    1/1     Running   0               175m
kube-system   storage-provisioner                1/1     Running   3 (3h57m ago)   17h
```

- kubectl get events --sort-by=.metadata.creationTimestamp



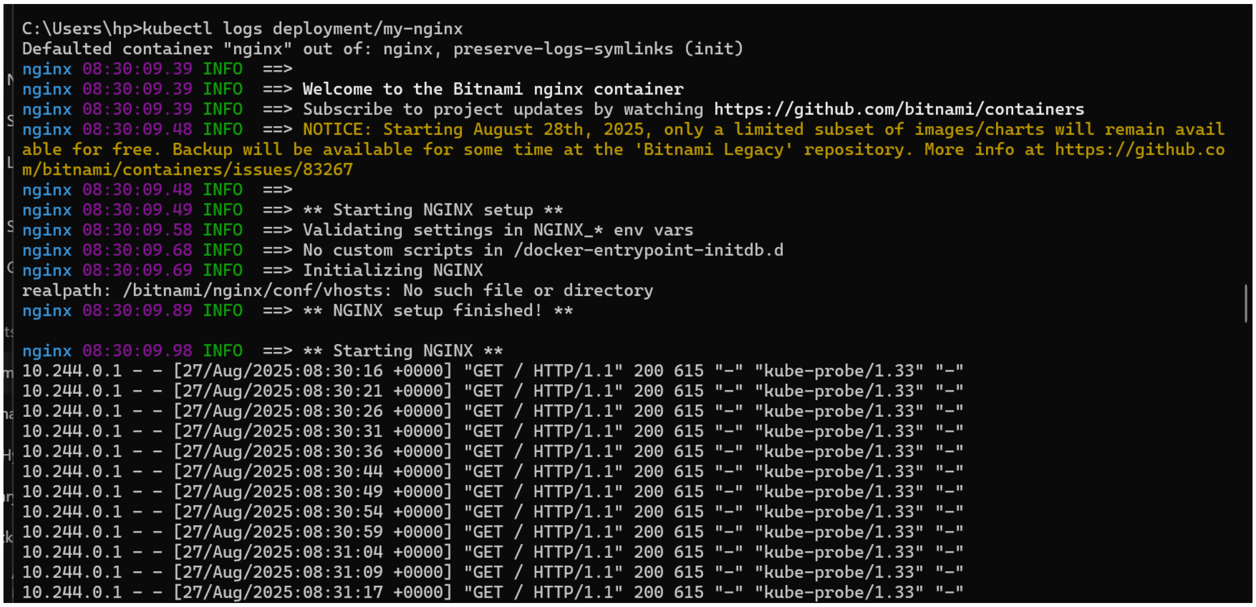
```
C:\Users\hp>kubectl get events --sort-by=.metadata.creationTimestamp
LAST SEEN   TYPE      REASON              OBJECT                                    MESSAGE
19m         Warning   FailedGetScale      horizontalpodautoscaler/my-deployment     Unauthorized
12m         Normal    Scheduled           pod/my-nginx-594d78ffc7-hkftj             Successfully assigned default/my-nginx
-594d78ffc7-hkftj to minikube
12m         Normal    SuccessfulCreate    replicaset/my-nginx-594d78ffc7            Created pod: my-nginx-594d78ffc7-hkftj
12m         Normal    NoPods              poddisruptionbudget/my-nginx              No matching pods found
12m         Normal    ScalingReplicaSet   deployment/my-nginx                       Scaled up replica set my-nginx-594d78f
fc7 from 0 to 1
12m         Normal    Pulling             pod/my-nginx-594d78ffc7-hkftj             Pulling image "docker.io/bitnami/nginx
:1.29.1-debian-12-r0"
12m         Normal    Pulled              pod/my-nginx-594d78ffc7-hkftj             Successfully pulled image "docker.io/b
itnami/nginx:1.29.1-debian-12-r0" in 26.196s (26.196s including waiting). Image size: 185096216 bytes.
12m         Normal    Created             pod/my-nginx-594d78ffc7-hkftj             Created container: preserve-logs-symli
nks
12m         Normal    Started             pod/my-nginx-594d78ffc7-hkftj             Started container preserve-logs-symlin
ks
12m         Normal    Pulled              pod/my-nginx-594d78ffc7-hkftj             Container image "docker.io/bitnami/ngi
nx:1.29.1-debian-12-r0" already present on machine
12m         Normal    Created             pod/my-nginx-594d78ffc7-hkftj             Created container: nginx
12m         Normal    Started             pod/my-nginx-594d78ffc7-hkftj             Started container nginx
```

- kubectl logs pod/mysql-0



```
C:\Users\hp>kubectl logs pod/mysql-0
2025-08-27 05:18:36+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.43-1.el9 started.
2025-08-27 05:18:36+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2025-08-27 05:18:36+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.43-1.el9 started.
2025-08-27 05:18:36+00:00 [Note] [Entrypoint]: Initializing database files
2025-08-27T05:18:36.959996Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be re
moved in a future release. Please use SET GLOBAL host_cache_size=0 instead.
2025-08-27T05:18:36.960156Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.43) initializing of server in p
rogress as process 81
2025-08-27T05:18:36.967959Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2025-08-27T05:18:37.519372Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2025-08-27T05:18:39.188887Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please c
onsider switching off the --initialize-insecure option.
2025-08-27 05:18:45+00:00 [Note] [Entrypoint]: Database files initialized
2025-08-27 05:18:45+00:00 [Note] [Entrypoint]: Starting temporary server
2025-08-27T05:18:46.338808Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be re
moved in a future release. Please use SET GLOBAL host_cache_size=0 instead.
2025-08-27T05:18:46.343030Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.43) starting as process 125
2025-08-27T05:18:46.365429Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2025-08-27T05:18:46.844165Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2025-08-27T05:18:47.331226Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2025-08-27T05:18:47.331286Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted conn
ections are now supported for this channel.
2025-08-27T05:18:47.335561Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/m
ysqld' in the path is accessible to all OS users. Consider choosing a different directory.
2025-08-27T05:18:47.357341Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld/mysq
lx.sock
2025-08-27T05:18:47.357546Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.43'
socket: '/var/run/mysqld/mysqld.sock'  port: 0  MySQL Community Server - GPL.
```

- kubectl logs deployment/my-nginx



```
C:\Users\hp>kubectl logs deployment/my-nginx
Defaulted container "nginx" out of: nginx, preserve-logs-symlinks (init)
nginx 08:30:09.39 INFO  ==>
nginx 08:30:09.39 INFO  ==> Welcome to the Bitnami nginx container
nginx 08:30:09.39 INFO  ==> Subscribe to project updates by watching https://github.com/bitnami/containers
nginx 08:30:09.48 INFO  ==> NOTICE: Starting August 28th, 2025, only a limited subset of images/charts will remain avail
able for free. Backup will be available for some time at the 'Bitnami Legacy' repository. More info at https://github.co
m/bitnami/containers/issues/83267
nginx 08:30:09.48 INFO  ==>
nginx 08:30:09.49 INFO  ==> ** Starting NGINX setup **
nginx 08:30:09.58 INFO  ==> Validating settings in NGINX_* env vars
nginx 08:30:09.68 INFO  ==> No custom scripts in /docker-entrypoint-initdb.d
nginx 08:30:09.69 INFO  ==> Initializing NGINX
realpath: /bitnami/nginx/conf/vhosts: No such file or directory
nginx 08:30:09.89 INFO  ==> ** NGINX setup finished! **

nginx 08:30:09.98 INFO  ==> ** Starting NGINX **
10.244.0.1 - - [27/Aug/2025:08:30:16 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:21 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:26 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:31 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:36 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:44 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:49 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:54 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:30:59 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:31:04 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:31:09 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
10.244.0.1 - - [27/Aug/2025:08:31:17 +0000] "GET / HTTP/1.1" 200 615 "-" "kube-probe/1.33" "-"
```

Common reasons for a pod being in CrashLoopBackOff:

- The app inside the container keeps crashing because of errors in code or wrong configuration.

- Wrong or missing environment variables, secrets, or config maps.

- Issues with the container image.

- Not enough memory/CPU, causing the pod to restart.

- The pod is trying to connect to another service (like a database) which isn’t ready yet.