

Факултет за информатички науки
и компјутерско инженерство

Проектна задача

Предмет: Континуирана интеграција и испорака

Наслов: Winery App

Професор:

Милош Јовановиќ

Панче Рибарски

Изработил:

Елена Стаменковска, 211180

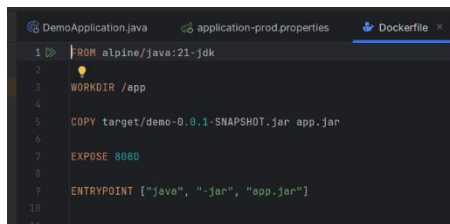
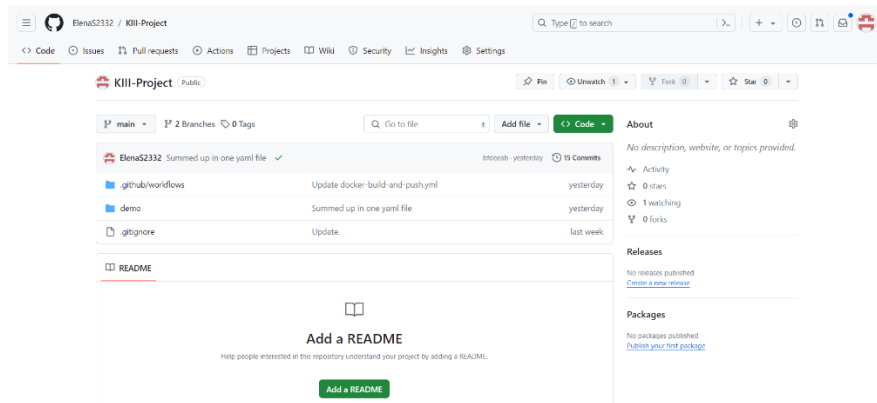
Датум и место:

25.6.2024, Скопје

Овој проект е изработен со Spring Framework, користејќи PostgreSQL база на податоци. Имено, проектот е изработен во група за потребите на предметот Дизајн и архитектура на софтвер и претставува едноставна апликација за листање на винарии со регистрација на корисници заради која се користи базата.

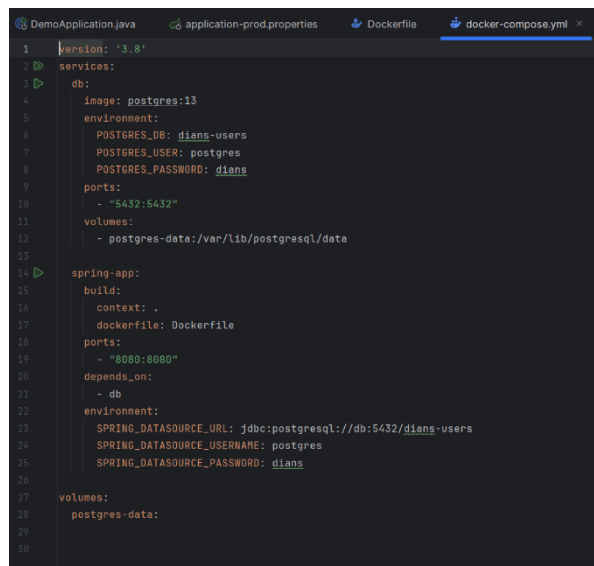
ДЕЛ 1: Docker, Docker Compose, GitHub, GitHub Actions

Најпрвин креирав нов репозиториум на GitHub на кој се наоѓа мојата апликација.



Понатаму креирав Dockerfile во кој се зема готов image за јава 21 од docker registry (DockerHub). Се копира содржината на изградената апликација на место кај што docker може да пристапи. После успешен docker build нашата апликација е контејнеризирана.

Со оглед на тоа што апликацијата користи база на податоци, сакаме таа база да биде во посебен контејнер. Поради тоа, правам docker-compose.yml file којшто прави посебни контејнери за базата и за главната апликација и ги стартнува (оркестрира) и двата заедно. По успешен build апликацијата е успешно стартната и можеме да ја пристапиме на localhost:8080.



© 2004 by Blackwell Publishing Ltd

Со помош на Docker Desktop можеме да ја следиме состојбата на нашите контејнери.

to

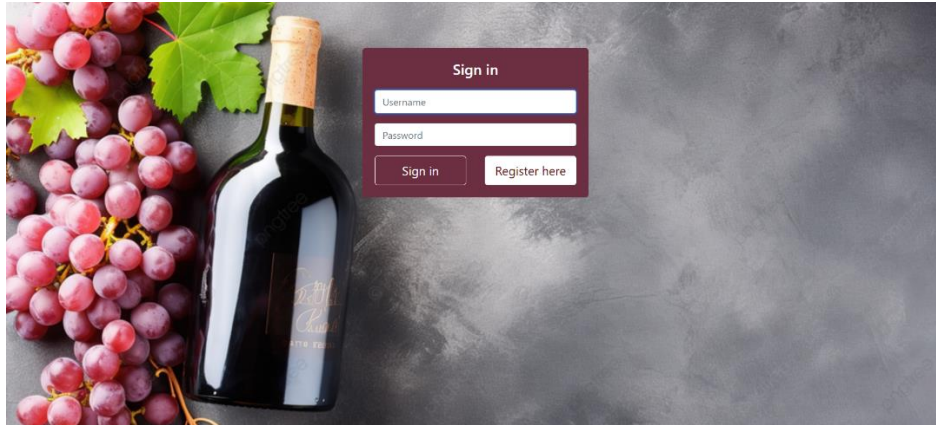
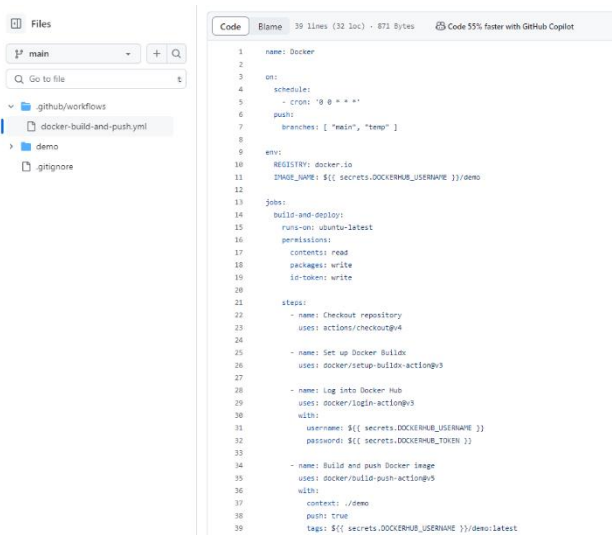
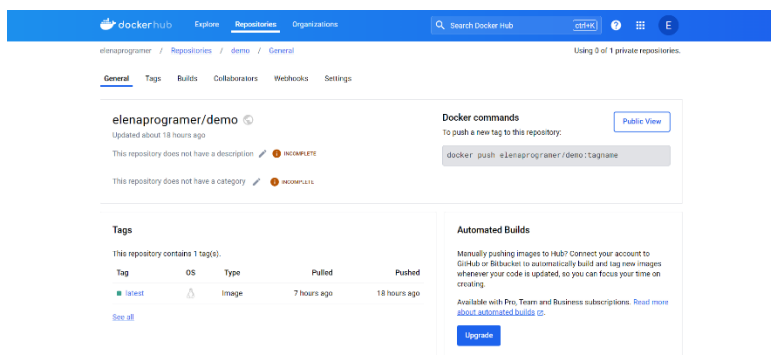
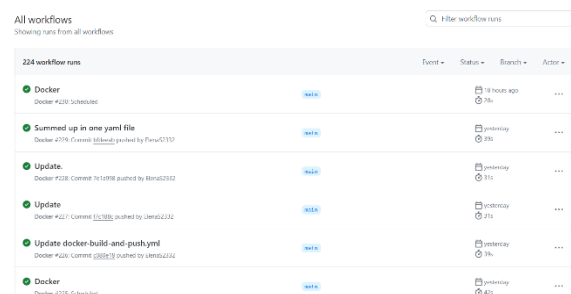
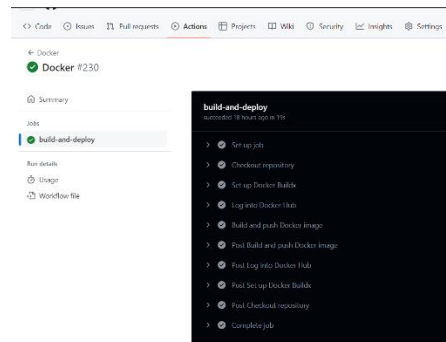
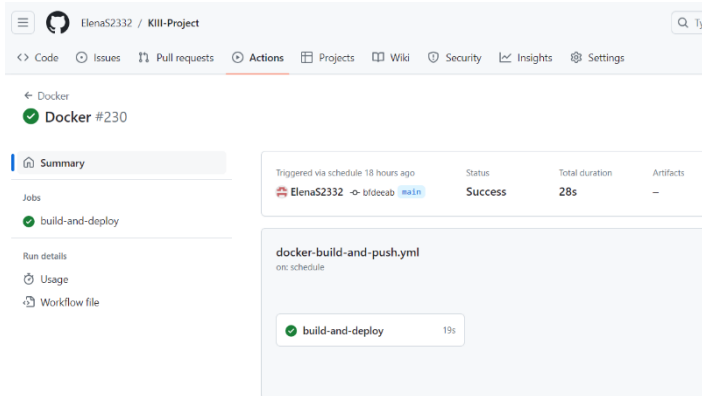


Image-от од новата апликација го ставам во нов репозиториум на Dockerhub. На ваков начин апликацијата е достапна за останатите луѓе да пристапат до неа со pull на мојот image.



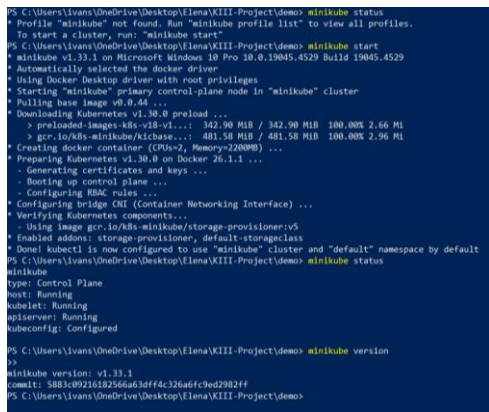
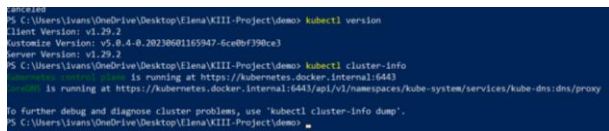
За CI/CD ја одбрав околината GitHub Actions, каде што модифицирав веќепостоечки docker build and push workflow, со тоа што мојот workflow ќе се извршува еднаш секој ден и на секој push на гранките main и temp.





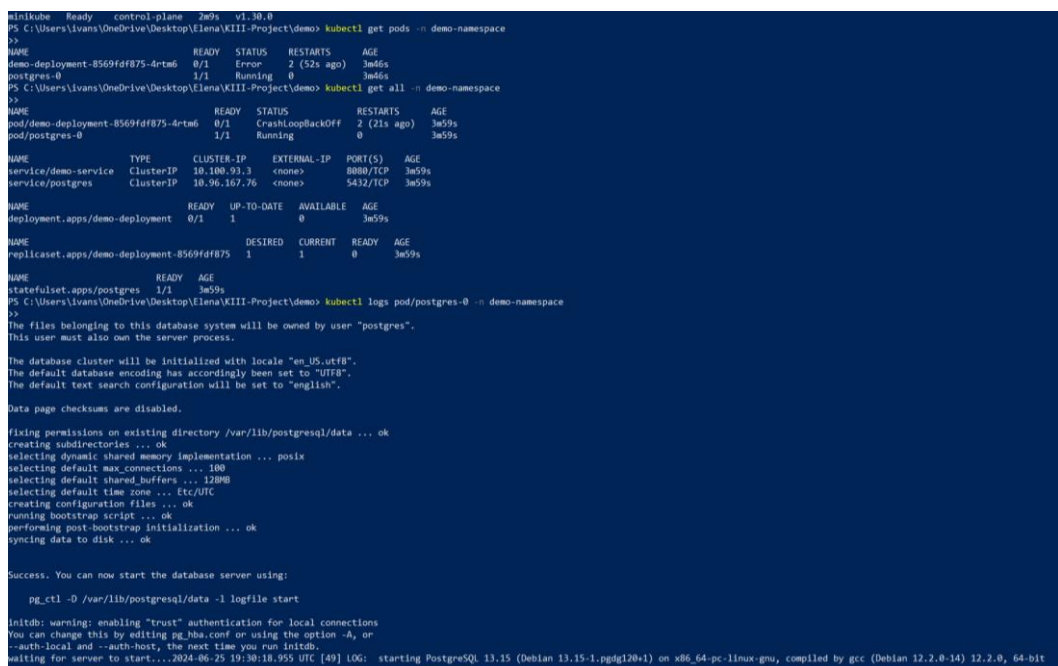
ДЕЛ 2: Kubernetes

За овој дел покрај овозможувањето на kubectl, инсталирав и minikube кој ми се чинеше најсоодветен за Windows. Креирањето на нов кластер со minikube е едноставно и се врши со командата: 'minikube start'.



Најпрвин креирам посебни yaml files за секој манифест, но за поедноставно повикување на командата 'kubectl apply' ги споив во еден yml file demo-deployment.yaml.

Како што може да се забележи креирам свој патесрасе на кој го прикачувам секој манифест. Базата на податоци работи во ред.



```
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

    pg_ctl -D /var/lib/postgresql/data -l logfile start

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
waiting for server to start...2024-06-25 19:30:18.955 UTC [49] LOG: starting PostgreSQL 13.15 (Debian 13.15-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2024-06-25 19:30:18.958 UTC [49] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-06-25 19:30:18.971 UTC [50] LOG: database system was shut down at 2024-06-25 19:30:17 UTC
2024-06-25 19:30:18.979 UTC [49] LOG: database system is ready to accept connections
done
server started
CREATE DATABASE

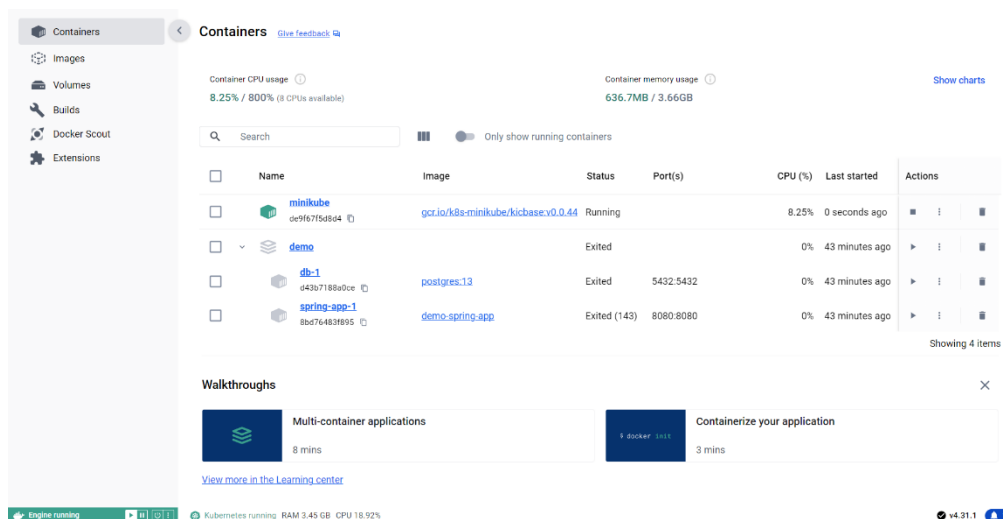
/usr/local/bin/docker-entrypoint.sh: Ignoring /docker-entrypoint-initdb.d/*

waiting for server to shut down...2024-06-25 19:30:19.750 UTC [49] LOG: received fast shutdown request
2024-06-25 19:30:19.754 UTC [49] LOG: aborting any active transactions
2024-06-25 19:30:19.757 UTC [49] LOG: background worker "logical replication launcher" (PID 56) exited with exit code 1
2024-06-25 19:30:19.761 UTC [51] LOG: shutting down
2024-06-25 19:30:19.787 UTC [49] LOG: database system is shut down
done
server stopped

PostgreSQL init process complete; ready for start up.

2024-06-25 19:30:20.076 UTC [1] LOG: starting PostgreSQL 13.15 (Debian 13.15-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2024-06-25 19:30:20.077 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
2024-06-25 19:30:20.077 UTC [1] LOG: listening on IPv6 address ":::", port 5432
2024-06-25 19:30:20.089 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-06-25 19:30:20.158 UTC [64] LOG: database system was shut down at 2024-06-25 19:30:19 UTC
2024-06-25 19:30:20.185 UTC [1] LOG: database system is ready to accept connections
PS C:\Users\livans\OneDrive\Desktop\Elena\K8S-Project\demo>
```

На Docker Desktop може да се забележи креацијата на Kubernetes контејнер преку minikube.



За потребите на проектот направив ConfigMap во кој ги чувам хостот и името на базата на податоци. Понатаму Secret во кој како base64 encoded ги чувам username и password на базата. Имам манифест за конфигурирање на Ingress, како и два сервиси од кои едниот е за самата апликација, а другиот е за базата. Базата ја чувам како StatefulSet и има свој Persistent Volumes Claim. Во овој StatefulSet ги користам key-value паровите од Secret и од ConfigMap. Единствен Deployment манифест е за самата главна апликација, кој исто така ги користи Secret и ConfigMap.

Изглед на манифестите:

```

apiVersion: v1
kind: Namespace
metadata:
  name: demo-namespace
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-conf
  namespace: demo-namespace
data:
  host: "localhost"
  name: "diana-users"
---
apiVersion: v1
kind: Secret
metadata:
  name: postgres-credentials
  namespace: demo-namespace
type: Opaque
data:
  postgres_user: c09zd0dyZXM=
  postgres_password: ZGlibnM=
---
apiVersion: v1
kind: Service
metadata:
  name: demo-service
  namespace: demo-namespace
spec:
  selector:
    app: demo
  ports:
    - name: http
      port: 80
      targetPort: 8080
  type: ClusterIP
---
postgres.user: c09zd0dyZXM=
postgres.password: ZGlibnM=
---
apiVersion: v1
kind: Ingress
metadata:
  name: demo-ingress
  namespace: demo-namespace
annotations:
  ingress.kubernetes.io/ssl-redirect: "true"
  traefik.ingress.kubernetes.io/ssl-passthrough: "false"
  traefik.ingress.kubernetes.io/rule-type: "PathPrefixStrip"
spec:
  ingressClassName: traefik
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            service:
              name: demo-service
              port:
                number: 80
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pv-claim
  namespace: demo-namespace
labels:
  app: postgres
  tier: database
spec:
  selector:
    app: postgres
    tier: database
  ports:
    - port: 5432
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
  namespace: demo-namespace
spec:
  selector:
    matchLabels:
      app: postgres
      tier: database
  template:
    metadata:
      labels:
        app: postgres
        tier: database
    spec:
      serviceAccountName: postgres
      containers:
        - name: postgres
          image: postgres:13
          ports:
            - containerPort: 5432
          env:
            - name: POSTGRES_USER
              valueFrom:
                secretKeyRef:
                  name: postgres-credentials
                  key: postgres_user
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: postgres-credentials
                  key: postgres_password
            - name: POSTGRES_DB
              value: diana-users
          volumeMounts:
            - name: postgres-storage
              mountPath: /var/lib/postgresql/data
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: postgres-pv-claim

```

```
spec:
  - port: 5432
  ---
  apiVersion: apps/v1
  kind: StatefulSet
  metadata:
    name: postgres
    namespace: demo-namespace
  spec:
    serviceName: postgres
    replicas: 1
    selector:
      matchLabels:
        app: postgres
    template:
      metadata:
        labels:
          app: postgres
      spec:
        containers:
          - name: postgres
            image: postgres:13
            env:
              - name: POSTGRES_USER
                valueFrom:
                  secretKeyRef:
                    name: postgres-credentials
                    key: postgres_user
              - name: POSTGRES_PASSWORD
                valueFrom:
                  secretKeyRef:
                    name: postgres-credentials
                    key: postgres_password
            ports:
              - containerPort: 5432
            volumeMounts:
              - name: postgres-persistence-storage
                mountPath: /var/lib/postgresql/data
            volumes:
              - name: postgres-persistence-storage
                persistentVolumeClaim:
                  claimName: postgres-pv-claim
  ---
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: demo-deployment
    namespace: demo-namespace
  spec:
    replicas: 3
    selector:
      matchLabels:
        app: demo
    strategy:
      rollingUpdate:
        maxSurge: 5
        maxUnavailable: 1
    template:
      metadata:
        labels:
          app: demo
      spec:
        containers:
          - name: demo
            image: alenaprogramer/demo:latest
            ports:
              - containerPort: 8080
            env:
              - name: DB_HOST
                valueFrom:
                  configMapKeyRef:
                    name: postgres-conf
                    key: host
              - name: DB_NAME
                valueFrom:
                  configMapKeyRef:
                    name: postgres-conf
                    key: name
              - name: POSTGRES_USER
                secretKeyRef:
                    name: postgres-credentials
                    key: postgres_user
              - name: POSTGRES_PASSWORD
                secretKeyRef:
                    name: postgres-credentials
                    key: postgres_password
```

Линк до проектот на GitHub: <https://github.com/ElenaS2332/KIII-Project>