|  |
| --- |
|  |
| 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.

# 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.

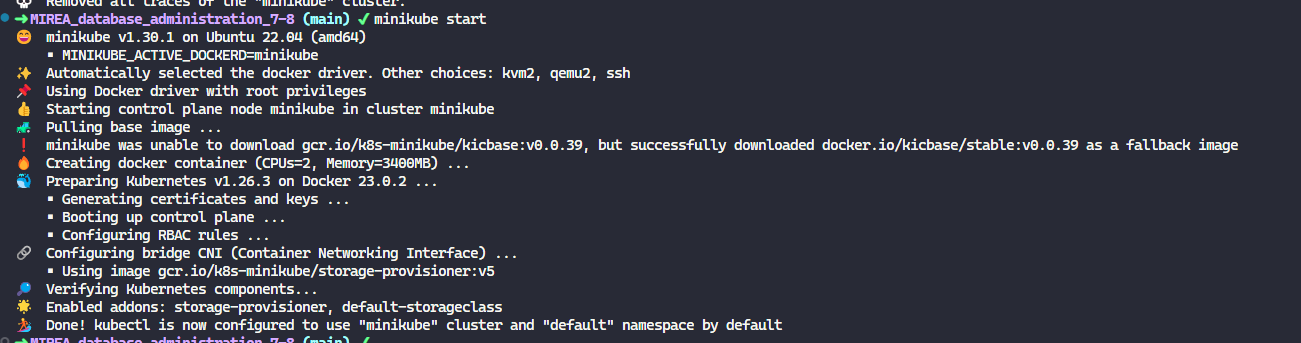
Приложение позволяет управлять товарами, категориями товаров и пользователями, и ролями.

Для категорий реализована иерархическая структура, для пользователей реализован RBAC.

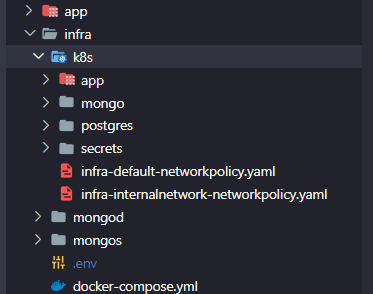
С кодом приложения и с файлами конфигурации можно ознакомиться здесь - <https://github.com/wallseat/MIREA_database_administration_7-8>

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

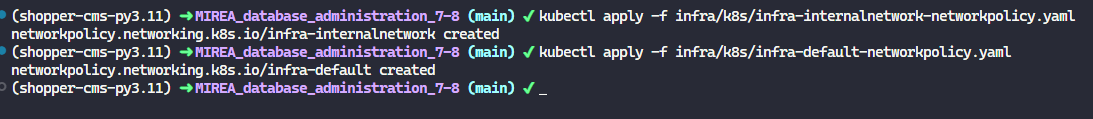
Сначала развернем minikube кластер по инструкции с [официального сайта](https://kubernetes.io/ru/docs/tasks/tools/install-minikube/). В качестве драйвера был выбран Docker.

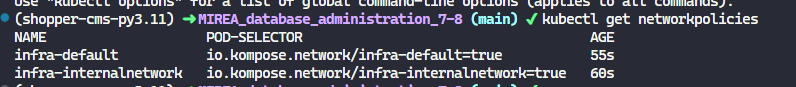


Перейдем в папку infra/k8s



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

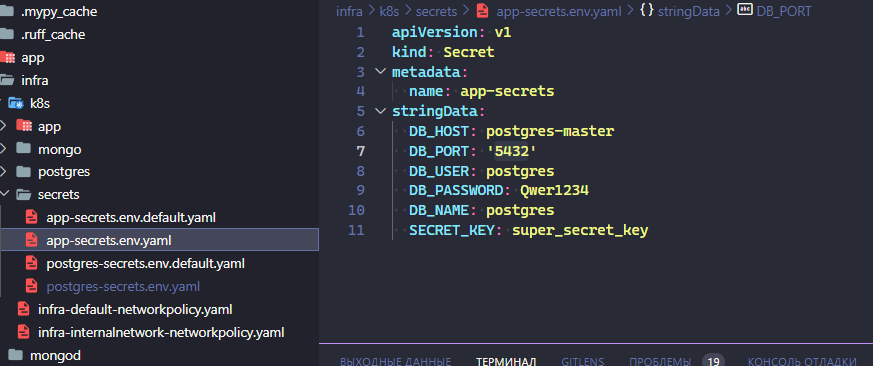


Проверяем  


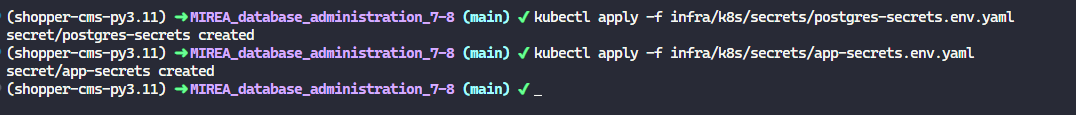
сети успешно созданы.

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

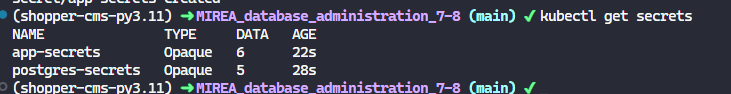
В данном примере, я не буду менять настройки, т.к. это тестовая среда для демонстрации, на продуктовом кластере секреты необходимо изменить!



Теперь так же применим их



Проверяем



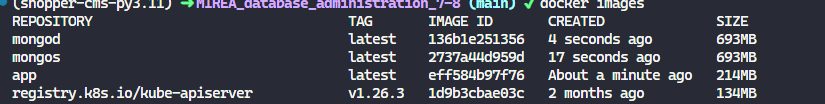
секреты успешно созданы.

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:

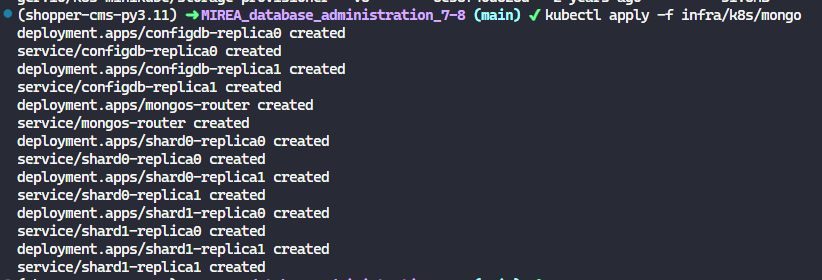
1. docker build -t app:latest app
2. docker build -t mongos:latest infra/mongos
3. docker build -t mongod:latest infra/mongod

Проверяем наличие образов в локальном регистре minikube

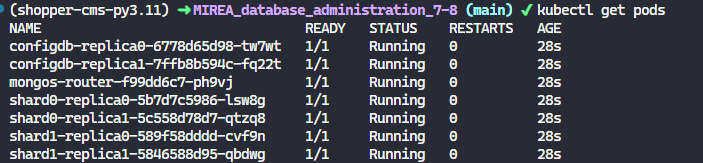


Образы собраны, теперь можем применять наши сервисы и деплойменты.

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

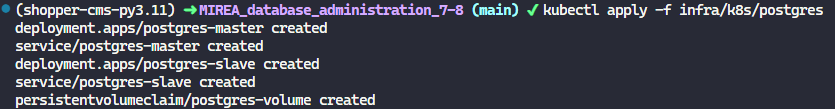


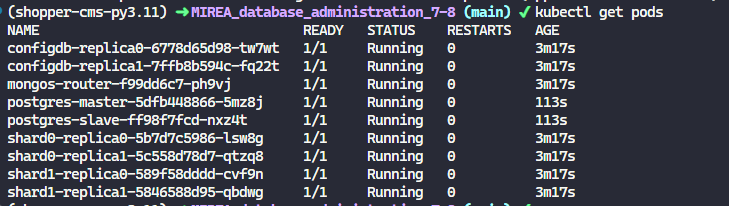
Проверяем поды



кластер mongo успешно развернут.

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



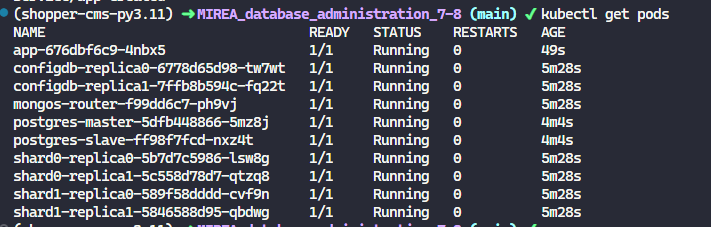
Проверяем поды  


кластер postgres успешно развернут.

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



Проверяем поды

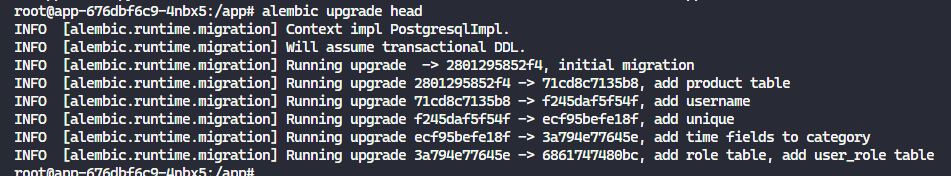


приложение успешно развернуто.

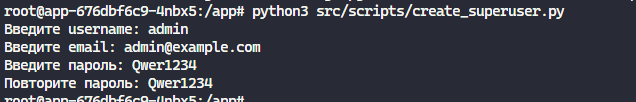
Теперь необходимо зайти на под с приложением и применить миграции. Используя команду kubectl exec -it <ИМЯ\_ПОДА> -- bash подключимся к консоли внутри нашего контейнера.



Теперь применим миграции командой alembic upgrade head

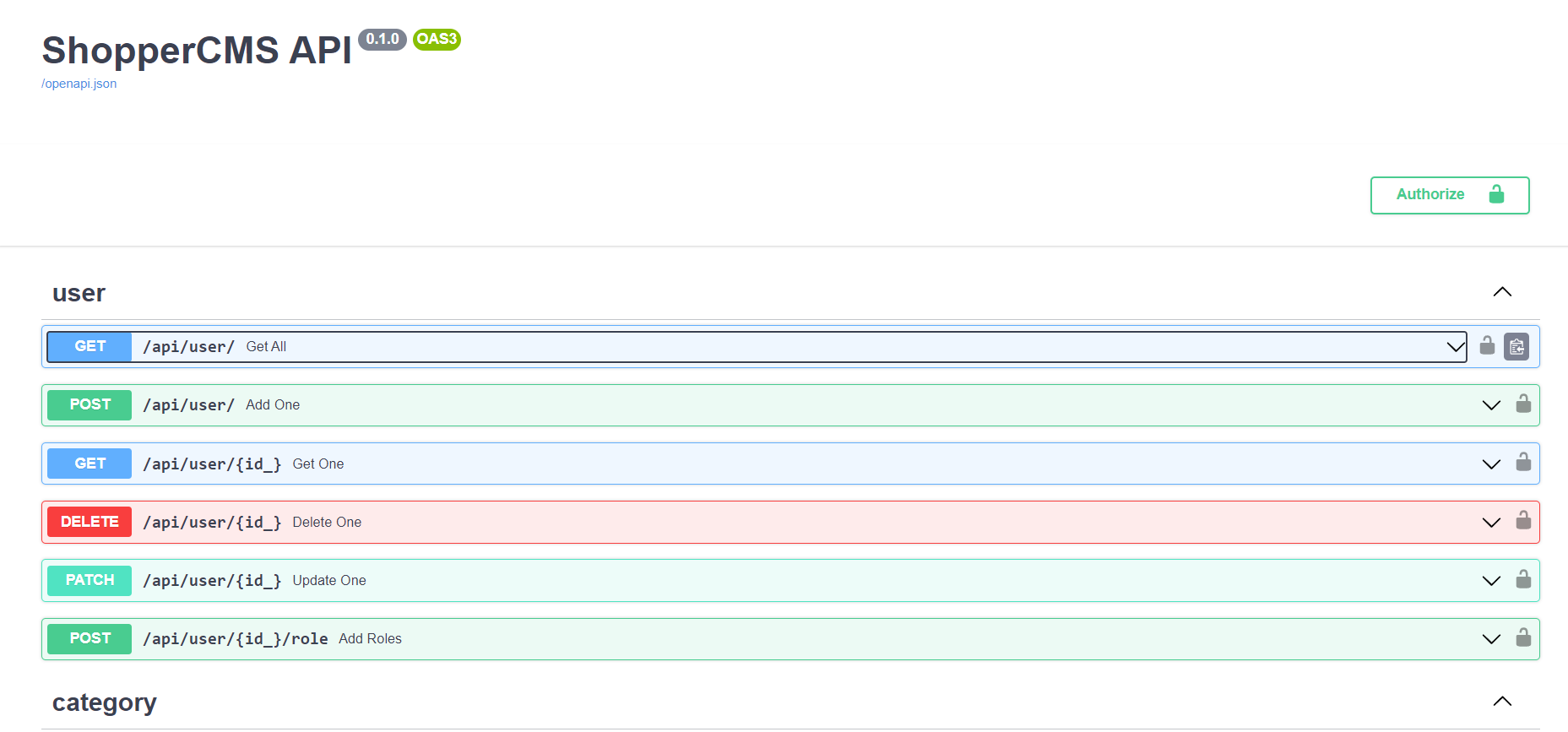


Теперь создадим пользователя администратора (создать его можно только используя скрипт, т.к. для создания пользователей из интерфейса необходимо иметь права, т.е. быть авторизованным). Для этого существует скрипт create\_superuser.py.



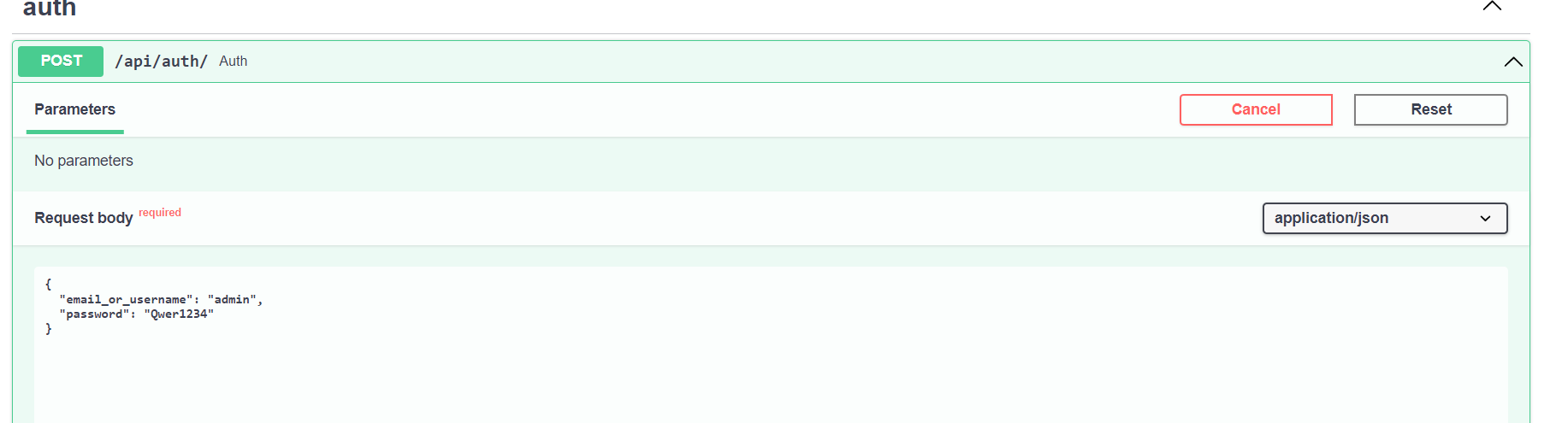
Пользователь успешно создан. Теперь мы можем воспользоваться нашим приложением.  
Чтобы получить доступ, необходимо прокинуть порт сервиса на наш локальный порт, сделать это можно через команду kubectl port-forward deployment/app 8080:8080

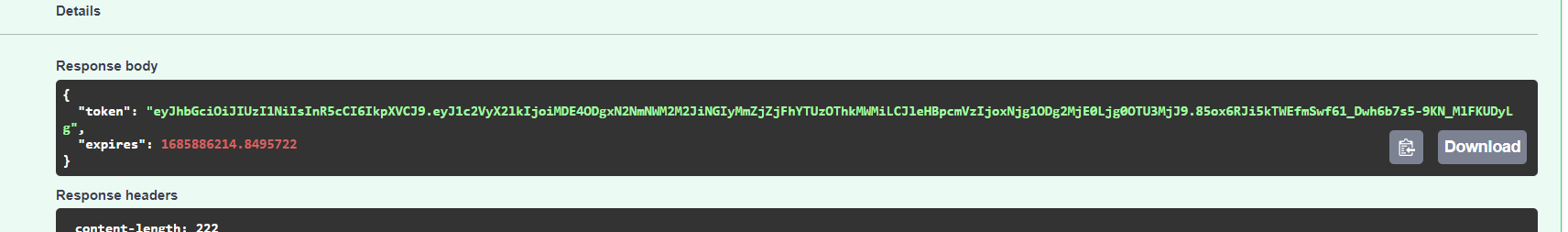
Перейдем по адресу localhost:8080/docs, чтобы открыть интерфейс swagger



Наше приложение доступно!

Теперь попробуем залогиниться под админской учеткой которую мы создали ранее





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

