

Департамент образования города Москвы

Государственное автономное образовательное учреждение  
высшего образования города Москвы  
«Московский городской педагогический университет»

Институт цифрового образования  
Департамент информатики, управления и технологий

**Практическая работа на вебинаре 29.03**  
**по дисциплине «Интеграция и развертывание программного**  
**обеспечения с помощью контейнеров»**

**Тема: «Основы работы с Kubernetes»**

Направление подготовки 38.03.05 – бизнес-информатика  
Профиль подготовки «Аналитика данных и эффективное управление»  
(очная форма обучения)

Выполнила:  
Студентка группы АДЭУ-211  
St\_88

Москва  
2025

## ВВЕДЕНИЕ

Цель работы: получить практические навыки работы с кластером Kubernetes, включая развертывание базовых компонентов, настройку мониторинга и работу с service mesh.

Задачи:

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

**Вариант 22** (т.к. 1 вариант был рассмотрен в качестве примера, а нас в группе где-то 16 человек, значит 22 вариант точно не занят).

### Вариант 22. Kubernetes. Часть 1 (redis с репликой)

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

Имеется YAML с деплоем для **redis**.  
Измените файл:  
– Запуск без пароля (`ALLOW_EMPTY_PASSWORD=yes`);  
– Фиксируйте образ на версии **6.0.15**;  
– Установите `replicas: 2`;  
– Добавьте Service для доступа.  
Приложите итоговый YAML.

Напишите команды kubectl для контейнера:  
– Выполнить команду `ps aux` внутри pod;  
– Просмотреть логи за 5 минут;  
– Удалить pod;  
– Пробросить локальный порт для отладки.

Доп. задание\*: Создайте YAML для:  
– ConfigMap с настройками для **redis**;  
– Deployment, использующий ConfigMap;  
– Ingress, направляющий запросы по пути `/cache` на сервис.

## Теоретические основы Kubernetes

### 1. Что такое Kubernetes? Почему организации его используют?

Kubernetes (K8s) — это платформа для оркестрации контейнеров, которая автоматизирует развертывание, масштабирование и управление контейнеризированными приложениями.

#### Основные возможности Kubernetes

- автоматическое масштабирование (может увеличивать/уменьшать количество подов (Pods) в зависимости от нагрузки);
- отказоустойчивость и самовосстановление (если контейнер падает, Kubernetes автоматически перезапускает его. Если узел (Node) выходит из строя, поды перезапускаются на других узлах);
- балансировка нагрузки и сервис-дискавери (Service автоматически распределяет трафик между подами, Ingress управляет внешним доступом к сервисам)
- управление конфигурацией и секретами (ConfigMaps и Secrets позволяют хранить настройки и чувствительные данные отдельно от кода);
- гибкость развертывания;
- мультиоблачность и гибридные среды.

Компании используют Kubernetes, т.к. он позволяет автоматически добавлять ресурсы при росте нагрузки, обеспечивает отказоустойчивость, поддерживает локальный запуск приложений и облачный, DevOps и CI/CD.

Однако он сложен в понимании, развертывании и обеспечении, поэтому должны быть реально веские обоснованные экономические причины для его использования.

### 2. Что такое кластер Kubernetes?

Кластер Kubernetes — это набор узлов (серверов), объединенных в единую систему для запуска и управления контейнеризированными

приложениями. Он обеспечивает автоматическое развертывание, масштабирование и отказоустойчивость приложений.

Кластер состоит из двух типов узлов:

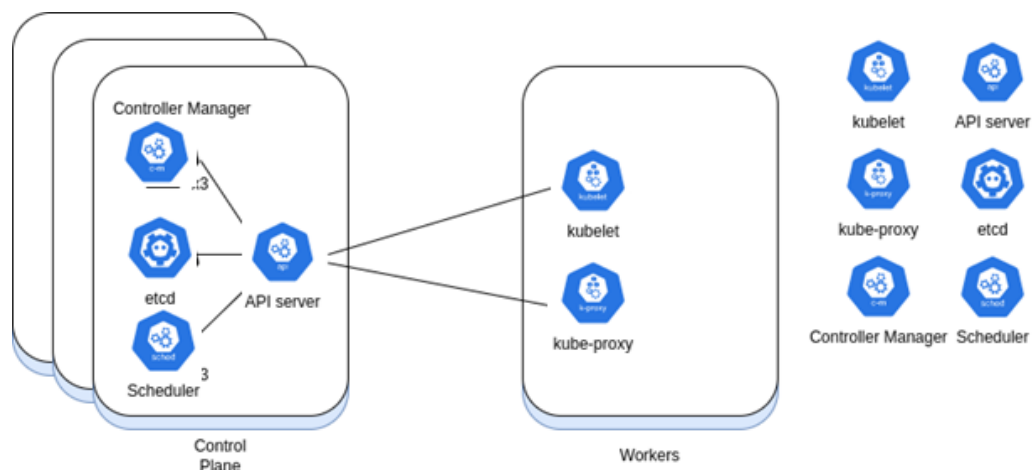
1. Control Plane (Master-узлы) - управляющая часть кластера, отвечающая за:

- API Server — входная точка для управления (через kubectl);
- Scheduler — решает, на каком узле запускать поды (Pods);
- Controller Manager — следит за состоянием кластера (репликация, исправность узлов);
- etcd — распределенное хранилище конфигураций и состояния кластера.

2. Worker Nodes (Рабочие узлы) - на них запускаются пользовательские приложения. Каждый worker включает:

- Kubelet — агент, который общается с Control Plane и управляет контейнерами;
- Kube-Proxy — обеспечивает сетевую маршрутизацию и балансировку нагрузки;
- Container Runtime (Docker, containerd, CRI-O) — запускает контейнеры.

3. Разместите компоненты с правой стороны изображения в нужном месте чертежа:



## Развертывание локального кластера на Kubernetes с использованием MiniKube

1. Т.к. второе задание индивидуального варианта подразумевает внесение изменений в файл YAMLс деплоем, то, чтобы не перезапускать миникуб, вносятся правки в файл сразу. Итак, необходимо для деплоя с redis:
  - обеспечить запуск без пароля (1);
  - зафиксировать образ версии на 6.0.15 (2);
  - установить replicas:2 (3);
  - добавить Service для доступа (4) – предоставляет Service для доступа к Redis на порту 6379

```
! redis-deployment-and-service.yml ●
! redis-deployment-and-service.yml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: redis-deployment
5    labels:
6      app: redis
7  spec:
8    replicas: 2 3
9    selector:
10     matchLabels:
11       app: redis
12   template:
13     metadata:
14       labels:
15         app: redis
16     spec:
17       containers:
18         - name: redis
19           image: bitnami/redis:6.0.15 2
20           env:
21             - name: ALLOW_EMPTY_PASSWORD 1
22               value: "yes"
23           ports:
24             - containerPort: 6379
25
26  apiVersion: v1
27  kind: Service
28  metadata:
29    name: redis-service 4
30  spec:
31    selector:
32      app: redis
33    ports:
34      - protocol: TCP
35        port: 6379
36        targetPort: 6379
```

## 2. Установка minikube:

```
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ 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 10.9M 0 0:00:10 0:00:10 --:--:-- 18.4M
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ █

mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
[sudo] password for mgpu:
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ █
```

## 3. Добавление пользователя в группу Docker, для предотвращения возможных ошибок недостаточно прав при обращении к kubeconfig:

```
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ sudo usermod -aG docker $USER && newgrp docker
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ █
```

## 4. Установка kubectl – командного инструмента для управления кластерами:

```
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ sudo snap install kubectl --classic
kubectl 1.32.3 from Canonical✓ installed
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ █
```

## 5. Запуск minikube с заданием параметров памяти, дабы избежать масштабирования до пределов ресурсов ПК:

```
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ minikube start --memory=2048mb --driver=docker
🐹 minikube v1.35.0 on Ubuntu 20.04 (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% 9.20 Mi
> gcr.io/k8s-minikube/kicbase...: 500.31 MiB / 500.31 MiB 100.00% 9.32 Mi
🔥 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
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ █
```

## 6. Настройки окружения командной строки Ubuntu для работы с Docker, который управляется Minikube:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ eval $(minikube docker-env)
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

## 7. Билд локального образа и его загрузка в Minikube:

```
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ docker build -t fastapi-app:local .
=> => resolve docker.io/library/python:3.10@sha256:8ad0e578e1b733f2a496b41f179175679374191a9c7ab8c63156446094a9cda8
=> => sha256:7cd785773db44407e20a679ce5439222e505475eed5b99f1910eb2cda51729ab 48.47MB / 48.47MB
=> => sha256:091eb8249475f42de217265c501e0186f0a3ea7490ef7f51458c30db91fb3cac 24.01MB / 24.01MB
=> => sha256:8ad0e578e1b733f2a496b41f179175679374191a9c7ab8c63156446094a9cda8 9.08kB / 9.08kB
=> => sha256:52c63d169c27d32435ff634ea772d5ca52c9d0793bb796e97b0977c582642727 2.33kB / 2.33kB
=> => sha256:bc0fc5e29abb0921de96874560ac409bb8e131545fa8623a27dd3791decf8ceb 6.18kB / 6.18kB
=> => sha256:255774e0027b72d2327719e78dbad5ad8c9cf446d055e45be7fc149418470bae 64.40MB / 64.40MB
=> => sha256:353e14e5cc47664fba714a7da288001d90427c705494847ac773f5cc08199451 211.35MB / 211.35MB
=> => sha256:0c64566c7562a4e1405a59f7b95f25b2b74a9a630b8b4b5916d3829a81e90ab4 6.16MB / 6.16MB
=> => extracting sha256:7cd785773db44407e20a679ce5439222e505475eed5b99f1910eb2cda51729ab
=> => sha256:57161121b343e07415ad5fddf4d3b635176622126bab8ff18e653439c9619f29 21.38MB / 21.38MB
=> => sha256:a90d73ac0e516c2cd69b099f3b5f957c2815844e088d741d737c95e7111d249c 249B / 249B
=> => extracting sha256:091eb8249475f42de217265c501e0186f0a3ea7490ef7f51458c30db91fb3cac
=> => extracting sha256:255774e0027b72d2327719e78dbad5ad8c9cf446d055e45be7fc149418470bae
=> => extracting sha256:353e14e5cc47664fba714a7da288001d90427c705494847ac773f5cc08199451
=> => extracting sha256:0c64566c7562a4e1405a59f7b95f25b2b74a9a630b8b4b5916d3829a81e90ab4
=> => extracting sha256:57161121b343e07415ad5fddf4d3b635176622126bab8ff18e653439c9619f29
=> => extracting sha256:a90d73ac0e516c2cd69b099f3b5f957c2815844e088d741d737c95e7111d249c
=> [internal] load build context
=> => transferring context: 125.81MB
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt requirements.txt
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:22f1d5fccef2447c9c60f25c63702a9bc8ca47a0e859be41374dfef4b32a5221

mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ kubectl create -f configmap.yml
configmap/fastapi-config created
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ kubectl create -f secret.yml
secret/fastapi-secret created
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ kubectl create -f fastapi-deployment-and-service.yml
deployment.apps/fastapi-deployment created
service/fastapi-service created
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$ kubectl create -f redis-deployment-and-service.yml
deployment.apps/redis-deployment created
service/redis-service created
mgpu@mgpu-VirtualBox:~/Downloads/lab4_1$
```

## 8. Проверка, что запустились все поды:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
fastapi-deployment-cf4dc69bc-6kk9h  1/1     Running   0           86s
fastapi-deployment-cf4dc69bc-8p6rh  1/1     Running   0           86s
redis-deployment-5dbcf44b5c-4gr7h   1/1     Running   0           77s
redis-deployment-5dbcf44b5c-5l7bg   1/1     Running   0           77s
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

## 9. Проверка, что запустились все сервисы:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl get services
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
fastapi-service   NodePort    10.108.198.191  <none>        80:30001/TCP     2m48s
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP          8m16s
redis-service     ClusterIP   10.103.138.255  <none>        6379/TCP         2m39s
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

## 10. Два поднятых деплоимента:

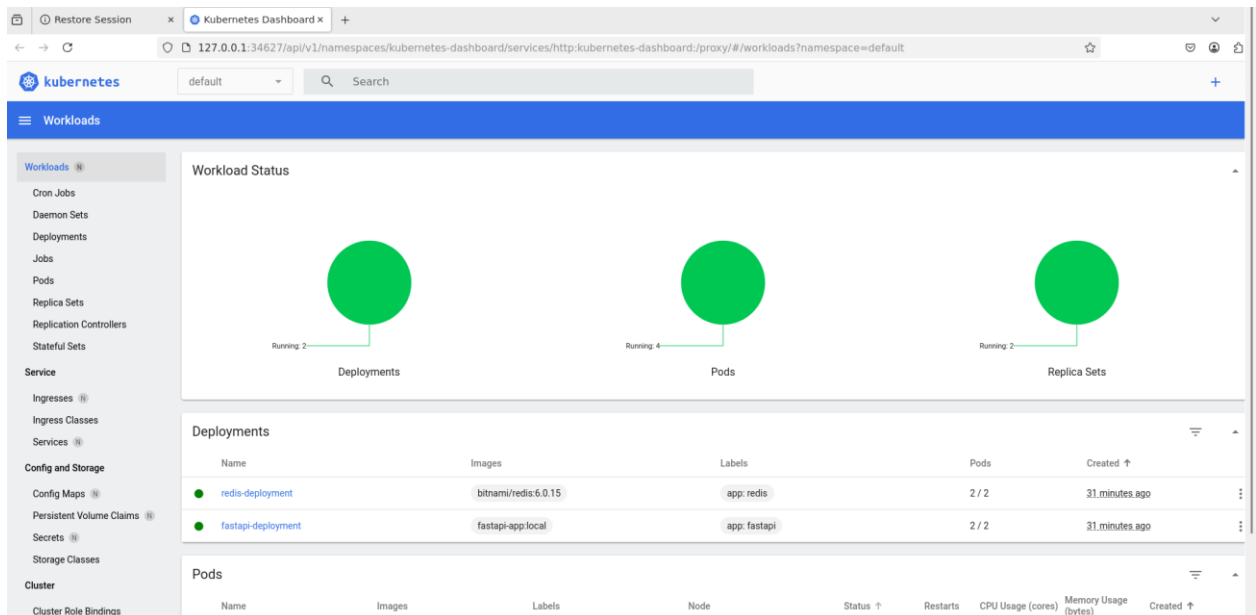
```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
fastapi-deployment  2/2     2             2           4m45s
redis-deployment    2/2     2             2           4m36s
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

## 11. Запуск графической оболочки Minikube:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ minikube dashboard
🔧 Enabling dashboard ...
  ■ Using image docker.io/kubernetesui/dashboard:v2.7.0
  ■ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡 Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

😄 Verifying dashboard health ...
🚀 Launching proxy ...
😄 Verifying proxy health ...
🔗 Opening http://127.0.0.1:34627/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
[]
```



Те же результаты, что и при отработке команд.

## 12. Архитектура запущенных ресурсов:

Minikube Node (192.168.49.2)

```
├─ System Pods (kube-system)
│   └─ etcd, kube-apiserver, kube-controller-manager, kube-scheduler
│       └─ kube-proxy (DaemonSet)
│           └─ coredns (Deployment)
│
```



```

└─ User Applications (default)
    └─ FastAPI
        └─ Pods: 10.244.0.3, 10.244.0.4
        └─ Service (NodePort: 30001)
    └─ Redis
        └─ Pods: 10.244.0.5, 10.244.0.6
        └─ Service (ClusterIP: 6379)
└─ Kubernetes Dashboard (kubernetes-dashboard)
    └─ Pods: 10.244.0.7, 10.244.0.8
└─ Service (ClusterIP: 80)

```

### Описание архитектуры:

#### 1. Системные компоненты (namespace kube-system)

- etcd-minikube - хранилище данных кластера (key-value база);
- kube-apiserver-minikube - основной API Kubernetes;
- kube-controller-manager-minikube - управляет контроллерами;
- kube-scheduler-minikube - отвечает за распределение Pod'ов по нодам.
- kube-proxy-k4t4f (DaemonSet) - обеспечивает сетевую маршрутизацию и балансировку;
- storage-provisioner - динамическое выделение PV (Persistent Volumes) в Minikube.

#### 2. Пользовательские приложения:

- FastAPI (веб-приложение) - Deployment: fastapi-deployment  
Реплики: 2 (fastapi-deployment-cf4dc69bc-cqvfb и fastapi-deployment-cf4dc69bc-kflsf);  
Образ: fastapi-app:local (локально собранный);
- Service: fastapi-service (NodePort) - порт: 80:30001;
- Redis (база данных) - Deployment: redis-deployment;

Реплики: 2 (redis-deployment-5dbcf44b5c-gvm5g и redis-deployment-5dbcf44b5c-jc87w)

Образ: bitnami/redis:6.0.15;

— Service: redis-service (ClusterIP) - Порт: 6379 (внутренний доступ по DNS-имени redis-service.default.svc.cluster.local).

3. Dashboard Kubernetes (namespace kubernetes-dashboard) -веб-интерфейс для управления кластером:

— Deployment: kubernetes-dashboard

— Deployment: dashboard-metrics-scraper;

— Service: kubernetes-dashboard (ClusterIP) - порт: 80 (доступ через minikube dashboard).

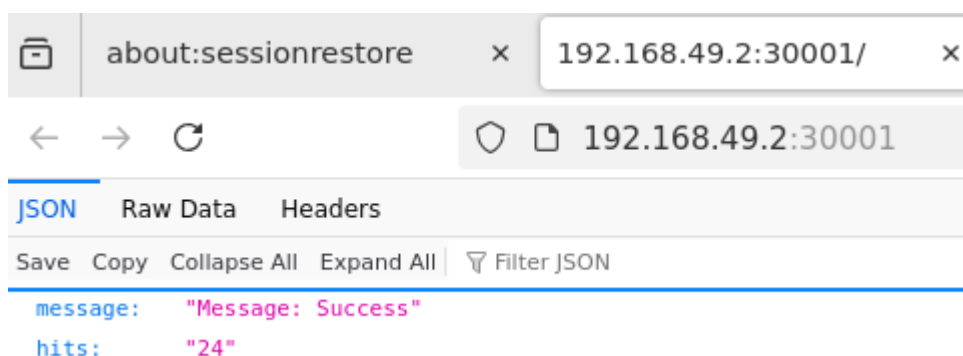
4. Сетевые службы:

— service/kubernetes - встроенный сервис Kubernetes API;

— service/kube-dns - DNS-сервер кластера.

13.Проверка доступности FastAPI – узнаем порт, переходим:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ minikube service fastapi-service --url  
http://192.168.49.2:30001
```



FastAPI - Swagger UI x +

Not Secure 192.168.49.2:30001/docs#/default/read\_root\_get

# FastAPI

0.1.0 OAS 3.1

/openapi.json

## default

GET / Read Root

Parameters

No parameters

Responses

Code	Description
200	Successful Response

Media type

application/json

Controls Accept header.

Example Value

Schema

"string"

GET /test Read Simple

#### 14. Проверим доступность redis на порту TCP 6379:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ minikube service redis-service --url
😿 service default/redis-service has no node port
! Services [default/redis-service] have type "ClusterIP" not meant to be exposed, however for local development minikube allows you to access this !
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

Как видно команда, аналогичная предыдущей, не отработала для redis т.к. он доступен на внутреннем порту, внутри кластера. Поэтому можно проверить доступность следующим методом -подключиться к redis изнутри кластера — успешно (ответ PONG):

```
• mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl exec redis-deployment-5dbcf44b5c-4gr7h -- redis-cli -h redis-service -p 6379 PING
PONG
○ mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

15. В соответствии с заданием 1 необходимо запустить команду `kubectl get po -n kube-system`, которая выводит список всех подов, работающих в пространстве имен `kube-system` в Kubernetes-кластере:

```
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl get po -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-668d6bf9bc-clwt9           1/1     Running   0           13m
etcd-minikube                       1/1     Running   0           13m
kube-apiserver-minikube             1/1     Running   0           13m
kube-controller-manager-minikube    1/1     Running   0           13m
kube-proxy-b864b                   1/1     Running   0           13m
kube-scheduler-minikube             1/1     Running   0           13m
storage-provisioner                 1/1     Running   1 (12m ago) 13m
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

16. В соответствии с 3 заданием необходимо:

— выполнить команду `ps aux` внутри `pod` — показать список всех запущенных процессов с детальной информацией внутри пода:

```
● mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl exec redis-deployment-5dbcf44b5c-4gr7h -- ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
1001         1  0.2  0.1 128600  7864 ?        Ssl  22:12   0:04 redis-server 0.0.0.0:6379
1001        60  0.0  0.0   7644  2756 ?        Rs   22:53   0:00 ps aux
○ mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

— просмотреть логи за 5 минут (для всех подов с `label = redis`):

```
● mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl logs -l app=redis --since=5m
1:M 03 Apr 2025 22:52:48.026 * 10 changes in 300 seconds. Saving...
1:M 03 Apr 2025 22:52:48.027 * Background saving started by pid 59
59:C 03 Apr 2025 22:52:48.033 * DB saved on disk
59:C 03 Apr 2025 22:52:48.034 * RDB: 0 MB of memory used by copy-on-write
1:M 03 Apr 2025 22:52:48.127 * Background saving terminated with success
1:M 03 Apr 2025 22:52:48.051 * 10 changes in 300 seconds. Saving...
1:M 03 Apr 2025 22:52:48.053 * Background saving started by pid 46
46:C 03 Apr 2025 22:52:48.062 * DB saved on disk
46:C 03 Apr 2025 22:52:48.063 * RDB: 0 MB of memory used by copy-on-write
1:M 03 Apr 2025 22:52:48.154 * Background saving terminated with success
○ mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$
```

— удалить `pod` — для этого сначала запустим тестовый под `redis`:

```
● mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl run -it --rm redis-test --image=redis:6.0.15 --restart=Never -- redis-cli -h redis-service -p 6379
If you don't see a command prompt, try pressing enter.
□
```

```

mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
fastapi-deployment-cf4dc69bc-6kk9h  1/1     Running   0           36m
fastapi-deployment-cf4dc69bc-8p6rh  1/1     Running   0           36m
redis-deployment-5dbcf44b5c-4gr7h   1/1     Running   0           36m
redis-deployment-5dbcf44b5c-5l7bg   1/1     Running   0           36m
redis-test                           1/1     Running   0           28s
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$

```

И теперь удалим:

```

mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl delete pod redis-test
pod "redis-test" deleted

mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl get pod redis-test
Error from server (NotFound): pods "redis-test" not found
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$

```

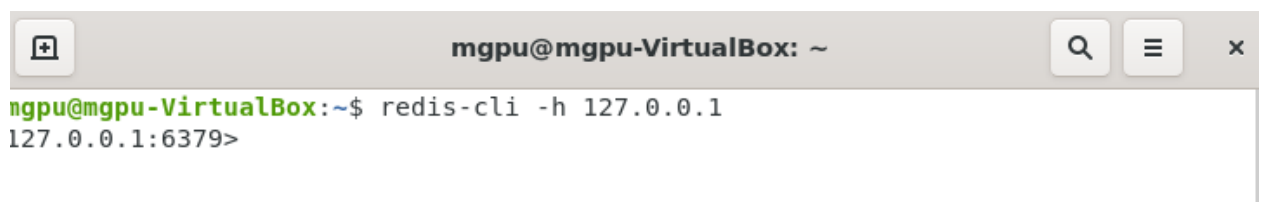
— Пробросить локальный порт для отладки (например, позволит подключиться к Redis из локальной машины через `redis-cli -h 127.0.0.1`):

```

mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl port-forward svc/redis-service 6379:6379
Forwarding from 127.0.0.1:6379 -> 6379
Forwarding from [::1]:6379 -> 6379

```

Проверка через терминал:



```

mgpu@mgpu-VirtualBox: ~
mgpu@mgpu-VirtualBox:~$ redis-cli -h 127.0.0.1
127.0.0.1:6379>

```

```

mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ kubectl port-forward svc/redis-service 6379:6379
Forwarding from 127.0.0.1:6379 -> 6379
Forwarding from [::1]:6379 -> 6379
Handling connection for 6379

```

17. Удаляем Minikube:

```

mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$ minikube delete
🔥 Deleting "minikube" in docker ...
🔥 Deleting container "minikube" ...
🔥 Removing /home/mgpu/.minikube/machines/minikube ...
💡 Removed all traces of the "minikube" cluster.
mgpu@mgpu-VirtualBox:~/CI_CD_25-main/practice/lab4_1$

```

## ВЫВОДЫ

В ходе выполнения практической работы на вебинаре были достигнуты поставленные цели:

- были изучены основные концепции Kubernetes через практические вопросы;
- научились анализировать и применять манифесты Kubernetes;
- был развернут локальный кластер с использованием Minikube, состоящий из пользовательских приложений – fastapi и redis.

Таким образом, была выполнена поставленная цель работы - получить практические навыки работы с кластером Kubernetes, включая развертывание базовых компонентов, настройку мониторинга и работу с service mesh.

Было усвоено, что для развертывания Kubernetes в компании (ни в коем случае не в мелкой или средне – только на крупной предприятии) нужна команда инженеров минимум из 20 человек, а также само применение технологии должно быть экономически оправдано. Иначе, вместо прибыли Kubernetes будет приносить только убытки.