



МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Интеллектуальные системы информационной безопасности»

Администрирование баз данных

Практическая работа № 7-8

ОТЧЁТ

Выполнил студент группы
БСБО-07-20
Любовский С.В.

Москва 2023

Выполнение задания.

Было реализовано CRUD приложение на python используя фреймворк FastAPI. Было использовано две базы данных – Mongo и Postgresql.

Для Postgresql была настроена репликация, а для MongoDB шардирование и репликация для каждого шарда + репликация для MongoRouter.

Приложение позволяет управлять товарами, категориями товаров и пользователями, и ролями.

Для категорий реализована иерархическая структура, для пользователей реализован RBAC.

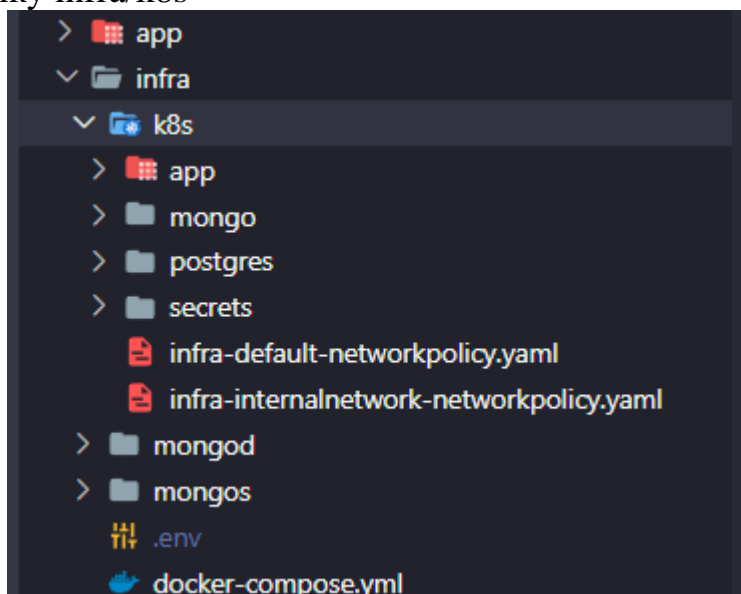
С кодом приложения и с файлами конфигурации можно ознакомиться здесь - https://github.com/wallseat/MIREA_database_administration_7-8

Для деплоя этого приложения можно воспользоваться двумя способами – docker-compose и Kubernetes. В рамках задания будем использовать Kubernetes.

Сначала развернем minikube кластер по инструкции с [официального сайта](#). В качестве драйвера был выбран Docker.

```
Removed all traces of the "minikube" cluster.
MIREA_database_administration_7-8 (main) ✓ minikube start
minikube v1.30.1 on Ubuntu 22.04 (amd64)
  MINIKUBE_ACTIVE_DOCKERD=minikube
  Automatically selected the docker driver. Other choices: kvm2, qemu2, ssh
  Using Docker driver with root privileges
  Starting control plane node minikube in cluster minikube
  Pulling base image ...
  minikube was unable to download gcr.io/k8s-minikube/kicbase:v0.0.39, but successfully downloaded docker.io/kicbase/stable:v0.0.39 as a fallback image
  Creating docker container (CPUs=2, Memory=3408MB) ...
  Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
    Generating certificates and keys ...
    Booting up control plane ...
    Configuring RBAC rules ...
  Configuring bridge CNI (Container Networking Interface) ...
    Using image gcr.io/k8s-minikube/storage-provisioner:v5
  Verifying Kubernetes components...
  Enabled addons: storage-provisioner, default-storageclass
  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
MIREA_database_administration_7-8 (main) ✓
```

Перейдем в папку infra/k8s



Тут мы видим yaml файлы описывающие наши сетки (infra-internalnetwork-networkpolicy, infra-default-networkpolicy), применим их.

```
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/infra-internalnetwork-networkpolicy.yaml
networkpolicy.networking.k8s.io/infra-internalnetwork created
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/infra-default-networkpolicy.yaml
networkpolicy.networking.k8s.io/infra-default created
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ _
```

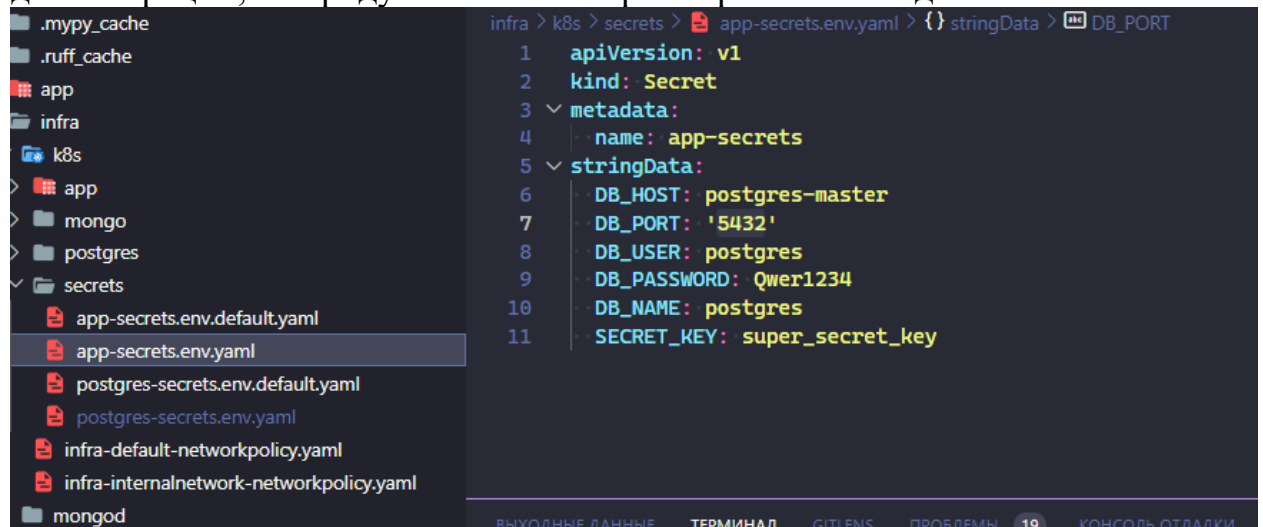
Проверяем

```
use kubectl options for a list of global command-line options (applies to all commands).
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl get networkpolicies
NAME                                POD-SELECTOR                                AGE
infra-default                       io.kompose.network/infra-default=true      55s
infra-internalnetwork               io.kompose.network/infra-internalnetwork=true 60s
```

сети успешно созданы.

Теперь зайдем в папку с секретами и создадим по шаблонам (app-secrets.env.default.yaml, postgres-secrets.env.default.yaml) два файла с таким же именем но без суффикса default

В данном примере, я не буду менять настройки, т.к. это тестовая среда для демонстрации, на продуктовом кластере секреты необходимо изменить!



```
infra > k8s > secrets > app-secrets.env.yaml > {} stringData > DB_PORT
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: app-secrets
5  stringData:
6    DB_HOST: postgres-master
7    DB_PORT: '5432'
8    DB_USER: postgres
9    DB_PASSWORD: Qwer1234
10   DB_NAME: postgres
11   SECRET_KEY: super_secret_key
```

Теперь так же применим их

```
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/secrets/postgres-secrets.env.yaml
secret/postgres-secrets created
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/secrets/app-secrets.env.yaml
secret/app-secrets created
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ _
```

Проверяем

```
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl get secrets
NAME            TYPE    DATA  AGE
app-secrets     Opaque  6      22s
postgres-secrets Opaque  5      28s
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓
```

секреты успешно созданы.

Теперь необходимо собрать наши докер образы (чтобы они были доступны нашему кластеру). Для того, чтобы использовать демон докера из кластера пропишем команду `eval $(minikube docker-env)`. Теперь мы подключены к демону внутри minikube и можно собирать наши образы.

Используем 3 команды, чтобы собрать образы приложения, mongod и mongos:

1. `docker build -t app:latest app`
2. `docker build -t mongos:latest infra/mongos`
3. `docker build -t mongod:latest infra/mongod`

Проверяем наличие образов в локальном регистре minikube

```
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mongod	latest	136b1e251356	4 seconds ago	693MB
mongos	latest	2737a44d959d	17 seconds ago	693MB
app	latest	eff584b97f76	About a minute ago	214MB
registry.k8s.io/kube-apiserver	v1.26.3	1d9b3cbae03c	2 months ago	134MB

Образы собраны, теперь можем применять наши сервисы и деплойменты.

Сначала поднимем кластер Mongo используя команду `ubectl apply -f infra/k8s/mongo`

```
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/mongo
deployment.apps/configdb-replica0 created
service/configdb-replica0 created
deployment.apps/configdb-replicat1 created
service/configdb-replicat1 created
deployment.apps/mongos-router created
service/mongos-router created
deployment.apps/shard0-replica0 created
service/shard0-replica0 created
deployment.apps/shard0-replicat1 created
service/shard0-replicat1 created
deployment.apps/shard1-replica0 created
service/shard1-replica0 created
deployment.apps/shard1-replicat1 created
service/shard1-replicat1 created
```

Проверяем поды

```
(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
configdb-replica0-6778d65d98-tw7wt	1/1	Running	0	28s
configdb-replicat1-7ffb8b594c-fq22t	1/1	Running	0	28s
mongos-router-f99dd6c7-ph9vj	1/1	Running	0	28s
shard0-replica0-5b7d7c5986-lsw8g	1/1	Running	0	28s
shard0-replicat1-5c558d78d7-qtzq8	1/1	Running	0	28s
shard1-replica0-589f58ddd-cvf9n	1/1	Running	0	28s
shard1-replicat1-5846588d95-qbdwg	1/1	Running	0	28s

кластер mongo успешно развернут.

Теперь развернем кластер Postgres используя команду `kubectl apply -f infra/k8s/postgres`

```

(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/postgres
deployment.apps/postgres-master created
service/postgres-master created
deployment.apps/postgres-slave created
service/postgres-slave created
persistentvolumeclaim/postgres-volume created

```

Проверяем поды

```

(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
configdb-replica0-6778d65d98-tw7wt  1/1     Running   0           3m17s
configdb-replica1-7ffb8b594c-fq22t  1/1     Running   0           3m17s
mongos-router-f99dd6c7-ph9vj        1/1     Running   0           3m17s
postgres-master-5dfb448866-5mz8j    1/1     Running   0           113s
postgres-slave-ff98f7fcd-nxz4t      1/1     Running   0           113s
shard0-replica0-5b7d7c5986-lsw8g    1/1     Running   0           3m17s
shard0-replica1-5c558d78d7-qtzq8    1/1     Running   0           3m17s
shard1-replica0-589f58ddd-cvf9n     1/1     Running   0           3m17s
shard1-replica1-5846588d95-qbdwg    1/1     Running   0           3m17s

```

кластер postgres успешно развернут.

Теперь развернем наше приложение используя команду `kubectl apply -f infra/k8s/app`

```

(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl apply -f infra/k8s/app
deployment.apps/app created
service/app created

```

Проверяем поды

```

(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
app-676dbf6c9-4nbx5                1/1     Running   0           49s
configdb-replica0-6778d65d98-tw7wt  1/1     Running   0           5m28s
configdb-replica1-7ffb8b594c-fq22t  1/1     Running   0           5m28s
mongos-router-f99dd6c7-ph9vj        1/1     Running   0           5m28s
postgres-master-5dfb448866-5mz8j    1/1     Running   0           4m4s
postgres-slave-ff98f7fcd-nxz4t      1/1     Running   0           4m4s
shard0-replica0-5b7d7c5986-lsw8g    1/1     Running   0           5m28s
shard0-replica1-5c558d78d7-qtzq8    1/1     Running   0           5m28s
shard1-replica0-589f58ddd-cvf9n     1/1     Running   0           5m28s
shard1-replica1-5846588d95-qbdwg    1/1     Running   0           5m28s

```

приложение успешно развернуто.

Теперь необходимо зайти на под с приложением и применить миграции. Используя команду `kubectl exec -it <ИМЯ_ПОДА> -- bash` подключимся к консоли внутри нашего контейнера.

```

(shopper-cms-py3.11) → MIREA_database_administration_7-8 (main) ✓ kubectl exec -it app-676dbf6c9-4nbx5 -- bash
root@app-676dbf6c9-4nbx5:/app#

```

Теперь применим миграции командой `alembic upgrade head`

```

root@app-676dbf6c9-4nbx5:/app# alembic upgrade head
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> 2801295852f4, initial migration
INFO [alembic.runtime.migration] Running upgrade 2801295852f4 -> 71cd8c7135b8, add product table
INFO [alembic.runtime.migration] Running upgrade 71cd8c7135b8 -> f245daf5f54f, add username
INFO [alembic.runtime.migration] Running upgrade f245daf5f54f -> ecf95befe18f, add unique
INFO [alembic.runtime.migration] Running upgrade ecf95befe18f -> 3a794e77645e, add time fields to category
INFO [alembic.runtime.migration] Running upgrade 3a794e77645e -> 6861747480bc, add role table, add user_role table
root@app-676dbf6c9-4nbx5:/app#

```

Теперь создадим пользователя администратора (создать его можно только используя скрипт, т.к. для создания пользователей из интерфейса необходимо иметь права, т.е. быть авторизованным). Для этого существует скрипт `create_superuser.py`.

```

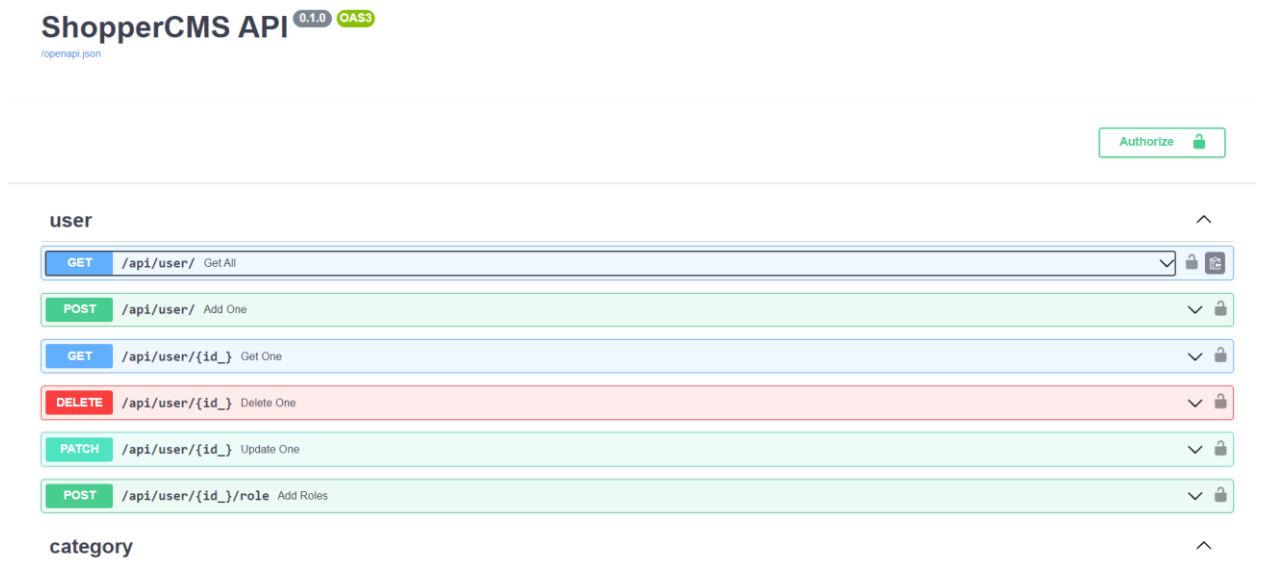
root@app-676dbf6c9-4nbx5:/app# python3 src/scripts/create_superuser.py
Введите username: admin
Введите email: admin@example.com
Введите пароль: Qwer1234
Повторите пароль: Qwer1234
root@app-676dbf6c9-4nbx5:/app#

```

Пользователь успешно создан. Теперь мы можем воспользоваться нашим приложением.

Чтобы получить доступ, необходимо прокинуть порт сервиса на наш локальный порт, сделать это можно через команду `kubectrl port-forward deployment/app 8080:8080`

Перейдем по адресу `localhost:8080/docs`, чтобы открыть интерфейс swagger



Наше приложение доступно!

Теперь попробуем залогиниться под админской учеткой которую мы создали ранее

auth

POST /api/auth/ Auth

Parameters

No parameters

Request body required application/json

```
{
  "email_or_username": "admin",
  "password": "Qwer1234"
}
```

Details

Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX21kIjo1MDE4ODg5N2NmNmM2M2JmZjZjFhYTUzOTkxMmM1LCJleHRpcwVzIjo4Njg1ODg2MjE0Ljg0TU3Mj39.85ox6R315kTWEfmSwf61_Dwh6b7s5-9KN_M1FKUDyL",
  "expires": 1685886214.8495722
}
```

Response headers

content-length: 222

Мы получили access token и теперь можем проводить авторизованные операции. Например, получить список всех пользователей.

Response body

```
[
  {
    "id": "0188817cf5c63bb4b22fcf1aa5398d1c",
    "username": "admin",
    "email": "admin@example.com",
    "created_at": "2023-06-03T13:38:51.219396",
    "updated_at": "2023-06-03T13:38:51.219490",
    "roles": [
      "Admin"
    ]
  }
]
```