

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ОТЧЕТ

по дисциплине «Интеграция и развертывание программного обеспечения с
помощью контейнеров»

Работа на лекции: Основы работы с Kubernetes

Направление подготовки – 38.03.05 «Бизнес-информатика».

профиль подготовки – «Аналитика данных и эффективное управление»

Выполнила:

St92

Руководитель:

Москва
2025 год

Цель работы: Получить практические навыки работы с кластером Kubernetes, включая развертывание базовых компонентов.

Задачи:

- Изучить основные концепции Kubernetes через практические вопросы.
- Научиться анализировать и применять манифесты Kubernetes.

Ход работы:

Задание 1. Теоретические основы Kubernetes.

Ответить на 3 случайных вопроса из репозитория:

Продemonстрировать понимание базовых концепций K8s

Kubernetes — это оркестратор контейнеров, который автоматизирует развертывание, масштабирование и управление приложениями

- Как проверить действительность манифеста?

Выполнить команду создания пода с `--dry-run` флагом, который на самом деле его не создаст, а проверит, и таким образом мы сможем обнаружить возможные проблемы с синтаксисом.

- Объясните, что такое Pod

Pod — это группа из одного или нескольких контейнеров, имеющих общее хранилище и сетевые ресурсы. Они запускаются на одной ноде и управляются как единое целое.

- Что такое "Deployment" в Kubernetes?

Deployment это объект Kubernetes, который управляет жизненным циклом пода. Deployment могут масштабировать количество реплик pod, включать контролируемое развертывание обновленного кода или откатываться к более ранней версии развертывания при необходимости.

Задание 2. Развертывание локального кластера на Kubernetes с использованием MiniKube

2.1. Установка minikube.

```
dev@dev-vm:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 119M 100 119M 0 0 26.8M 0 0:00:04 0:00:04 --:--:-- 27.7M
dev@dev-vm:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Рисунок 1 - Выполнение команды для установки minikube

```
dev@dev-vm:~$ cd /usr/local/bin
dev@dev-vm:/usr/local/bin$ ls
minikube
```

Рисунок 2 – Проверка установки minikube

Добавление пользователя в группу Docker.

```
dev@dev-vm:/usr/local/bin$ sudo usermod -aG docker $USER && newgrp docker
```

Рисунок 3 – Выполнение команды для добавления пользователя в группу

2.2. Установите kubectl <https://kubernetes.io/ru/docs/tasks/tools/install-kubectl/>

```
dev@dev-vm:~$ sudo snap install kubectl --classic
kubectl 1.32.3 from Canonical✓ installed
```

Рисунок 4 – Выполнение команды для установки kubectl

2.3. Убедитесь, что kubectl работает и произведите осмотрите кластера:
Клонируем учебный репозиторий в ОС:

```
dev@dev-vm:~/Downloads/seminar_1$ git clone https://github.com/BosenkoTM/CI_CD_25.git
Cloning into 'CI_CD_25'...
remote: Enumerating objects: 896, done.
remote: Counting objects: 100% (87/87), done.
remote: Compressing objects: 100% (82/82), done.
remote: Total 896 (delta 51), reused 5 (delta 5), pack-reused 809 (from 2)
Receiving objects: 100% (896/896), 3.83 MiB | 10.12 MiB/s, done.
Resolving deltas: 100% (301/301), done.
```

Рисунок 5 – Скачивание необходимого репозитория

Запустим minikube:

```

dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ minikube start --memory=2048mb --driver=docker
🐳 minikube v1.35.0 on Ubuntu 22.04 (vbox/amd64)
🔧 Using the docker driver based on user configuration
🔑 Using Docker driver with root privileges
👉 Starting "minikube" primary control-plane node in "minikube" cluster
📦 Pulling base image v0.0.46 ...
📦 Downloading Kubernetes v1.32.0 preload ...
> preloaded-images-k8s-v18-v1...: 333.57 MiB / 333.57 MiB 100.00% 24.51 M
> gcr.io/k8s-minikube/kicbase...: 500.31 MiB / 500.31 MiB 100.00% 20.30 M
🔥 Creating docker container (CPUs=2, Memory=2048MB) ...
🚢 Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌞 Enabled addons: storage-provisioner, default-storageclass
🏡 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

```

Рисунок 6 – Запуск minikube с параметрами

Выполним команды для проверки работы kubectl:

```

dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get node
NAME          STATUS    ROLES          AGE      VERSION
minikube      Ready     control-plane   8m13s    v1.32.0
dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get po
No resources found in default namespace.
dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get po -A
NAMESPACE     NAME                                                    READY   STATUS    RESTARTS   AGE
kube-system    coredns-668d6bf9bc-pmm4x                             1/1     Running   0           9m19s
kube-system    etcd-minikube                                           1/1     Running   0           9m24s
kube-system    kube-apiserver-minikube                                1/1     Running   0           9m23s
kube-system    kube-controller-manager-minikube                       1/1     Running   0           9m23s
kube-system    kube-proxy-pmxjv                                         1/1     Running   0           9m19s
kube-system    kube-scheduler-minikube                                1/1     Running   0           9m23s
kube-system    storage-provisioner                                     1/1     Running   1 (8m39s ago) 9m18s
dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get svc
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes    ClusterIP     10.96.0.1    <none>        443/TCP    9m32s

```

Рисунок 7 – Выполнение команд для проверки работы kubectl

2.4. Установите графический интерфейс Dashboard

<https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/> - необходимо выполнить шаги Deploying the Dashboard UI и Accessing the Dashboard UI.

Скачаем менеджер пакетов helm:

```

dev@dev-vm:~$ sudo snap install helm --classic
helm 3.17.2 from Snapcrafters installed

```

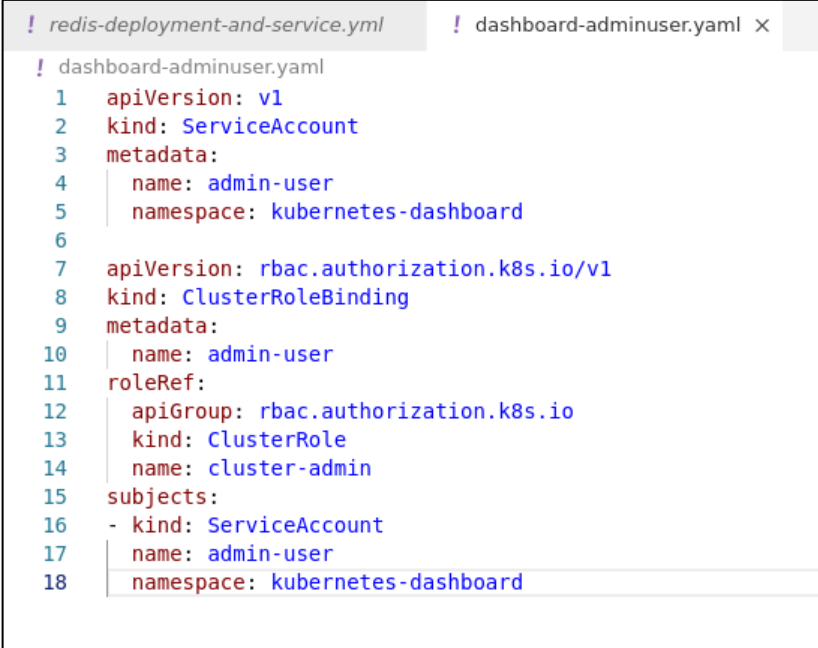
Рисунок 8 – Выполнение команды для установки менеджера пакетов helm

Выполним команды для установки Kubernetes-dashboard:

```
dev@dev-vm:~$ helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
"kubernetes-dashboard" has been added to your repositories
dev@dev-vm:~$ helm upgrade --install kubernetes-dashboard kubernetes-dashboard/k
ubernetes-dashboard --create-namespace --namespace kubernetes-dashboard
Release "kubernetes-dashboard" does not exist. Installing it now.
NAME: kubernetes-dashboard
LAST DEPLOYED: Sat Mar 29 13:22:06 2025
NAMESPACE: kubernetes-dashboard
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
*****
*****
*** PLEASE BE PATIENT: Kubernetes Dashboard may need a few minutes to get up and
```

Рисунок 9 – Выполнение команды для установки Kubernetes-dashboard

Создадим манифест для dashboard где укажем сервисный аккаунт и роль:



```
! redis-deployment-and-service.yml ! dashboard-adminuser.yaml x
! dashboard-adminuser.yaml
1  apiVersion: v1
2  kind: ServiceAccount
3  metadata:
4    name: admin-user
5    namespace: kubernetes-dashboard
6
7  apiVersion: rbac.authorization.k8s.io/v1
8  kind: ClusterRoleBinding
9  metadata:
10   name: admin-user
11  roleRef:
12   apiGroup: rbac.authorization.k8s.io
13   kind: ClusterRole
14   name: cluster-admin
15  subjects:
16  - kind: ServiceAccount
17    name: admin-user
18    namespace: kubernetes-dashboard
```

Рисунок 10 – Код для dashboard-adminuser.yaml

Внесем изменения в конфигурацию из файла dashboard-adminuser.yaml, проверим созданную роли и запросим токен для админа:


```

● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ eval $(minikube docker-env)
○ dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ docker build -t fastapi-app:local .
2025/03/29 14:29:31 in: []string{}
2025/03/29 14:29:31 Parsed entitlements: []
[+] Building 1.5s (1/2)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 239B
=> [internal] load metadata for docker.io/library/python:3.10

```

Рисунок 14 – Выполнение команд для создания локального образа в миникубе

Создадим ресурсы в Kubernetes кластере:

```

● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl create -f configmap.yml
configmap/fastapi-config created
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl create -f secret.yml
secret/fastapi-secret created
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl create -f fastapi-deployment-and-service.yml
deployment.apps/fastapi-deployment created
service/fastapi-service created
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl create -f redis-deployment-and-service.yml
deployment.apps/redis-deployment created
service/redis-service created

```

Рисунок 15 – Выполнение команд для создания ресурсов

Проверим создание ресурсов командами:

```

● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
fastapi-deployment-cf4dc69bc-ncgtp  1/1     Running   0           11m
fastapi-deployment-cf4dc69bc-rhm4l  1/1     Running   0           11m
redis-deployment-748ffbc5f5-tgfql   1/1     Running   0           11m
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get configmaps
NAME          DATA   AGE
fastapi-config 1       11m
kube-root-ca.crt 1       97m
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get secrets
NAME          TYPE      DATA   AGE
fastapi-secret Opaque    1       11m
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
fastapi-deployment 2/2     2             2           11m
redis-deployment  1/1     1             1           11m
● dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl get services
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
fastapi-service  NodePort    10.107.51.43  <none>        80:30001/TCP     11m
kubernetes      ClusterIP   10.96.0.1     <none>        443/TCP          97m
redis-service   ClusterIP   10.101.85.156 <none>        6379/TCP         11m

```

Рисунок 16 – Выполнение команд для проверки создания ресурсов

Обратимся к <https://localhost:8443> для просмотра dashboard

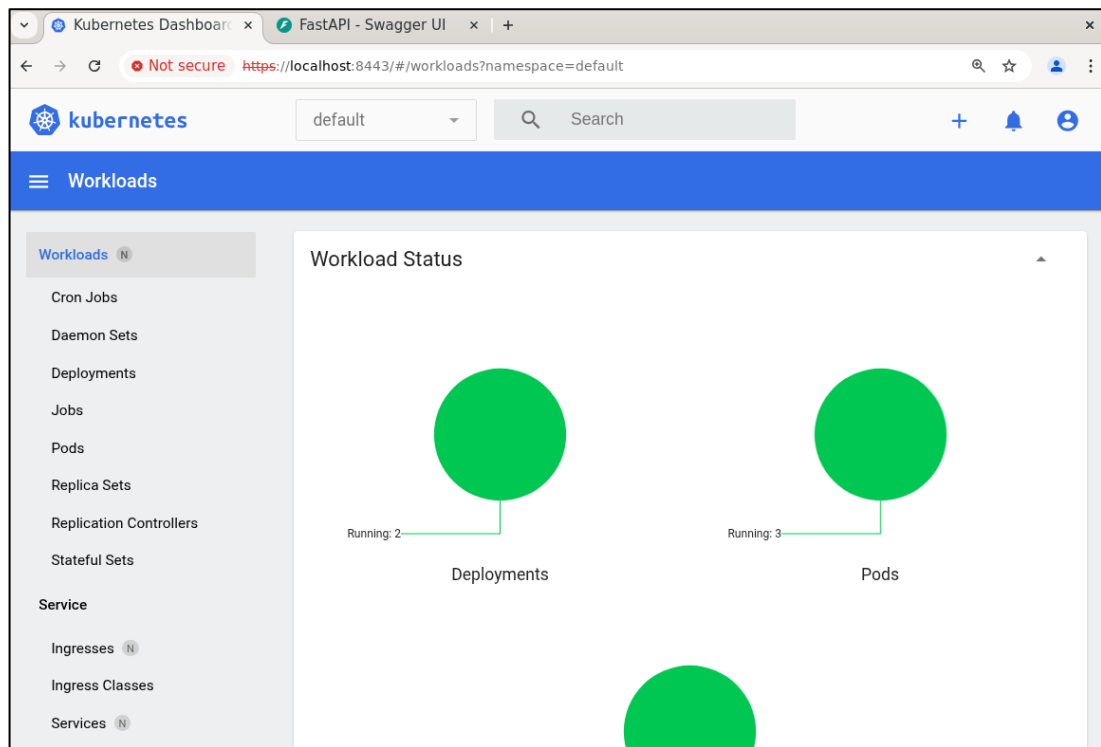


Рисунок 17 – Вывод dashboard-kubernetes

Обратимся к FastAPI, убедимся в успешном подключении.

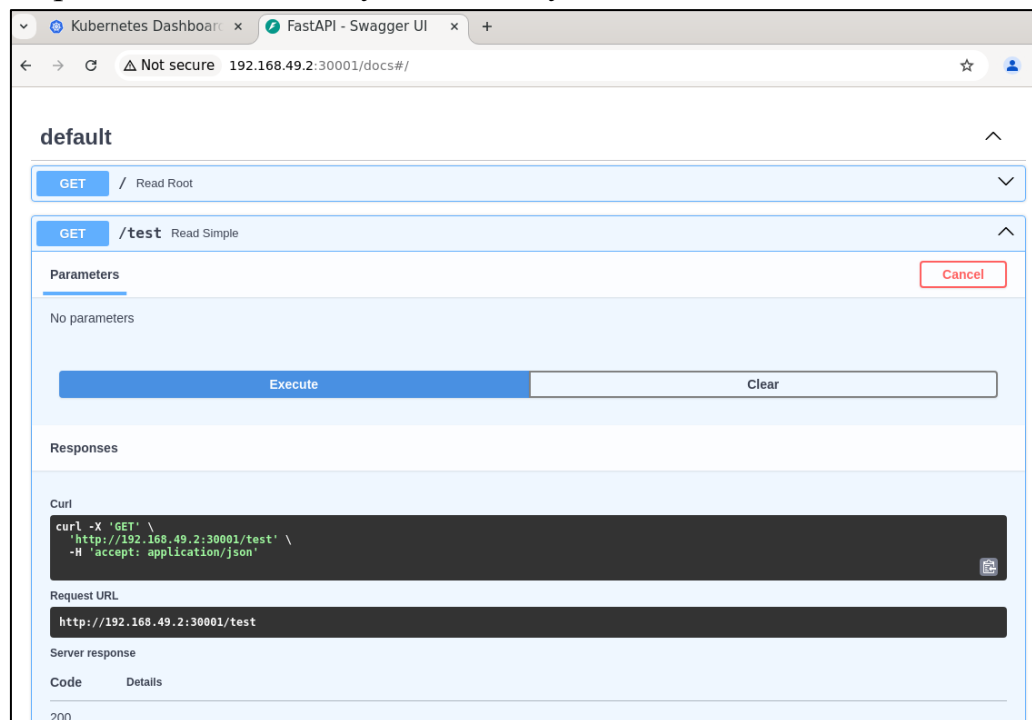


Рисунок 18 – Вывод FastAPI

Выведем подробную информацию о поде fastapi


```

dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl describe pod fastapi-deployment-cf4dc69bc-ncgtp
Name:          fastapi-deployment-cf4dc69bc-ncgtp
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Sat, 29 Mar 2025 14:32:23 +0300
Labels:        app=fastapi
               pod-template-hash=cf4dc69bc
Annotations:   <none>
Status:        Running
IP:            10.244.0.9
IPs:           IP: 10.244.0.9
Controlled By: ReplicaSet/fastapi-deployment-cf4dc69bc
Init Containers:

```

Рисунок 19 – Выполнение команды для вывода подробной информации

Выведем информацию о конфигурации:

```

dev@dev-vm:~/Downloads/seminar_1/CI_CD_25/practice/lab4_1$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/dev/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Sat, 29 Mar 2025 13:06:19 MSK
        provider: minikube.sigs.k8s.io
        version: v1.35.0
        name: cluster_info
    server: https://192.168.49.2:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Sat, 29 Mar 2025 13:06:19 MSK
        provider: minikube.sigs.k8s.io
        version: v1.35.0
        name: context_info
    namespace: default
    user: minikube
  name: minikube

```

Рисунок 20 – Выполнение команды для вывода информации о конфигурации

Индивидуальное задание:

Вариант 5 – Mongo.

1. Запустите Kubernetes локально (k3s или minikube). Проверьте работу системных контейнеров и приложите скриншот команды: `kubectl get po -n kube-system`.

Выполним создание миникуба с параметрами:

```

dev@dev-vm:~/Desktop/lab_4_1_sam$ minikube start --memory=2048mb --driver=docker
minikube v1.35.0 on Ubuntu 22.04 (vbox/amd64)
Using the docker driver based on user configuration
Using Docker driver with root privileges
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Creating docker container (CPUs=2, Memory=2048MB) ...
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  Generating certificates and keys ...
  Booting up control plane ...
  Configuring RBAC rules ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: default-storageclass, storage-provisioner
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

```

Рисунок 21—Выполнение команды для создания миникуба

Билдим локальный образ и загружаем его в миникуб:

```

dev@dev-vm:~/Desktop/lab_4_1_sam$ eval $(minikube docker-env)
dev@dev-vm:~/Desktop/lab_4_1_sam$ docker build -t fastapi-app:local .
2025/03/31 14:18:43 in: []string{}
2025/03/31 14:18:43 Parsed entitlements: []
[+] Building 2.4s (1/2)
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 239B
=> [internal] load metadata for docker.io/library/python:3.10

```

Рисунок 22 – Выполнение команды для создания образа в миникуб

Создадим манифесты:

```

dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl create -f configmap.yml
configmap/fastapi-config created
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl create -f secret.yml
secret/fastapi-secret created
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl create -f fastapi-deployment-and-service.yml
deployment.apps/fastapi-deployment created
service/fastapi-service created
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl create -f mongo-deployment-and-service.yml
error: the path "mongo-deployment-and-service.yml" does not exist
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl create -f mongo-deployment-and-service.yaml
error: error parsing mongo-deployment-and-service.yaml: error converting YAML to JSON: yaml:
  allowed in this context
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl create -f mongo-deployment-and-service.yaml
deployment.apps/mongo-deployment created
service/mongo-service created

```

Рисунок 23 – Выполнение команд для создания ресурсов

Посмотрим dashboard-kubernetes:

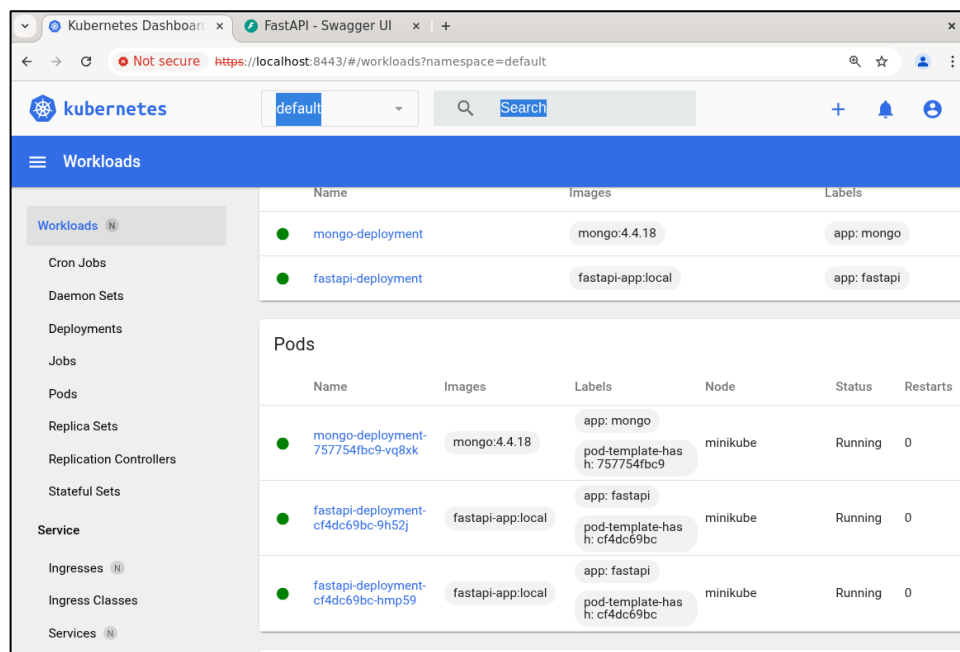


Рисунок 24 – Вывод dashboard-kubernetes

Выполним команду для просмотра подов:

```

Cdev@dev-vm: ~/Desktop/lab_4_1_sam$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
fastapi-deployment-cf4dc69bc-9h52j  1/1     Running   0           11m
fastapi-deployment-cf4dc69bc-hmp59  1/1     Running   0           11m
mongo-deployment-757754fbc9-vq8xk   1/1     Running   0           10m

```

Рисунок 25 – Выполнение команды для просмотра подов

Посмотрим список подов в системном неймспейсе Kubernetes:

```

Cdev@dev-vm: ~/Desktop/lab_4_1_sam$ kubectl get po -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-668d6bf9bc-qxw47           1/1     Running   0           23m
etcd-minikube                       1/1     Running   0           23m
kube-apiserver-minikube             1/1     Running   0           23m
kube-controller-manager-minikube    1/1     Running   0           23m
kube-proxy-tp4qd                    1/1     Running   0           23m
kube-scheduler-minikube             1/1     Running   0           23m
storage-provisioner                 1/1     Running   1 (22m ago) 23m

```

Рисунок 26 – Выполнение команды kubectl get po -n kube-system

2. Имеется YAML с деплоем для **mongo**. Измените файл:
 - Запустите без пароля (уберите переменную аутентификации);
 - Фиксируйте образ на версии **4.2**;
 - Добавьте Service для доступа к базе.

Приложите итоговый YAML.

Изменим yaml для mongo:

```
! mongo-deployment-and-service.yaml x ! secret.yml main.py requi
! mongo-deployment-and-service.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mongo-deployment
5    labels:
6      app: mongo
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: mongo
12    template:
13     metadata:
14       labels:
15         app: mongo
16     spec:
17       containers:
18       - name: mongo
19         image: mongo:4.2
20         ports:
21         - containerPort: 27017
22         command: ["mongod", "--bind_ip_all"]
```

Рисунок 27 – Deployment MongoDB

```
! mongo-deployment-and-service.yaml x ! secret.yml
! mongo-deployment-and-service.yaml
7  spec:
12   template:
16     spec:
17       containers:
18       - name: mongo
22         command: ["mongod", "--bind_ip_
23   ---
24   apiVersion: v1
25   kind: Service
26   metadata:
27     name: mongo-service
28   spec:
29     selector:
30       app: mongo
31     ports:
32     - protocol: TCP
33       port: 27017
34       targetPort: 27017
```

Рисунок 28 – Service mongoDB

Удалим манифест и создадим его заново.

Проверим доступность MongoDB, используя FastAPI:

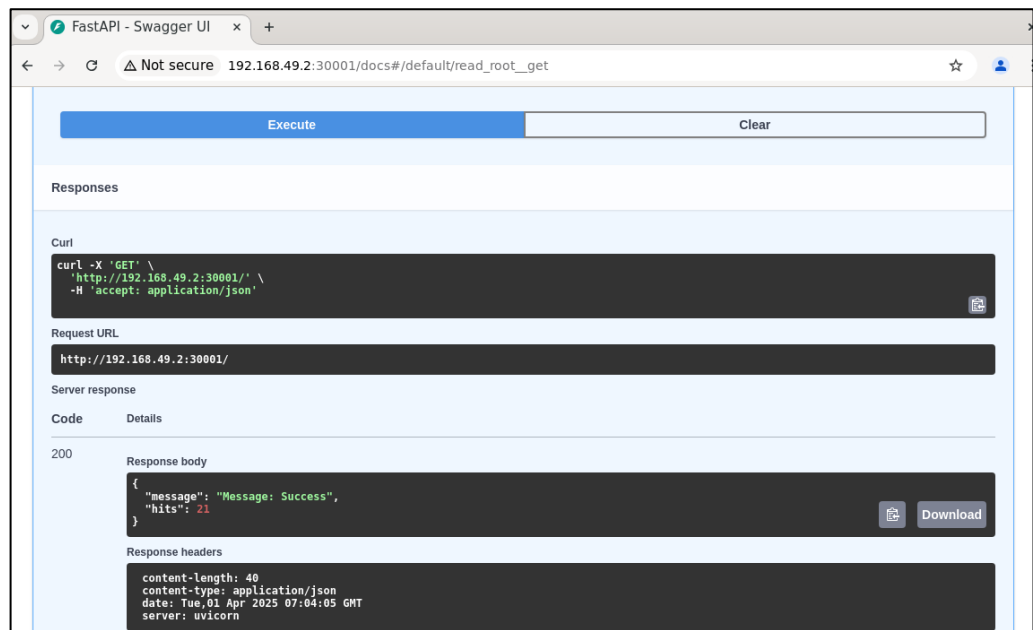


Рисунок 29 – Ответ MongoDB

3. Напишите команды kubectl для контейнера:

- Выполнить **ps aux** внутри контейнера;
- Просмотреть логи за 5 минут;
- Удалить pod;
- Пробросить порт для отладки

Выполним команду ps aux:

```
dev@dev-vm: ~/Desktop/lab_4_1_sam$ kubectl exec -it mongo-deployment-56fc879c5c-wnztck -- ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.4   2.7 1588208 110464 ?        Ssl   06:58   0:05 mongod --bind_ip_all
root        43   0.0   0.0   34416   2816 pts/0    Rs+   07:21   0:00 ps aux
```

Рисунок 30 – Выполнение команды для вывода информации о процессах

Посмотрим логи за 5 минут и за все время:

```
dev@dev-vm: ~/Desktop/lab_4_1_sam$ kubectl logs --since=5m mongo-deployment-56fc879c5c-wnztck
dev@dev-vm: ~/Desktop/lab_4_1_sam$ kubectl logs mongo-deployment-56fc879c5c-wnztck
2025-04-01T06:58:25.541+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisable
dProtocols 'none'
2025-04-01T06:58:25.556+0000 W ASIO [main] No TransportLayer configured during NetworkInterface startup
2025-04-01T06:58:25.556+0000 I CONTROL [initandlisten] MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-bit host=mong
o-deployment-56fc879c5c-wnztck
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] db version v4.2.24
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] git version: 5e4ec1d24431fcd28b579a024c5c801b8cde4e2
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] allocator: tcmalloc
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] modules: none
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] build environment:
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] distmod: ubuntu1804
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] distarch: x86_64
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] target arch: x86_64
2025-04-01T06:58:25.557+0000 I CONTROL [initandlisten] options: { net: { bindIp: "*" } }
2025-04-01T06:58:25.558+0000 I STORAGE [initandlisten]
2025-04-01T06:58:25.558+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the
WiredTiger storage engine
2025-04-01T06:58:25.558+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2025-04-01T06:58:25.558+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=1446M,cache_overflow=(file
_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archi
ve=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250)
,statistics_log=(wait=0),verbose=[recovery progress,checkpoint progress],
2025-04-01T06:58:25.891+0000 I STORAGE [initandlisten] WiredTiger message [1743490705:891660][1:0x70a3980e0b00], txn-recove
r: Set global recovery timestamp: (0, 0)
2025-04-01T06:58:26.049+0000 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2025-04-01T06:58:26.140+0000 I STORAGE [initandlisten] Timestamp monitor starting
2025-04-01T06:58:26.179+0000 I CONTROL [initandlisten]
```

Рисунок 31- Выполнение команд для просмотра логов

Удалим под и заметим, что он снова запустится:

```
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl delete pod mongo-deployment-56fc879c5c-wnztk
pod "mongo-deployment-56fc879c5c-wnztk" deleted
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
fastapi-deployment-cf4dc69bc-c68xk	1/1	Running	0	30m
fastapi-deployment-cf4dc69bc-qpqtj	1/1	Running	0	30m
mongo-deployment-56fc879c5c-k67zr	1/1	Running	0	12s

Рисунок 32 – Выполнение команды для удаления пода

Пробросим порт для отладки:

```
dev@dev-vm:~/Desktop/lab_4_1_sam$ kubectl port-forward pod/mongo-deployment-56fc879c5c-k67zr 27017:27017
Forwarding from 127.0.0.1:27017 -> 27017
Forwarding from [::1]:27017 -> 27017
```

Рисунок 33 – Выполнение команды для проброса порта

Выводы:

В ходе работы мы получили практические навыки работы с кластером Kubernetes, включая развертывание базовых компонентов. Также нам удалось настроить мониторинг через Dashboard Kubernetes.