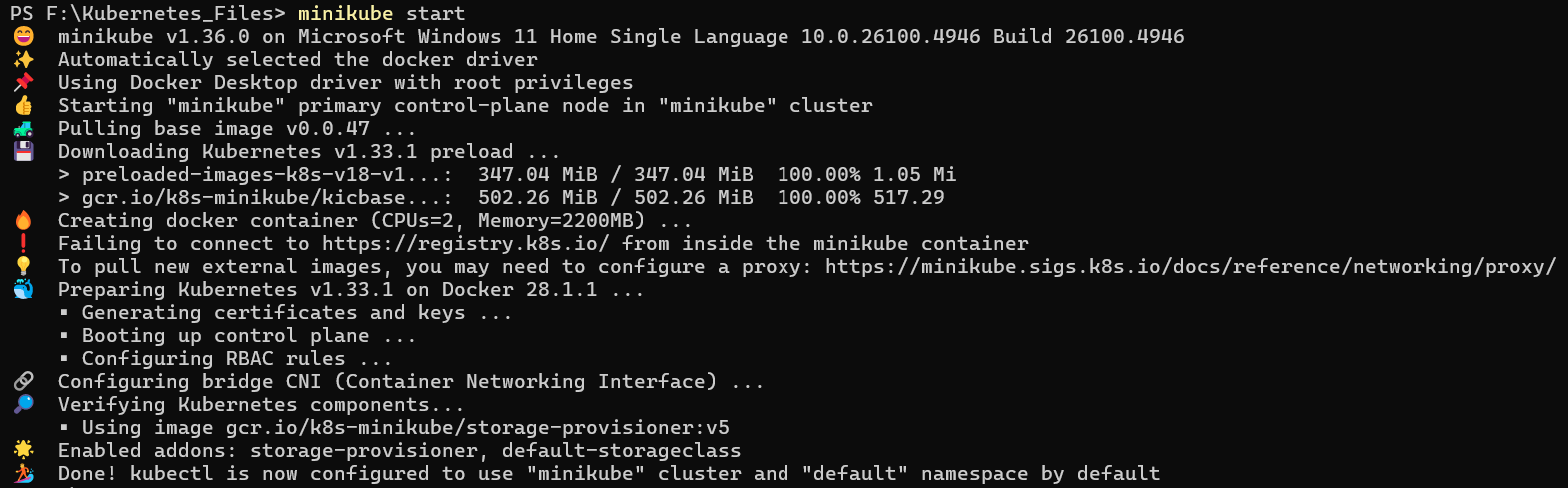
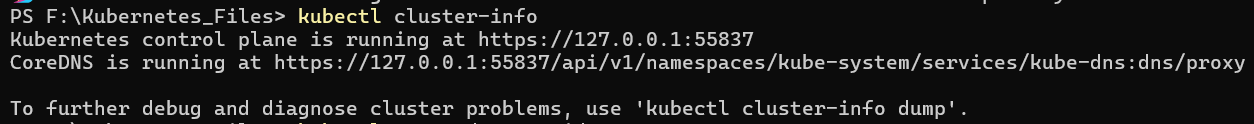
Exercise 1 - Setting up your Kubernetes Cluster

Commands :

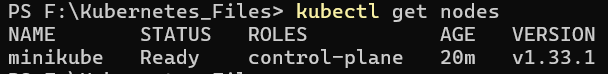
* minikube start



* kubectl cluster-info



* kubectl get nodes



Exercise 2 - Creating and Managing Pods

Commands :

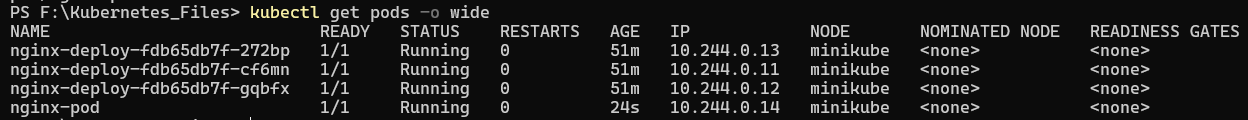
1. Create a pod :

* kubectl apply -f manifests/ex2-pod.yaml

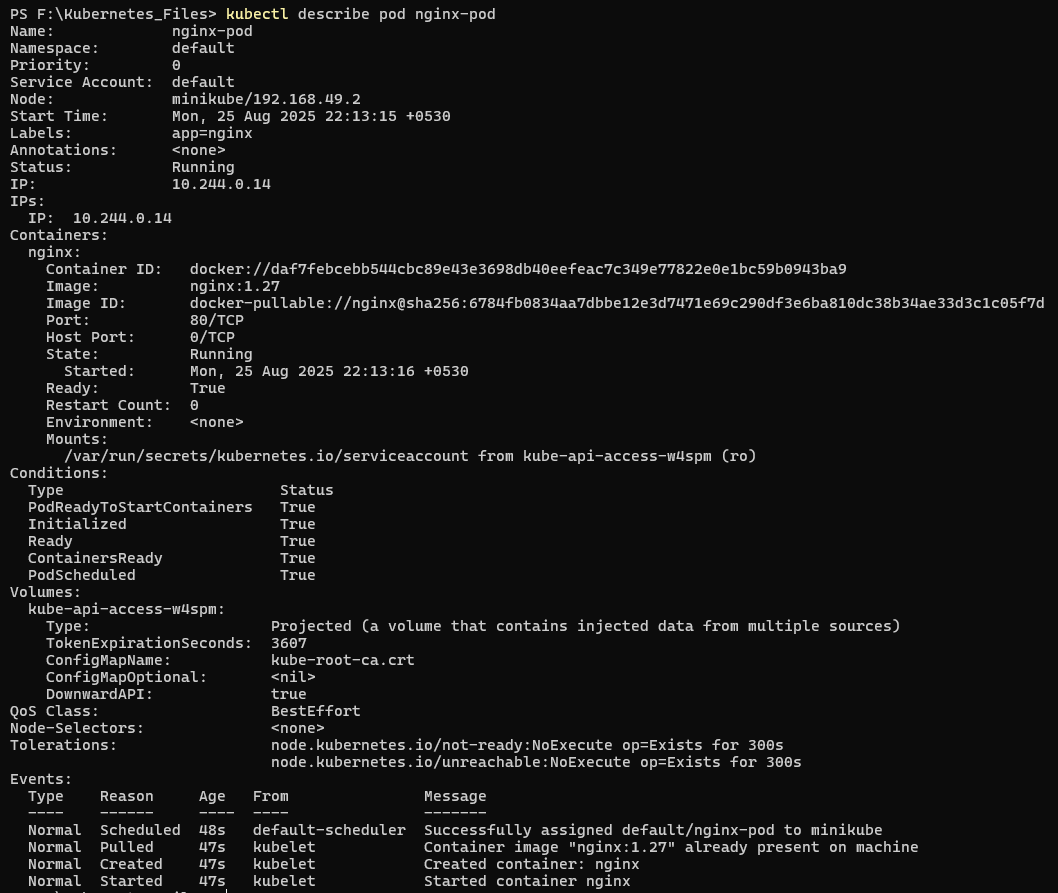


1. Check pod status:

* kubectl get pods -o wide

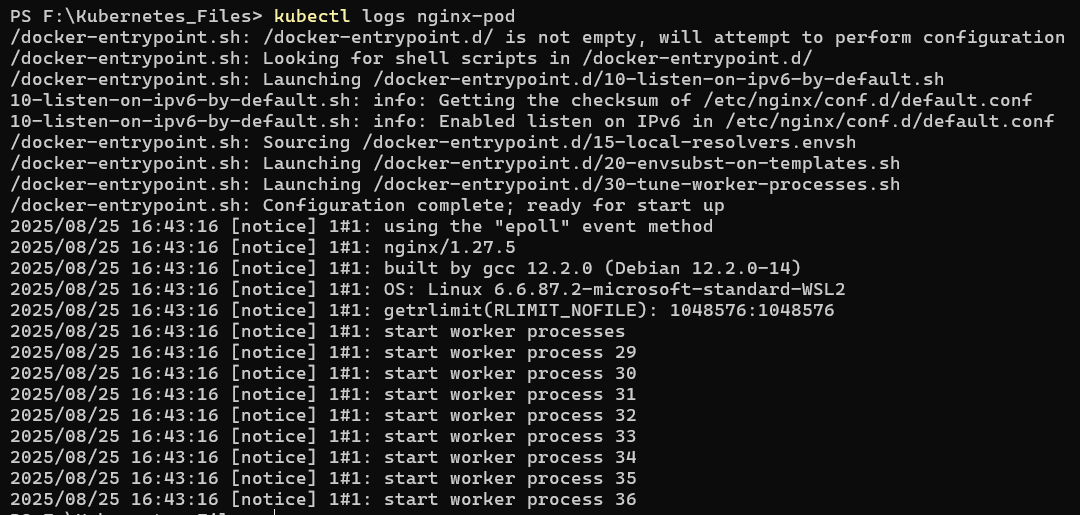


* kubectl describe pod nginx-pod

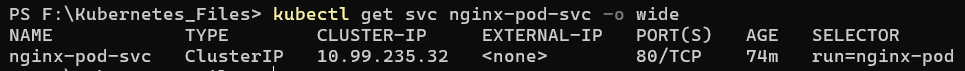


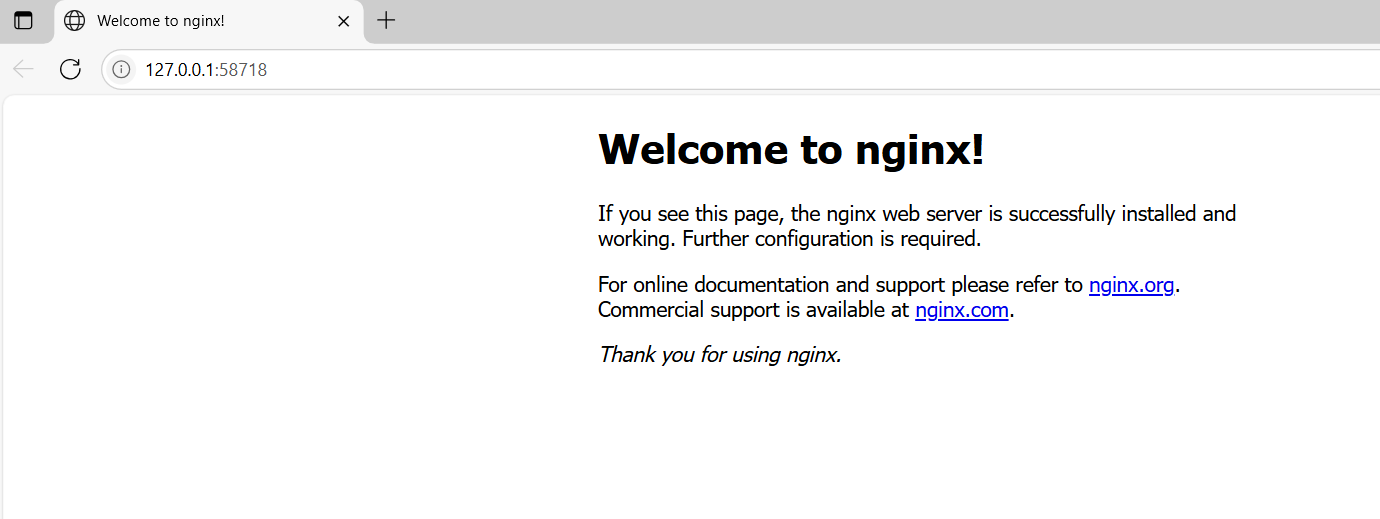
1. View logs:

* kubectl logs nginx-pod



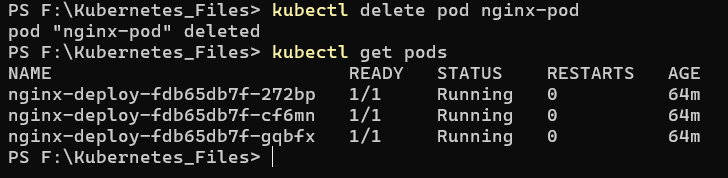
1. Expose the pod via a service and get URL:

* kubectl expose pod nginx-pod --type=ClusterIP --port=80 --name=nginx-pod-svc
* kubectl get svc nginx-pod-svc -o wide
* minikube service nginx-pod-svc –url



1. Checkpoint test - delete pod:

* kubectl delete pod nginx-pod
* kubectl get pods



**Observation :** Deleting this standalone Pod removes it permanently and there is no controller to recreate it.

Exercise 3 — Working with Deployments

Commands:

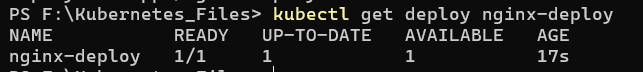
1. Create a deployment :

* kubectl apply -f manifests/ex3-deployment.yaml



1. Scale to 3 replicas:

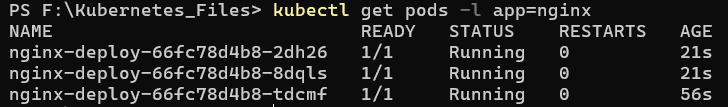
* kubectl get deploy nginx-deploy



* kubectl scale deployment/nginx-deploy --replicas=3



* kubectl get pods -l app=nginx

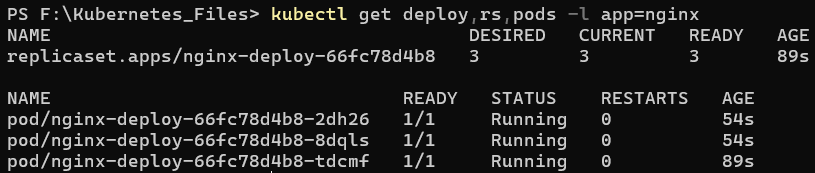


1. Verify deployment & rollout:

* kubectl rollout status deployment/nginx-deploy

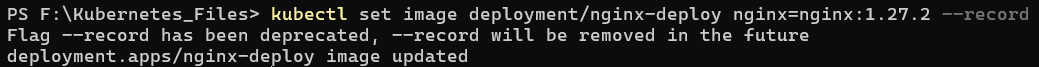


* kubectl get deploy,rs,pods -l app=nginx



1. Update deployment image (imperative) and record:

* kubectl set image deployment/nginx-deploy nginx=nginx:1.27.2 –record

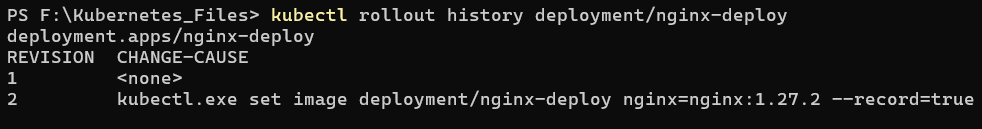


* kubectl rollout status deployment/nginx-deploy



1. View rollout history and rollback:

* kubectl rollout history deployment/nginx-deploy



* kubectl rollout undo deployment/nginx-deploy --to-revision=1



Exercise 4 - Services and Networking

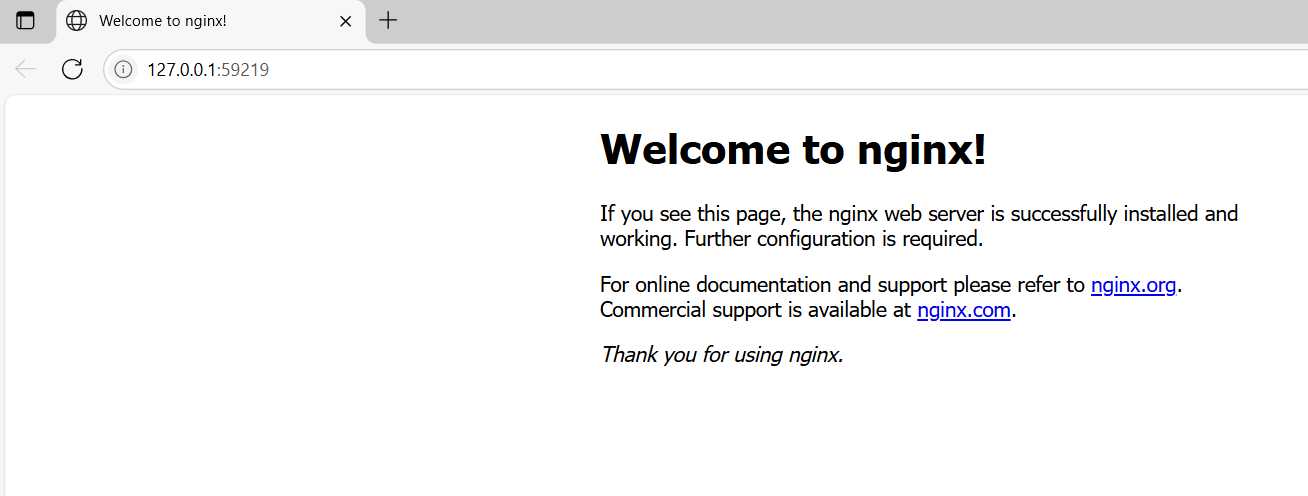
Commands :

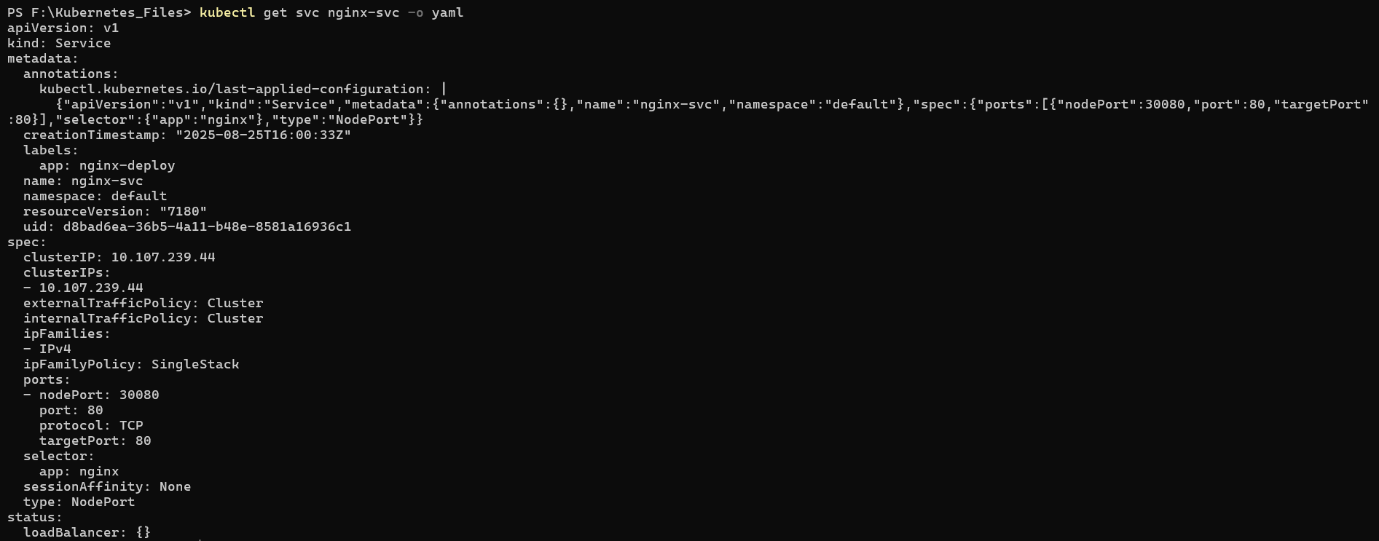
1. Create a NodePort service and get URL :

* kubectl apply -f manifests/ex4-service-nodeport.yaml



* minikube service nginx-svc –url



* kubectl get svc nginx-svc -o yaml

Checkpoint :

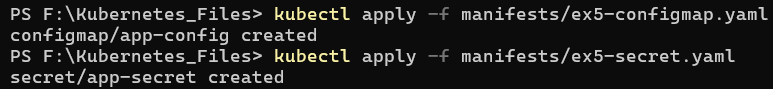
* **ClusterIP:** internal-only, default for inter-service comms.
* **NodePort:** exposes a static port on every node good for dev, bare-metal, or quick external exposure.
* **LoadBalancer:** requests a cloud provider external IP and LB—use in managed clouds for production.

Exercise 5 - ConfigMaps and Secrets

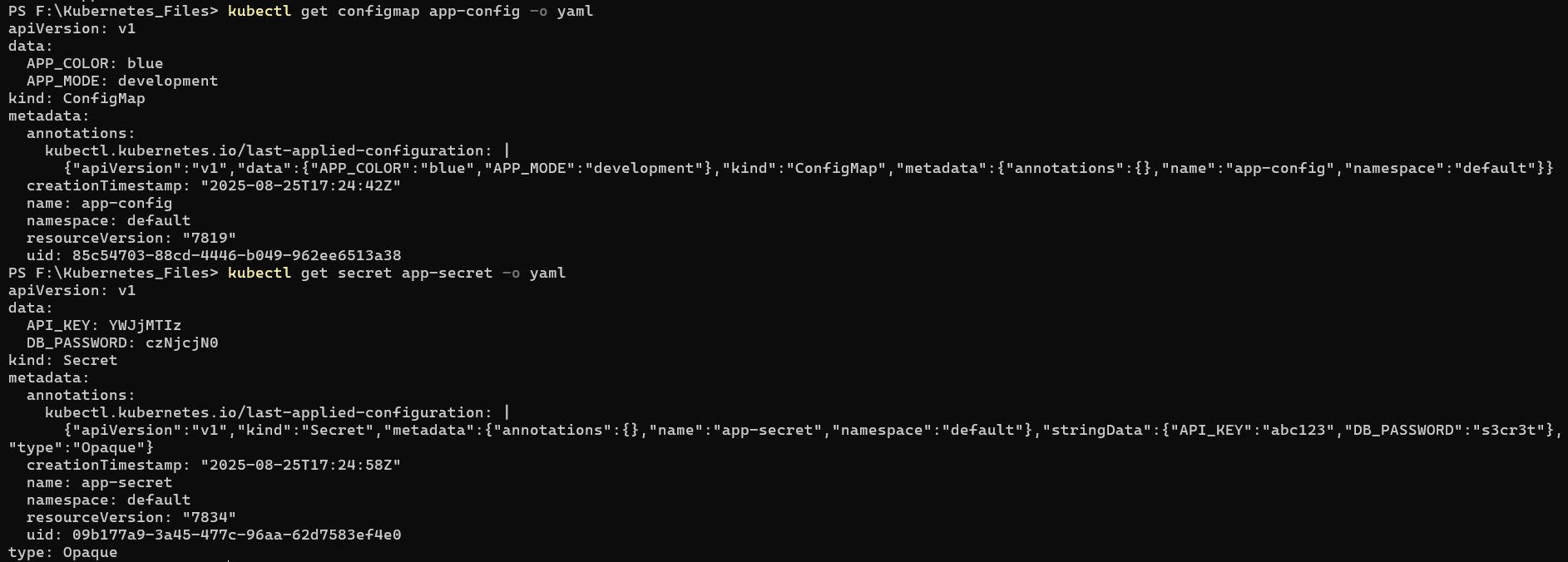
Commands :

1. Create ConfigMap & Secret:

* kubectl apply -f manifests/ex5-configmap.yaml
* kubectl apply -f manifests/ex5-secret.yaml

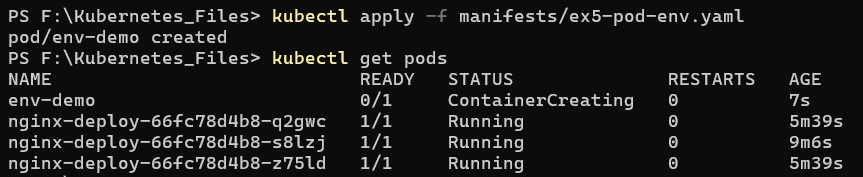


* kubectl get configmap app-config -o yaml
* kubectl get secret app-secret -o yaml

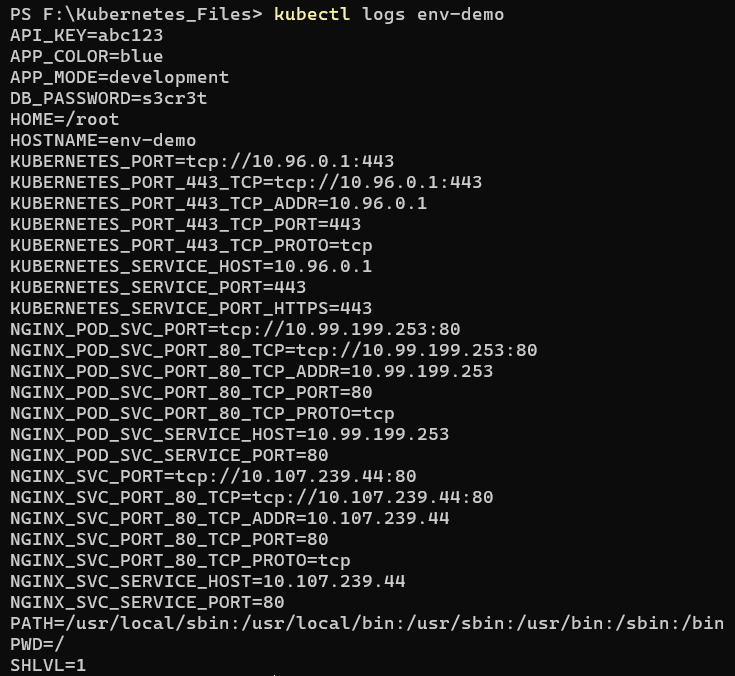


1. Create a pod that consumes them :

* kubectl apply -f manifests/ex5-pod-env.yaml
* kubectl get pods

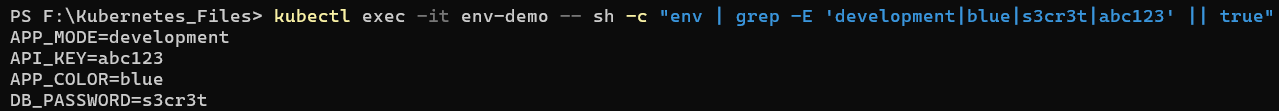


* kubectl logs env-demo



1. Inside pod, verify env:

* kubectl exec -it env-demo -- sh -c "env | grep -E 'APP\_MODE|APP\_COLOR|DB\_PASSWORD|API\_KEY' || true"



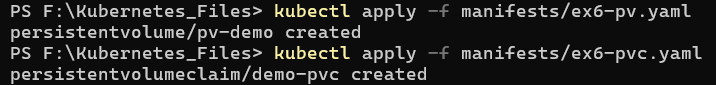
**Checkpoint :**  
Access ConfigMaps/Secrets by injecting them as environment variables or mount them as files in a volume.

Exercise 6 - Persistent Volumes (PV) & Persistent Volume Claims

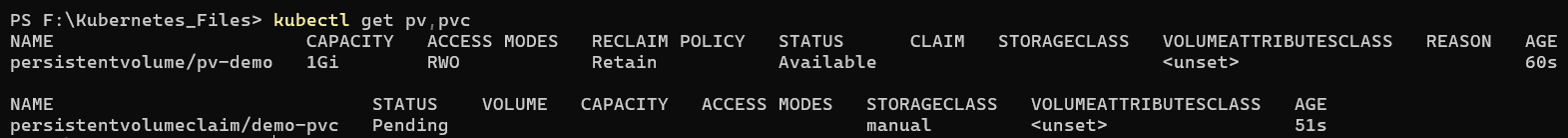
Commands :

1. Create PV and PVC from YAML:

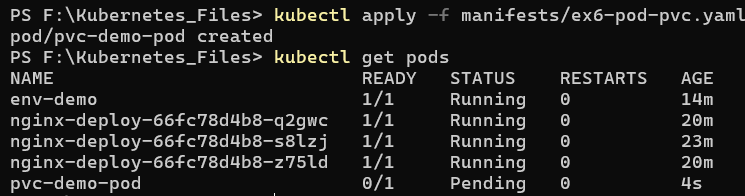
* kubectl apply -f manifests/ex6-pv.yaml
* kubectl apply -f manifests/ex6-pvc.yaml

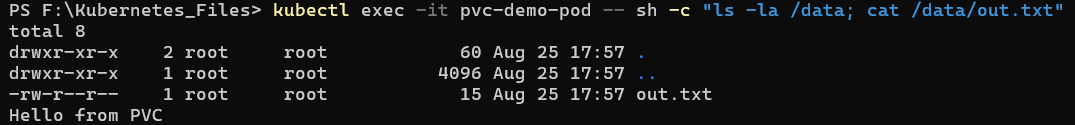


* kubectl get pv,pvc



1. Create a pod that mounts the PVC:

* kubectl apply -f manifests/ex6-pod-pvc.yaml
* kubectl get pods
* kubectl exec -it pvc-demo-pod -- sh -c "ls -la /data; cat /data/out.txt"



1. Restart pod and verify persistence:

* kubectl delete pod pvc-demo-pod



* kubectl apply -f manifests/ex6-pod-pvc.yaml



* kubectl exec -it pvc-demo-pod -- sh -c "cat /data/out.txt"



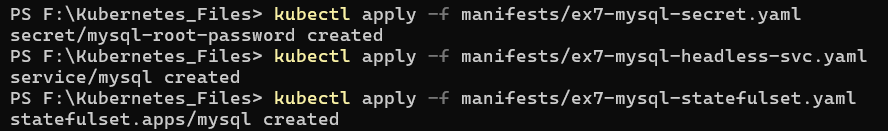
**Checkpoint :**  
Deleting the PVC releases the claim. Whether the underlying PV/storage is deleted depends on the PV’s persistentVolumeReclaimPolicy. If Retain, the PV remains (data preserved); if Delete the underlying storage is deleted.

Exercise 7 - Stateful Sets

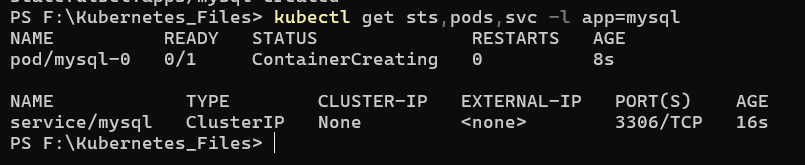
Commands :

1. Apply secret, headless service, and Stateful Set:

* kubectl apply -f manifests/ex7-mysql-secret.yaml
* kubectl apply -f manifests/ex7-mysql-headless-svc.yaml
* kubectl apply -f manifests/ex7-mysql-statefulset.yaml



* kubectl get sts,pods,svc -l app=mysql



1. Access pod by stable DNS (mysql-0.mysql):

* kubectl run -it mysql-client --image=mysql:8.0 --restart=Never --rm -- mysql -h mysql-0.mysql -uroot -prootpass123 -e "SELECT VERSION();"



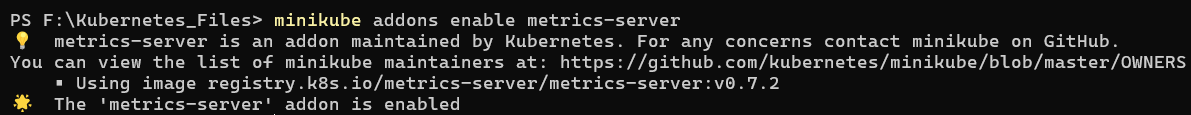
**Checkpoint:**  
StatefulSets provide stable network IDs and stable persistent storage. Use Stateful Sets for stateful workloads (databases, message queues). Deployments are for stateless, replaceable pods.

Exercise 8 - Horizontal Pod Autoscaling (HPA)

Commands:

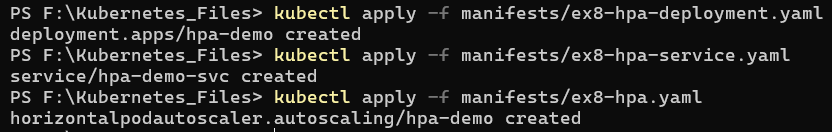
1. Enable metrics server:

* minikube addons enable metrics-server



1. Deploy HPA example:

* kubectl apply -f manifests/ex8-hpa-deployment.yaml
* kubectl apply -f manifests/ex8-hpa-service.yaml
* kubectl apply -f manifests/ex8-hpa.yaml



* kubectl get hpa



1. Watch HPA:

* kubectl get hpa -w



**Checkpoint :**  
HPA looks at metrics. It compares the observed metric against the target and scales pods up/down to match the target.

Exercise 9 - Helm Basics

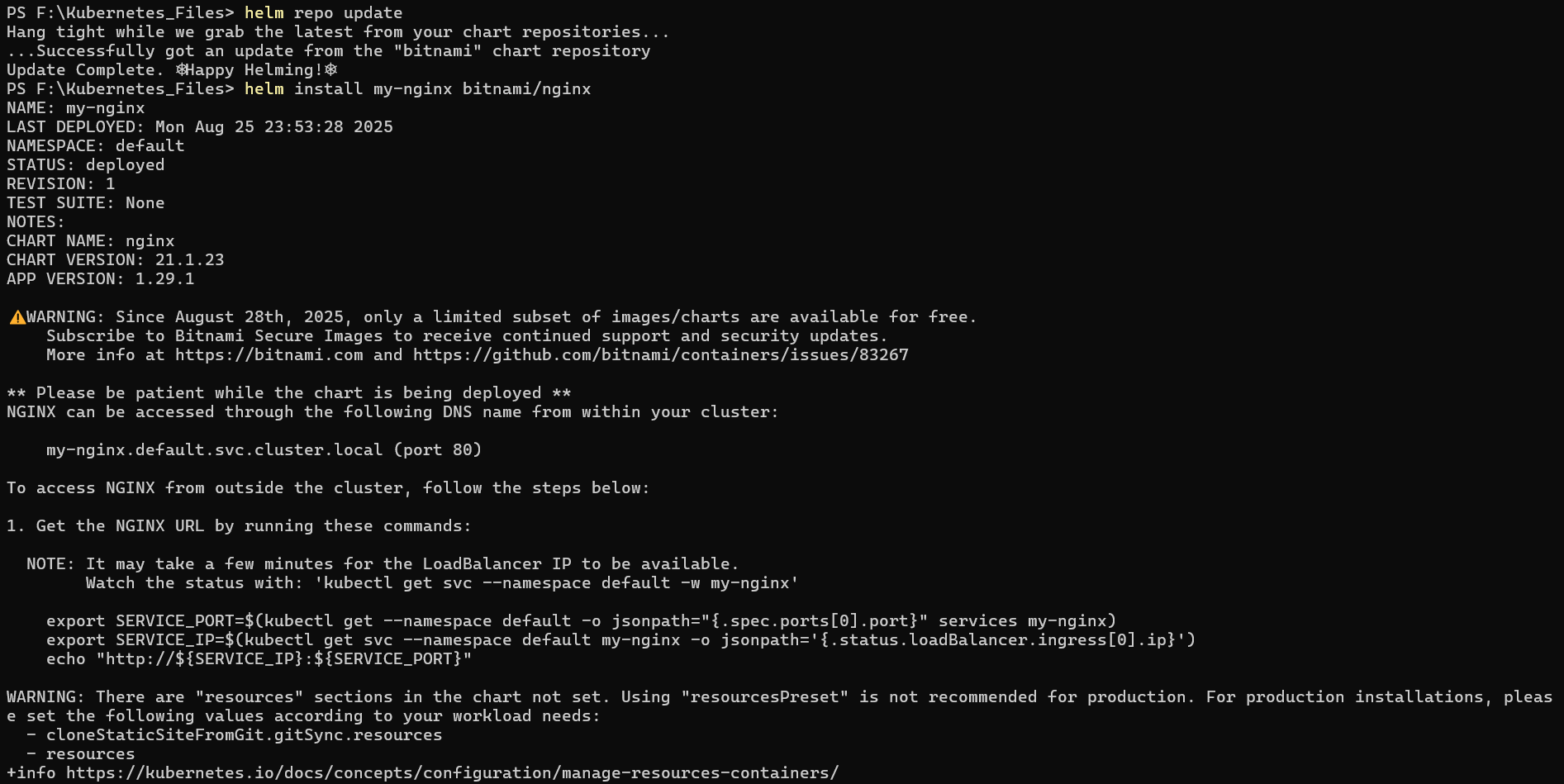
Commands:

1. Install Helm (Helm docs), then:

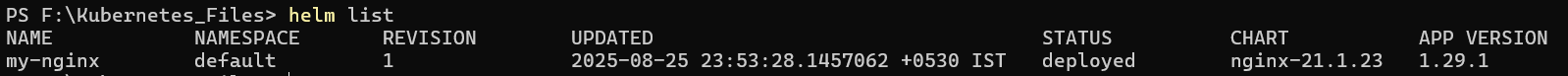
* helm repo add bitnami <https://charts.bitnami.com/bitnami>



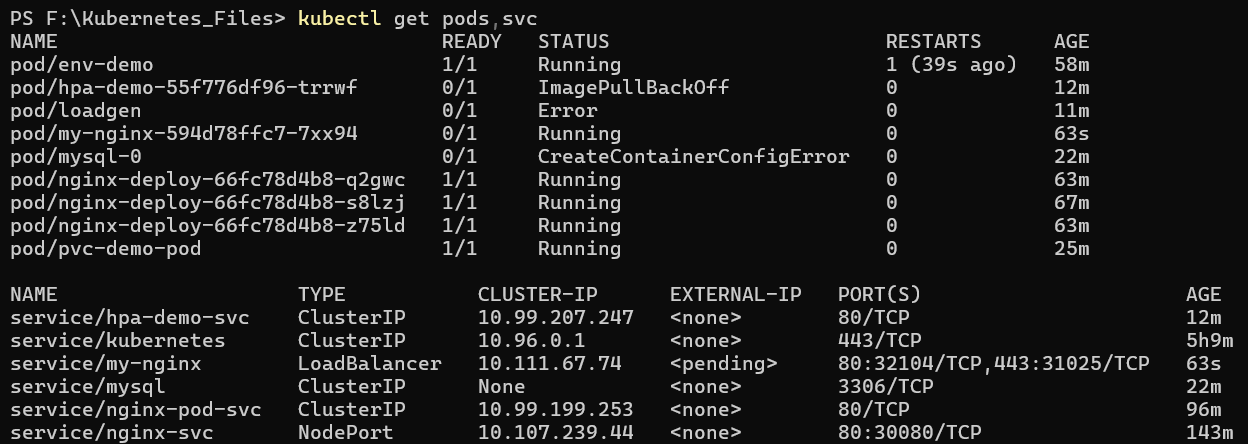
* helm repo update
* helm install my-nginx bitnami/nginx



* helm list



* kubectl get pods,svc

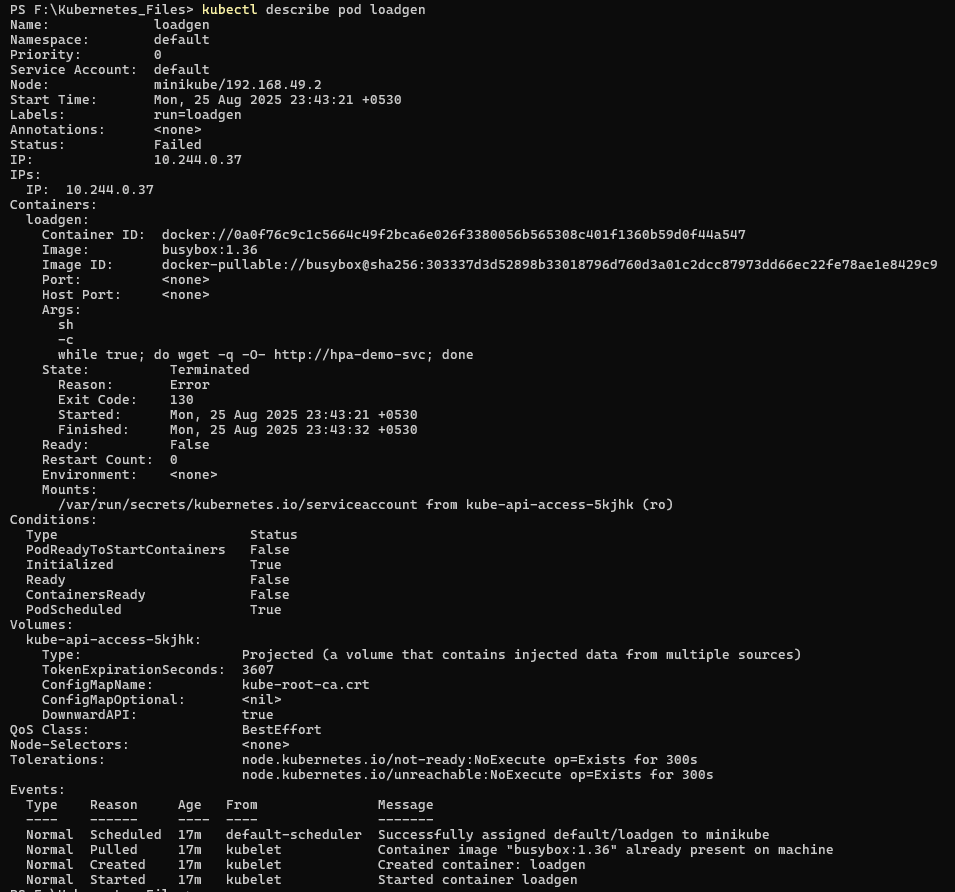


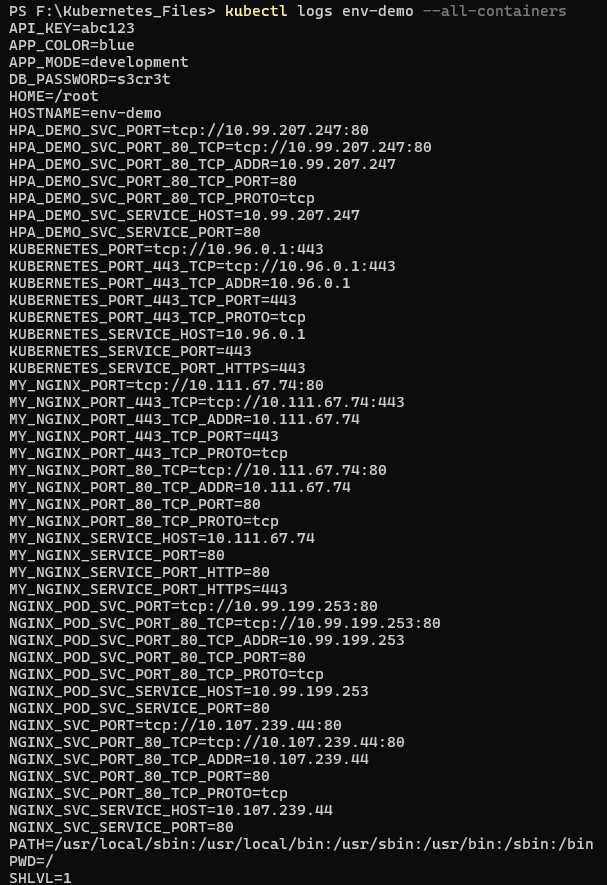
**Checkpoint:**  
Helm packages Kubernetes resources as charts with templating, values and versioning, simplifying installs, upgrades, and rollbacks vs manually managing YAML and kubectl apply.

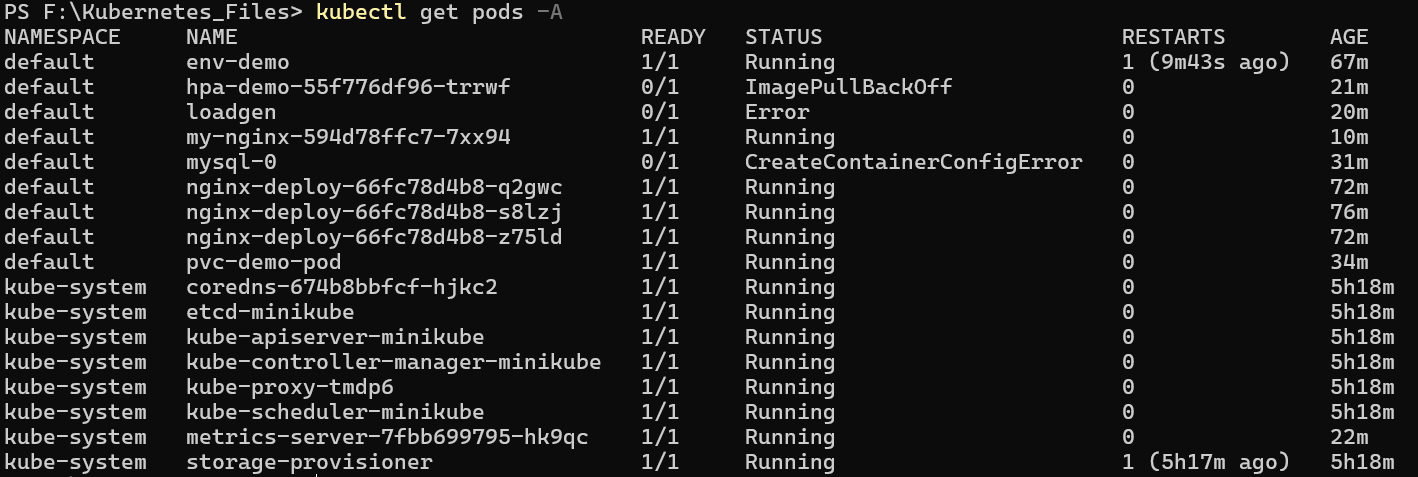
Exercise 10 — Debugging and Troubleshooting

Commands:

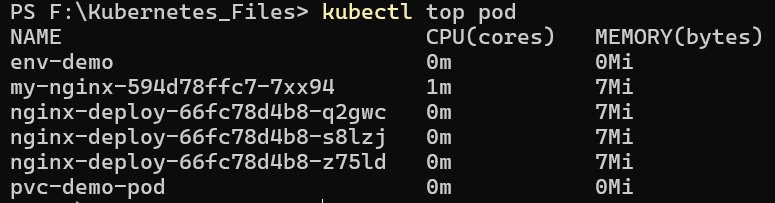
* kubectl describe pod <pod-name>



* kubectl logs <pod-name> --all-containers
* kubectl get pods -A



* kubectl top pod



**Checkpoint:**  
Common CrashLoopBackOff causes: application crashes on start, failed readiness/liveness probes, missing configuration/secrets, incorrect volume mounts, OOMKilled due to resource limits, or image pull failures.

Capstone Project - Deploy production-grade multi-container app (Fast API + Nginx sidecar)

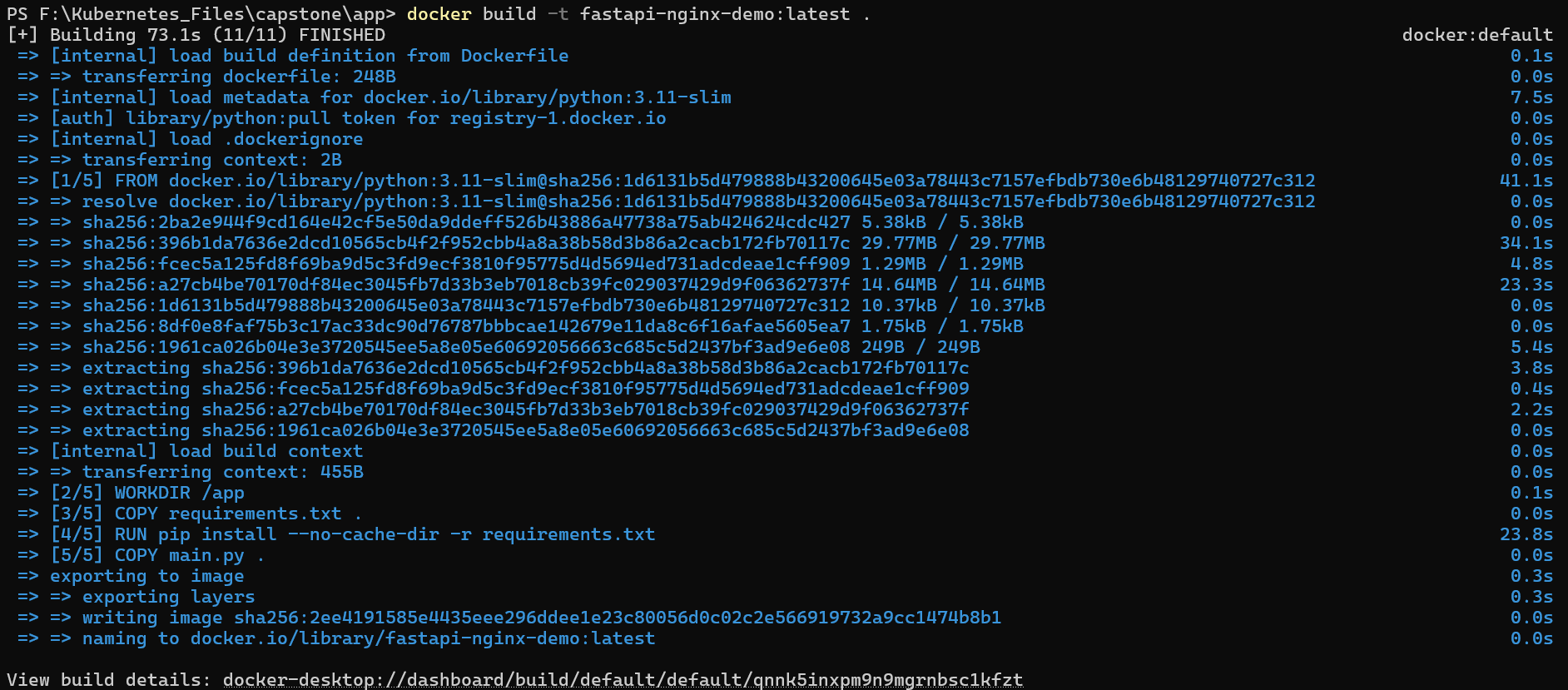
App files (simple FastAPI)

* main.py serves /, /healthz, /echo/{msg}
* Dockerfile provided (simple Python image)

Steps to build & deploy on Minikube

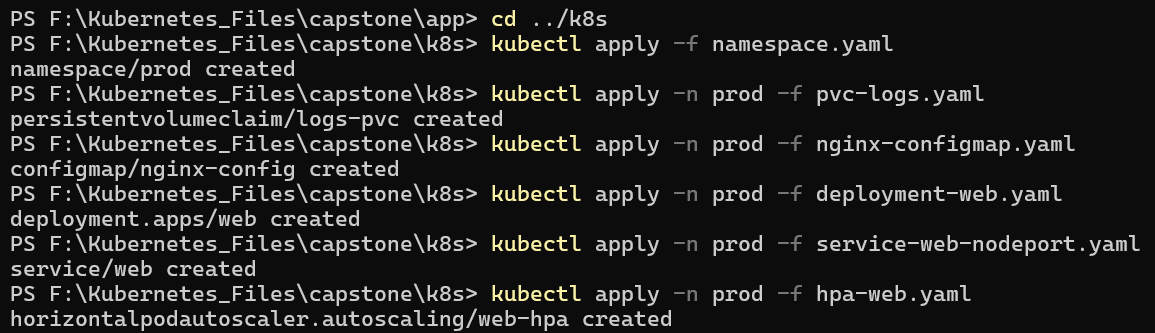
1. Build the Docker image inside Minikube’s Docker daemon:

Windows PowerShell:

* minikube -p minikube docker-env --shell powershell | Invoke-Expression
* cd capstone/app
* docker build -t fastapi-nginx-demo:latest . 

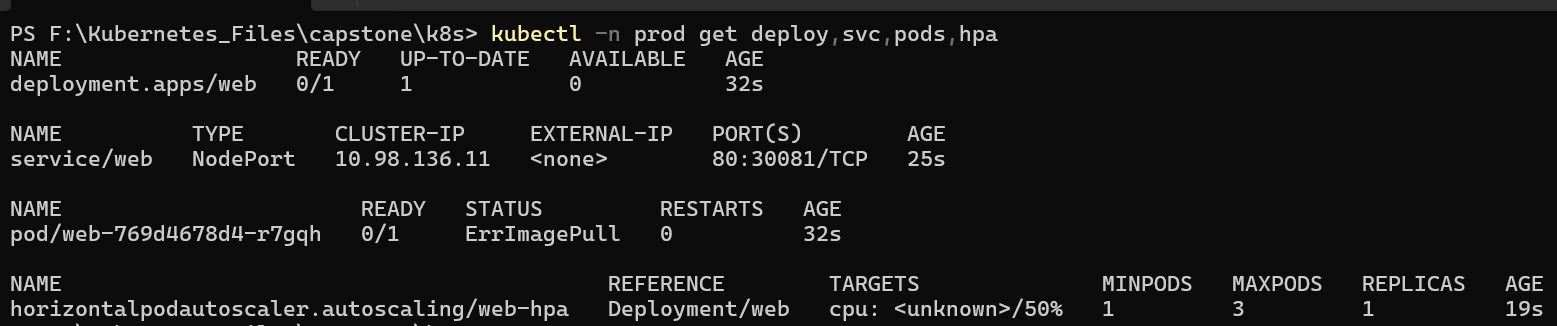
1. Apply Kubernetes resources:

* cd ../k8s
* kubectl apply -f namespace.yaml
* kubectl apply -n prod -f pvc-logs.yaml
* kubectl apply -n prod -f nginx-configmap.yaml
* kubectl apply -n prod -f deployment-web.yaml
* kubectl apply -n prod -f service-web-nodeport.yaml
* kubectl apply -n prod -f hpa-web.yaml

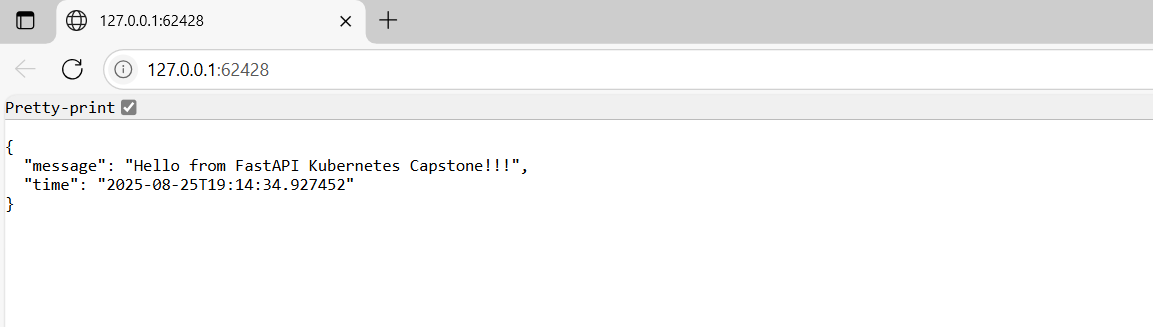


1. Verify:

* kubectl -n prod get deploy,svc,pods,hpa



* minikube service -n prod web –url



1. Test endpoints :

