



"Ss. Cyril and Methodius" University in Skopje
**FACULTY OF COMPUTER
SCIENCE AND ENGINEERING**

MOVIE WATCHLIST APP

Завршен проект по предметот Континуирана интеграција и испорака

Docker, Kubernetes, Github Actions & AWS

Ментор:

д-р Панче Рибарски

м-р Стефан Андонов

Изработила:

Бојана Андонова 221225

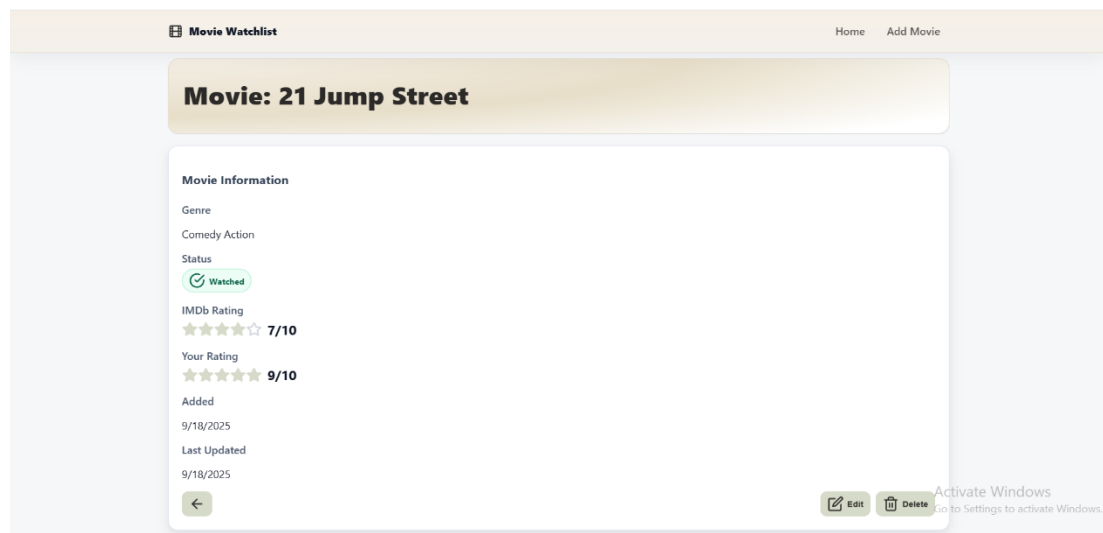
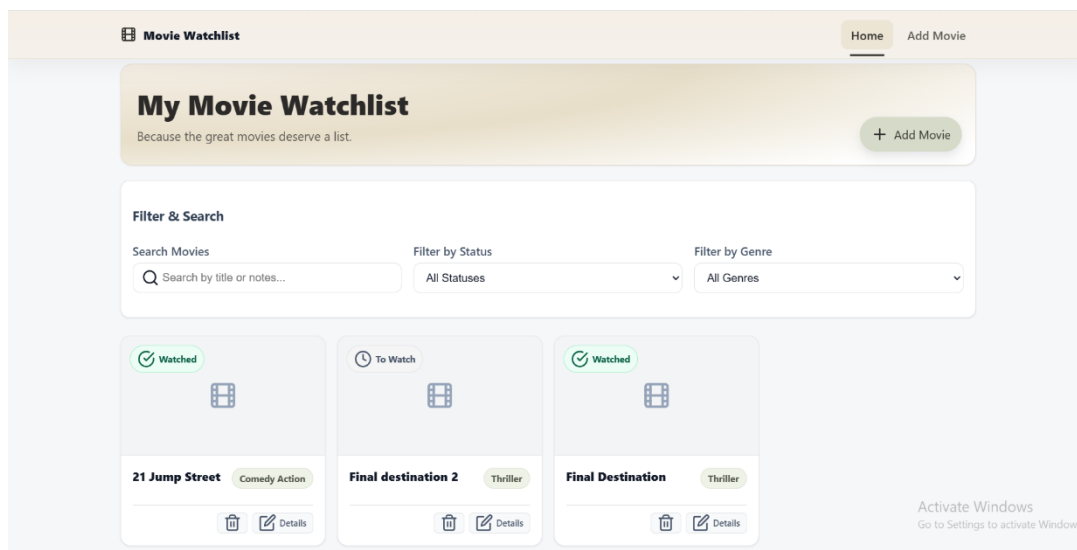
Содржина

Вовед	3
1. Github Репозиториум	4
2. Докеризација	4
3. Оркестрација со Docker Compose	5
4. Github Actions и автоматски Deploy	6
5. Kubernetes	9

Вовед

Movie Watchlist App е веб-апликација за управување со листа на филмови што сакате да ги гледате или веќе сте ги гледале. Апликацијата е развиена како финален проект за предметот Континуирана интеграција и испорака. Овозможува да додавате филмови во вашата колекција, да ги организирате според статус (за гледање/гледано), да давате лични рејтинзи и белешки, да пребарувате и филтрирате според жанр или наслов, и да проследувате статистики за вашата колекција. Апликацијата е изработена со React/TypeScript за frontend, FastAPI/Python за backend, MongoDB за база на податоци.

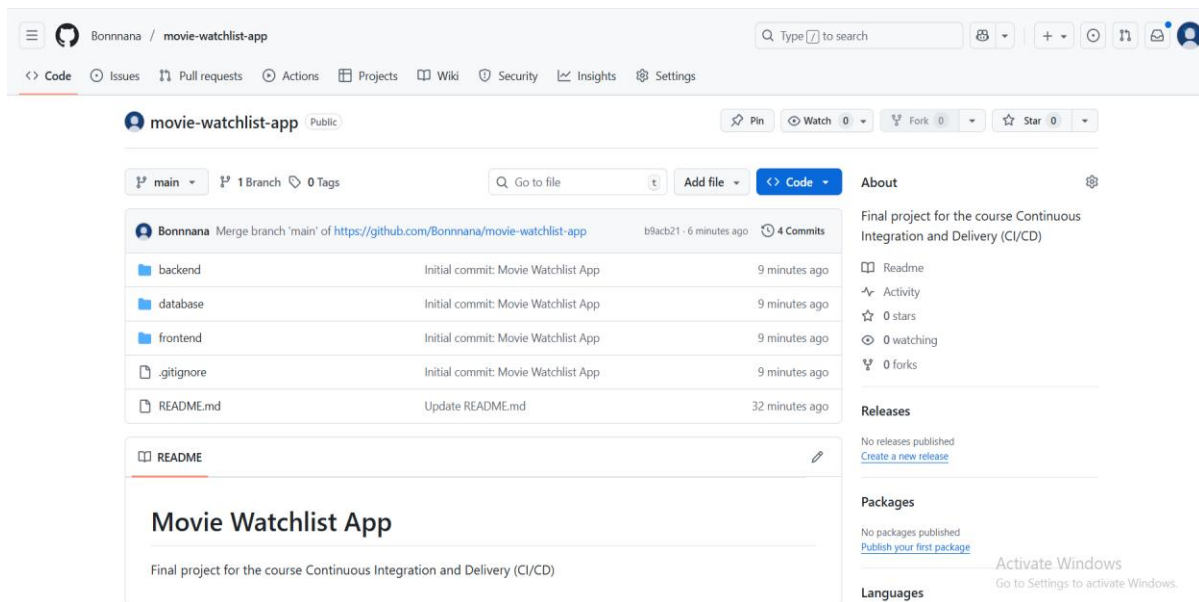
За потребите на овој предмет, соодветно се извршени сите барања кои подетално се опишани подолу.



1. Github Репозиториум

Апликацијата е поставена на јавен Github репозиториум достапен на следниот линк:

<https://github.com/Bonnnana/movie-watchlist-app.git>



2. Докеризација

Во рамките на гит репозиториумот, поставени се два Dockerfile фајлови, еден за frontend и еден за backend. За базата ја користам официјалната MongoDB Docker слика од Docker Hub.

```
1 FROM node:20-alpine as build
2
3 WORKDIR /app
4
5 COPY package*.json ./
6
7 RUN npm ci
8
9 COPY . ./
10
11 RUN npm run build
12
13 FROM nginx:alpine
14
15 COPY --from=build /app/dist /usr/share/nginx/html
16
17 COPY nginx.conf /etc/nginx/nginx.conf
18
19 EXPOSE 80
20
21 CMD ["nginx", "-g", "daemon off;"]

1 FROM python:3.11-slim
2
3 WORKDIR /app
4
5 COPY requirements.txt .
6 RUN pip install --no-cache-dir -r requirements.txt
7
8 COPY . .
9
10 EXPOSE 5000
11
12 CMD ["python", "main.py"]
```

3. Оркестрација со Docker Compose

Во `docker-compose.yml` фајл-от за оркестрација на базата на податоци и апликацијата имам дефинирано три сервиси:

mongodb: користи последна верзија од MongoDB и ги сетира потребните `environment` променливи како и `volumes` за базата, односно `mongodb_data` за персистентност и `init-mongo.js` за иницијализација.

backend: користејќи го `Dockerfile`-от се `build`-нува апликацијата и сетира соодветна околина со `environment` променливи и порти (`5000:5000`). Дополнително се означува дека апликацијата `backend` зависи од `mongodb`, односно за да биде стартувана апликацијата, прво ни е потребен сервисот `mongodb` за базата на податоци

frontend: исто така користејќи го `Dockerfile`-от се `build`-нува апликацијата и сетира соодветна околина со `environment` променливи и порти (`3000:80`). Дополнително се означува дека апликацијата `frontend` зависи од `backend` сервисот.

Сите сервиси се поврзани преку дефинираната `movie-watchlist-network` мрежа за комуникација помеѓу контејнерите.

```
version: '3.8'

services:
  mongodb:
    image: mongo:latest
    container_name: movie-watchlist-mongodb
    ports:
      - "27017:27017"
    volumes:
      - mongodb_data:/data/db
    networks:
      - movie-watchlist-network

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: movie-watchlist-backend
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=${DATABASE_URL}
      - FRONTEND_URL=${FRONTEND_URL}
```

```

    depends_on:
      - mongodb
    networks:
      - movie-watchlist-network

frontend:
  build:
    context: ./frontend
    dockerfile: Dockerfile
  container_name: movie-watchlist-frontend
  ports:
    - "3000:80"
  environment:
    - VITE_API_URL=${VITE_API_URL}
  depends_on:
    - backend
  networks:
    - movie-watchlist-network

volumes:
  mongodb_data:

networks:
  movie-watchlist-network:
    driver: bridge

(.venv) PS C:\Users\Bojana\Desktop\Proekti\movie-watchlist-app> docker compose up -d
time="2025-09-16T23:37:48+02:00" level=warning msg="C:\\Users\\Bojana\\Desktop\\Proekti\\movie-watchlist-app\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2025-09-16T23:37:48+02:00" level=warning msg="C:\\Users\\Bojana\\Desktop\\Proekti\\movie-watchlist-app\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
"
✓ Container movie-watchlist-mongodb          Started      0.7s
✓ Container movie-watchlist-backend          Started      0.9s
✓ Container movie-watchlist-frontend          Started      1.0s
"

(.venv) PS C:\Users\Bojana\Desktop\Proekti\movie-watchlist-app> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
2e4906f86cbc   movie-watchlist-app-frontend        "/docker-entrypoint..." 11 minutes ago Up 11 minutes 0.0.0.0:3000->80/tcp                movie-watchlist-frontend
b72bb0d69fae   movie-watchlist-app-backend         "python main.py"         11 minutes ago Up 11 minutes 0.0.0.0:5000->5000/tcp              movie-watchlist-backend
a70ec2c84744   mongo:latest                        "docker-entrypoint.s..." 11 minutes ago Up 11 minutes 0.0.0.0:27017->27017/tcp            movie-watchlist-mongodb

```

4. Github Actions и автоматски Deploy

За делот на Continuous Integration користам Github Actions со цел да се автоматизира процесот за build и deploy на Docker Images. Со тоа, секогаш кога одредена промена е направена на Git репозиториумот, GitHub Actions го започнува **docker-build-and-push.yml** workflow-от (pipeline-от) за да се направи нова Docker Image и истата да се стави на DockerHub, така што секоја нова верзија на кодот автоматски се транспарентира во нов

Docker Image, подготвен за користење при локален развој, тестирање или деплојмент во Kubernetes.

По успешното завршување на build процесот, се активира `deploy.yml` workflow-от кој автоматски се поврзува на AWS EC2 инстанцата преку SSH и ја деплојнува апликацијата со pull на најновиот код од Git репозиториумот, рестартирање на контејнерите со `docker-compose`, и извршување на `health check` за да се потврди дека апликацијата работи правилно во продукциското окружување.

Дополнително, за сето ова да работи, во делот `Secrets` додадени се потребните кренденцијали и променливи. Ова овозможува истите да не се хардкодирани и јавно достапни.

Repository secrets			New repository secret
Name ↕	Last updated		
DATABASE_URL	8 hours ago		
DOCKERHUB_TOKEN	2 days ago		
DOCKERHUB_USERNAME	2 days ago		
EC2_HOST	2 hours ago		
EC2_SSH_KEY	2 hours ago		
MONGO_APP_PASSWORD	8 hours ago		
MONGO_APP_USERNAME	8 hours ago		
MONGO_INITDB_ROOT_PASSWORD	8 hours ago		
MONGO_INITDB_ROOT_USERNAME	8 hours ago		

Bonnana / movie-watchlist-app

<> Code Issues Pull requests Actions Projects Wiki Security Insights

Build and Push Docker images to Docker Hub

Update docker-build-and-push.yml #4

Summary

Jobs

docker

Run details Usage Workflow file

docker

succeeded now in 27s

Set up job

Checkout code

Login to Docker Hub

Set up Docker Buildx

Install Docker Compose

Build and push Docker images using docker-compose

Post Set up Docker Buildx

Post Login to Docker Hub

Post Checkout code

Bonnana / movie-watchlist-app

<> Code Issues Pull requests Actions Projects Wiki Security Insights

Deploy to EC2

Deploy to EC2 #9

Summary

Jobs

deploy

Run details Usage Workflow file

deploy

succeeded 1 hour ago in 55s

Set up job

Build appleboy/ssh-action@v1.0.3

Checkout code

Debug EC2 Connection

Deploy to EC2

Health Check

Post Checkout code

Complete job

hub Explore My Hub Search Docker Hub CtrlK

bojanaandonova Docker Personal

Repositories

All repositories within the bojanaandonova namespace.

Search by repository name All content Create a repository

Name	Last Pushed	Contains	Visibility	Scout
bojanaandonova/movie-watchlist-frontend	less than a minute ago	IMAGE	Public	Inactive
bojanaandonova/movie-watchlist-backend	less than a minute ago	IMAGE	Public	Inactive
bojanaandonova/static-site	4 months ago	IMAGE	Public	Inactive
bojanaandonova/my-nginx	4 months ago	IMAGE	Public	Inactive

aws Search [Alt+S] United States (N. Virginia) Account ID: 1583-9434-9097 Bojana Andonova

EC2 > Instances

EC2 Dashboard EC2 Global View Events

Instances

Instances (1) Info

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Movie Watchlist App	i-0bf42c711c4959193	Running	t3.micro	3/3 checks passed	View alarms	us-east-1b

Not secure 54.90.233.171:3000

Movie Watchlist Home Add Movie

My Movie Watchlist

Because the great movies deserve a list.

+ Add Movie

Filter & Search

Search Movies Search by title or notes...

Filter by Status All Statuses

Filter by Genre All Genres

Апликацијата сега е јавно достапна на <http://54.90.233.171:3000/>

5. Kubernetes

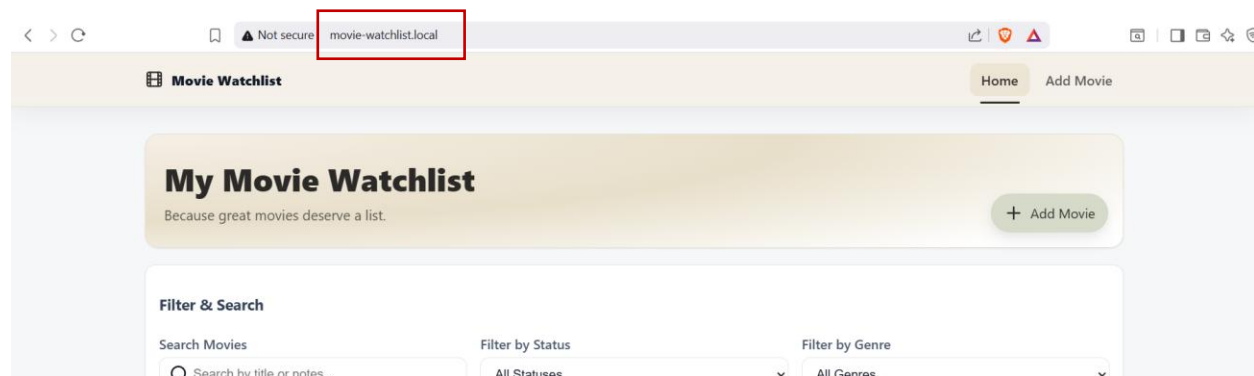
За овој дел најпрво креирам посебен namespace movie-watchlist со namespace.yml file. За базата на податоци дефинирам secrets.yml file каде што се дефинира secret за лозинката на MongoDB базата на податоци, како и Stateful Set што овозможува стабилна мрежа и перзистирање на податоците што ги имаме за апликацијата, менаџира со подови од image од самиот MongoDB и ни овозможува scaling на подовите. Перзистирањето на податоците е овозможено со Persistent Volume Claim и креирање на Persistent Volume од тој claim со што податоците се чуваат и после уништување на подот и се достапни после креирање на нов под. Понатаму, за backend апликацијата дефинирам backend/deployment.yml што содржи Deployment и backend/service.yml што содржи Service за FastAPI апликацијата. Преку

```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: movie-watchlist
5    labels:
6      name: movie-watchlist
```

```
spec:
  ingressClassName: nginx
  rules:
    - host: movie-watchlist.local
      http:
        paths:
          - path: /api
            pathType: Prefix
            backend:
              service:
                name: movie-watchlist-backend
                port:
                  number: 5000
          - path: /
            pathType: Prefix
            backend:
              service:
                name: movie-watchlist-frontend
                port:
                  number: 80
```

Deployment овозможуваме креирање и менаџирање на повеќе исти Pods со што се обезбедува Scaling, Self-Healing, Rollouts и Rollbacks, а преку Service овозможуваме пристап до самите подови. Слично, за frontend делот дефинирам frontend/deployment.yml и frontend/service.yml за React апликацијата. Последно, креирам nginx-proxy Ingress што овозможува пристап до нашата апликација од надвор и го користи претходно дефинираните сервиси за frontend и backend деловите од апликацијата. Ова овозможува пристап до апликацијата преку адресата:

<http://movie-watchlist.local/>



Во configmap.yml file се дефинирани конфигурациони податоци за апликацијата како што се database URL, API endpoints и други environment variables кои се потребни за правилно функционирање на апликацијата

Дополнително, креирам и kustomization.yml file кој служи како централна точка за управување со сите Kubernetes ресурси во апликацијата. Овој фајл овозможува лесно деплојување на целата апликација со една команда `kubectl apply -k .` наместо да се применуваат сите YAML фајлови посебно

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

namespace: movie-watchlist

resources:
- namespace.yaml
- configmap.yaml
- mongodb/secret.yaml
- mongodb/configmap.yaml
- mongodb/statefulset.yaml
- mongodb/service.yaml
- backend/deployment.yaml
- backend/service.yaml
- frontend/deployment.yaml
- frontend/service.yaml
- ingress/ingress.yaml

commonLabels:
  app.kubernetes.io/name: movie-watchlist
  app.kubernetes.io/version: "1.0.0"
```