Kubernetes Intro Exercises

Deployments, ConfigMaps, and Services

Setup

- Register at https://killercoda.com
- Open the Kubernetes playground: https://killercoda.com/playgrounds/scenario/kubernetes
- Check whether `kubectl` can talk to the cluster:

```
kubectl get nodes
```

- Apply the exercise resources:

```
kubectl apply -f https://georg-schwarz.com/2026-ohm-lectures/ex2.yaml
```

Exercise 1: Pods With and Without Controllers

- a) Which pod in `animals` is a standalone pod, and which pods are managed by a Deployment?
- b) Delete one pod with the word `tiger` in its name in namespace `animals`. What happens after 30 seconds?
- c) Delete one pod with the word `lion` in its name in namespace `animals`. What happens after 30 seconds?
- d) Explain the differences you observed in b and c.

```
# Useful commands
kubectl get pods
kubectl get replicasets
kubectl get deployments
kubectl describe pod <pod>
kubectl delete pod <pod>
```

Exercise 2: Deployments, ReplicaSets, and Labels

- a) Inspect the `cats` Deployment in namespace `animals`. How many replicas should run?
- b) Which ReplicaSet belongs to the Deployment?
- c) Which labels do the `cat` pods have?
- d) Scale the `cats` Deployment to 3 replicas and verify the result.
- e) There is a Service named `cats` in namespace `animals`. Why does it not route to any pods?
- f) Fix the service investigated in e).

```
# Useful commands
kubectl get deployments
kubectl get deployment <deployment>
kubectl describe deployment <deployment>
kubectl get pods --show-labels
kubectl scale deployment <deployment> --replicas=<replicas>
kubectl get service <service> -o yaml
kubectl get endpoints
kubectl get endpoints <endpoint>
kubectl edit service <service>
```

Exercise 3: ConfigMap as Environment Configuration

- a) Inspect the ConfigMap `dog-config` in namespace `animals`. Which values does it contain?
- b) Inspect the `dogs` Deployment. How is the ConfigMap connected to the container?
- c) Read the environment variables from a running `dog` pod.
- d) Call the `dogs` Service from a temporary debug pod and check which message is returned.

```
# Useful commands
kubectl get configmaps
kubectl describe configmap <configmap>
kubectl describe deployment <deployment>
kubectl get pods -l <label>
kubectl exec deploy/<deployment> -- env
kubectl run <podname> --image=curlimages/curl:8.10.1 -it --rm -- sh
```

Exercise 4: ConfigMap Changes

- a) Call the `repeater` service of namespace `iot` from a temporary debug pod.
- b) Change the value `MESSAGE` in ConfigMap `repeater` of namespace `iot` to `hello from changed config`.
- c) Call `repeater` again. Did the response change?
- d) Restart the Deployment rollout and wait until it finishes.
- e) Call `repeater` again. What changed?

```
# Useful commands
kubectl edit configmap <configmap>
kubectl rollout restart deployment/<deployment>
kubectl rollout status deployment/<deployment>
kubectl get pods -l <label>
kubectl run <podname> --image=curlimages/curl:8.10.1 -it --rm -- sh
```

Exercise 5: ClusterIP, Pod IP, and DNS

- a) Start a temporary debug pod in namespace `animals`.
- b) Find the Pod IPs behind the `mouse` Deployment and call one pod directly.
- c) Find the ClusterIP of the `mouse` Service and call it.
- d) Call the same Service through DNS names: `mouse`, `mouse.animals`, and `mouse.animals.svc.cluster.local`.
- e) Start the debug pod now in the `default` namespace and try again. What can you observe?

```
# Useful commands
kubectl get pods -l <label> -o wide
kubectl get services
kubectl describe service <service>
kubectl get service <service> -o yaml
kubectl run <podname> --image=curlimages/curl:8.10.1 -it --rm -- sh
```

Exercise 6: Create a ClusterIP Service

- a) The `recommendations` Deployment in `iot` has no Service. Confirm that it is not reachable through DNS yet.
- b) Create a ClusterIP Service named `recommendations` for the Deployment.
- c) Verify that the Service has endpoints.
- d) Call the Service from a debug pod by DNS name.

```
# Useful commands
kubectl get deployment <deployment>
kubectl get service <service>
kubectl expose deployment <deployment> --port=<service-port> --target-port=<deployment-port>
kubectl get endpoints
kubectl run <podname> --image=curlimages/curl:8.10.1 -it --rm -- sh
```

Exercise 7: Change ClusterIP to NodePort

- a) Change the `recommendations` Service from `ClusterIP` to `NodePort`.
- b) Find the assigned node port.
- c) Find a node IP address.

- d) Call the Service from the control plane using `<node-ip>:<node-port>`.

```
# Useful commands
kubectl get services
kubectl edit service <service>
kubectl get nodes -o wide
curl http://<node-ip>:<node-port>
```

Exercise 8: Rolling Update Visualization

- a) Inspect the current image and command of the `rollout-demo` Deployment in `letsroll`.
- b) Start the watch script in one terminal.
- c) In a second terminal, update the Deployment from version `v1` to `v2`.
- d) Watch how responses change during the rolling update.
- e) Inspect rollout history and ReplicaSets after the rollout.
- f) Roll back to the previous version.

Terminal 1:

```
curl -LO https://georg-schwarz.com/2026-ohm-lectures/ex2-rollout-watch.sh
chmod +x ex2-rollout-watch.sh

# Run watch script
./ex2-rollout-watch.sh
```

Terminal 2:

```
# Useful when a rollout changes the container image:
kubectl set image deployment/<deployment> <container>=<image>:<tag> -n <namespace>

kubectl edit deployment <deployment>
kubectl rollout status deployment/<deployment>
kubectl rollout history deployment/<deployment>
```