Exercise 02 - Robert Zacchia

Installation

Installation is pretty straight forward in Ubuntu. I used the snap package for kubectl, and downloaded the binary for minikube.

```
sudo snap install kubectl --classic

curl -Lo minikube
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64
chmod +x minikube
sudo cp minikube /usr/local/bin && rm minikube
```

After installing both programs I verified the installations via checking their versions.

```
minikube version

minikube version: v1.37.0
commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3


kubectl version

Client Version: v1.34.1
Kustomize Version: v5.7.1
Unable to connect to the server: dial tcp 192.168.49.2:8443: connect: no
route to
```

Configuration

The Configuration can be split into 3 parts

1. Creating the images for the application
2. Setting up deployment scripts
3. Applying them to the cluster

1. Images for the application

I decided to keep the application BookRent which I used in exercise 01. BookRent consists of 5 Services and 4 Databases.

Service Name	Database Name	External Callable
catalog-service	catalog-db	False

Service Name	Database Name	External Callable
identity-service	identity-db	False
renting-service	renting-db	False
user-service	user-db	False
orchestrator-service	None	True

Only the orchestrator-service is able to call the other services and be called from external sources. Since I already had Dockerfiles to install the images in minikube do the folowing

```
eval $(minikube -p bookrent docker-env)

docker build -t bookrent_catalog-service:latest \
  -f Services/BookRent.Catalog/Dockerfile .

docker build -t bookrent_orchestrator-service:latest \
  -f BookRent.Orchestrator/Dockerfile .

docker build -t bookrent_identity-service:latest \
  -f Services/BookRent.Identity/Dockerfile .

docker build -t bookrent_renting-service:latest \
  -f Services/BookRent.Renting/Dockerfile .

docker build -t bookrent_user-service:latest \
  -f Services/BookRent.User/Dockerfile .
```

2. Setting up deployment

2.1 Resource setup

Before I can setup the deployment I first needed to setup an environment and checked the status.

```
minikube start --profile=bookrent --driver=docker --cpus=4 --memory=12g

😄  [bookrent] minikube v1.37.0 on Ubuntu 24.04
✨  Using the docker driver based on user configuration
...
...
🏄  Done! kubectl is now configured to use "bookrent" cluster and "default"
namespace by default


minikube status --profile=bookrent
bookrent
type: Control Plane
```

```
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

After the minikube cluster was created I could set it up to be used by kubectl via

```
kubectl config use-context bookrent
Switched to context "bookrent".
kubectl config set-context --current --namespace=bookrent


kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
bookrent   Ready    control-plane   2m12s   v1.34.0
```

2.2 Setup Databases

Before I setup the database, I made an sql-secrety.yaml file to store the password for the databases. In real applications this file should not be in the source control but I included it for that example. I also use the same credentials for all databases.

```
apiVersion: v1
kind: Secret
metadata:
  name: mssql-secret
type: Opaque
stringData:
  SA_PASSWORD: "Your_SA_Password_123!"
```

This helps me to set the password in the databases.

The databases consist of multiple statefulsets which are defined in the sql.yaml file.

```
# ========================
# CATALOG DB (StatefulSet)
# ========================
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: catalog-db
  namespace: bookrent
  labels:
    app: catalog-db
spec:
  serviceName: catalog-db
  replicas: 1
```

```
  ...
  spec:
    containers:
      - name: mssql
        image: mcr.microsoft.com/mssql/server:2022-latest
        ports:
          - containerPort: 1433
        env:
         ...
          - name: SA_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mssql-secret
                key: SA_PASSWORD
        resources:
          ...
        volumeMounts:
          - name: mssqldata
            mountPath: /var/opt/mssql
    volumeClaimTemplates:
      -...
```

Furthermore I add services to each database to have persistent communication between the different parts of the distributed application.

```
apiVersion: v1
kind: Service
metadata:
  name: catalog-db
  namespace: bookrent
spec:
  selector:
    app: catalog-db
  ports:
    - port: 1433
      targetPort: 1433
```

2.3 Setup APIs

The Apis are setup similar except that they are Deployments and not StatefulSets.

2.4 Create SSL and Ingress and Network policies

To be able to use the application via https we have to create an SSL certificate at first. Please note that self signed certificates will cause warnings in most modern browsers.

```
openssl req -x509 -nodes -days 365 \
  -newkey rsa:2048 \
  -keyout api-bookrent.key \
  -out api-bookrent.crt \
  -subj "/CN=api.bookrent.local/O=api.bookrent.local"

kubectl create secret tls bookrent-tls \
  --cert=api-bookrent.crt \
  --key=api-bookrent.key \
  -n bookrent
```

The ingress can be found in orchestrator_ingress.yaml The ingress setup ensures, that only that named service can be called from outside.

The networkpolicy ensures, that every catalog-db can only be called from a catalog-service.

3. Startup After Setup

After the Cluster files are generated they need to be executed in the following order.

```
# 1) secret
kubectl apply -f k8s/sql-secrety.yaml

# 2) DBs (persistent)
kubectl apply -f k8s/sql.yaml

# 3) APIs / services
kubectl apply -f k8s/services.yaml

# 4) NetworkPolicies
kubectl apply -f k8s/network-policy.yaml

# 5) Ingress (prod + dev)
kubectl apply -f k8s/orchestrator_ingress.yaml
```

To get the ip adress of the ingress and see if the cluster runs

```
minikube addons enable ingress --profile=bookrent
minikube ip --profile=bookrent
kubectl get ingress -n bookrent
```

Issues

Currenty I have still issues with running the application and sadly not enough time to fix it before the deadline.

```

```

```
kubectl get pods -n bookrent

NAME                                READY   STATUS             RESTARTS
AGE
catalog-db-0                        1/1     Running            1 (7h4m
ago)   7h22m
catalog-service-7f7f75db74-gpr8k    0/1     ErrImagePull       0
12s
catalog-service-89b458f9-j8xcf      0/1     ImagePullBackOff   0
7h22m
identity-db-0                       1/1     Running            1 (7h4m
ago)   7h22m
identity-service-56cbbb6cfc-gps79   0/1     ImagePullBackOff   0
7h22m
orchestrator-service-689d977cbf-jcgsr   0/1     ImagePullBackOff   0
7h22m
renting-db-0                        1/1     Running            1 (7h4m
ago)   7h22m
renting-service-5694497899-6p7xz    0/1     ImagePullBackOff   0
7h22m
user-db-0                           1/1     Running            1 (7h4m
ago)   7h22m
user-service-7dd646bc89-8lvhk       0/1     ImagePullBackOff   0
7h22m


 kubectl describe pod catalog-service-7f7f75db74-gpr8k -n bookrent
Name:             catalog-service-7f7f75db74-gpr8k
Namespace:        bookrent
Priority:         0
Service Account:  default
Node:             bookrent/192.168.49.2
Start Time:       Wed, 05 Nov 2025 06:44:16 +0100
Labels:           app=catalog-service
...
...
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  94s                default-scheduler  Successfully
assigned bookrent/catalog-service-7f7f75db74-gpr8k to bookrent
  Normal   BackOff    20s (x4 over 92s)  kubelet            Back-off
pulling image "bookrent_catalog-service:latest"
  Warning  Failed     20s (x4 over 92s)  kubelet            Error:
ImagePullBackOff
  Normal   Pulling    5s (x4 over 94s)   kubelet            Pulling image
"bookrent_catalog-service:latest"
  Warning  Failed     4s (x4 over 92s)   kubelet            Failed to pull
image "bookrent_catalog-service:latest": Error response from daemon: pull
access denied for bookrent_catalog-service, repository does not exist or
may require 'docker login': denied: requested access to the resource is
denied
```

```
  Warning  Failed     4s (x4 over 92s)   kubelet            Error:
ErrImagePull
```