

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ОТЧЕТ

по расчётно-графической работе по дисциплине “Технологии виртуализации”

Выполнил студент Селиванов В. В.

Ф.И.О.

Группы ИБ-022

Работу принял

ассистент кафедры ВС
Романюта А. А.

подпись

Защищена

Оценка

Новосибирск – 2024

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ	3
Установка программного обеспечения	4
Развертывание приложения	7
1.1 Описание приложения	7
1.2 Развертывание приложения при помощи сущности Deployment.....	7
1.3 Обращение к приложению через сервис	8
ЗАКЛЮЧЕНИЕ.....	10
ПРИЛОЖЕНИЕ.....	11

ПОСТАНОВКА ЗАДАЧИ

Установить minikube. Развернуть приложение в minikube. Приложение может быть как из задания лабораторной работы #1, так и на выбор (Например traefik/whoami). Развертывание при помощи сущности kubernetes - Deployment. Продемонстрировать работу приложения в kubernetes. Создать сущность kubernetes – Service, для обращения к приложению, развернутому в Deployment. Обратиться к приложению через сервис. Изменить количество реплик приложения. Проверить, что обращение через сервис работает

Установка программного обеспечения

Для запуска minikube используется Docker Desktop, установленный с официального сайта

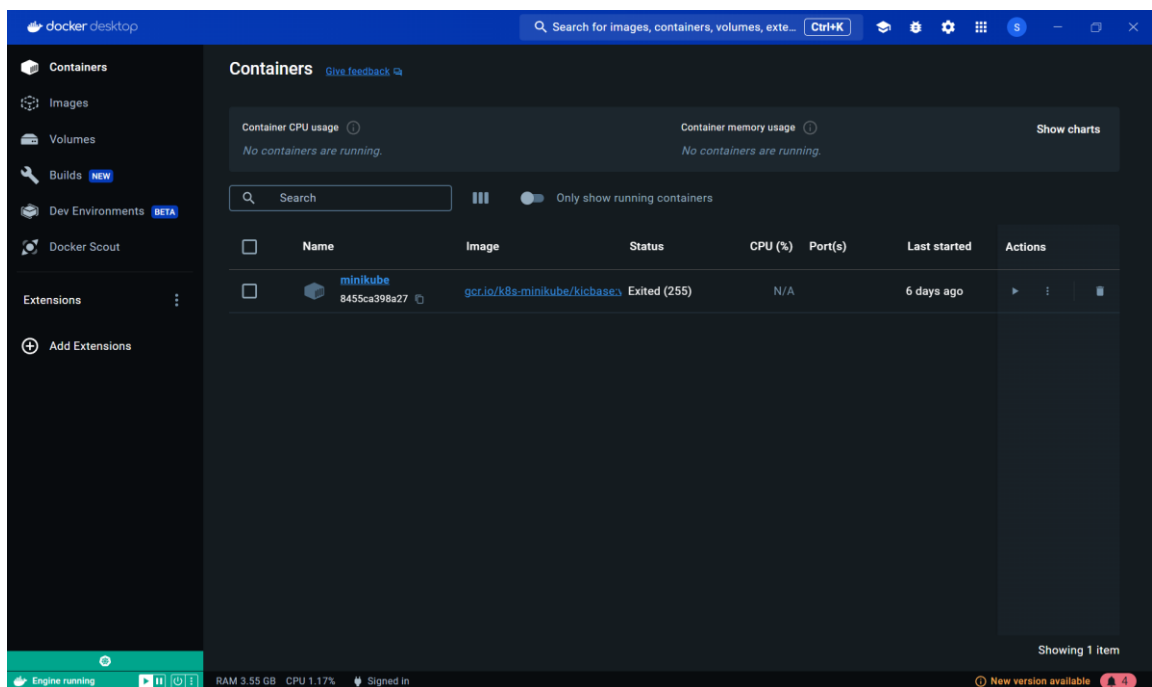


Рисунок 1 – интерфейс установленного Docker Desktop

Minikube установлен при помощи скачанного с официального сайта исполняемого файла

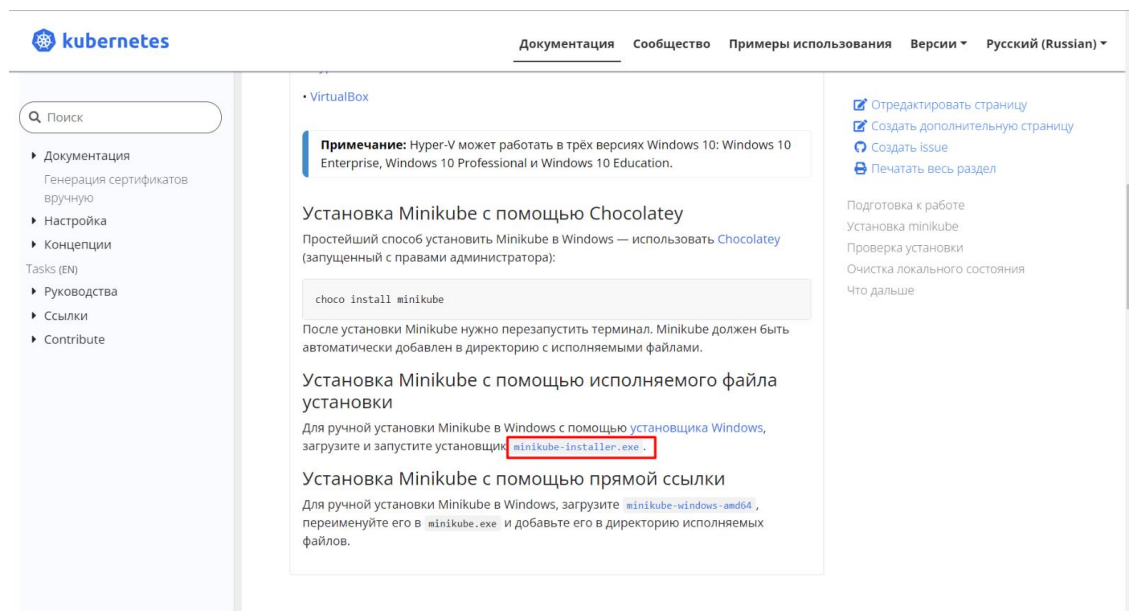
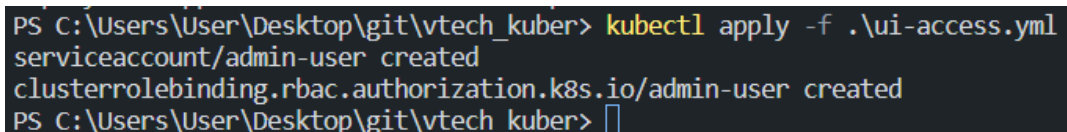


Рисунок 2 – скачивание исполняемого файла minikube

Также, для удобного отслеживания состояния кластера, установлен Kubernetes Dashboard. Kubernetes Dashboard – удобный инструмент для получения сведений о состоянии кластера Kubernetes и базового управления им. Для получения доступа к веб интерфейсу инструмента, необходимо применить манифест следующего содержания:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```



```
PS C:\Users\User\Desktop\git\vtch_kuber> kubectl apply -f .\ui-access.yml
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
PS C:\Users\User\Desktop\git\vtch_kuber> █
```

Рисунок 3 – применение манифеста для получения доступа к веб интерфейсу
Kubernetes Dashboard

Затем, извлечь т.н. барьерный токен для созданного при помощи манифеста сервисного аккаунта

[illegible]

Рисунок 4 – извлечение барьерного токена для сервисного аккаунта

После выполнения команды `kubectl proxy`, необходимо в адресной строке браузера выполнить запрос и ввести полученный токен по адресу `http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/`

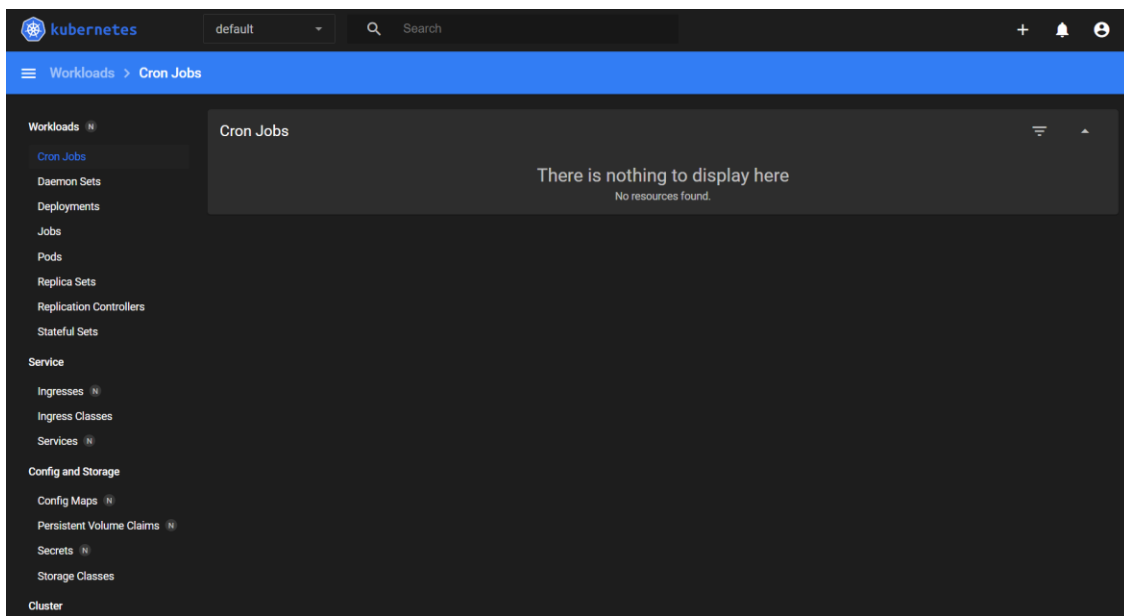


Рисунок 5 – веб интерфейс Kubernetes Dashboard

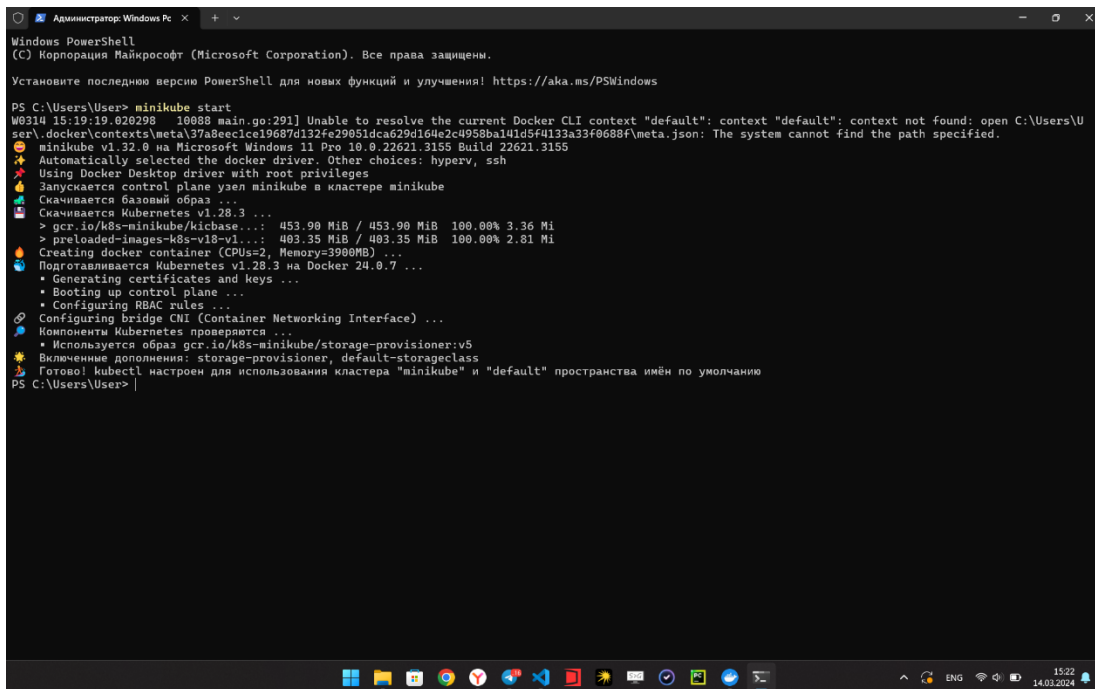


Рисунок 6 – запуск minikube

Развертывание приложения

1.1 Описание приложения

Для развертывания написано простейшее серверное приложение на языке программирования Python (с использованием модуля Flask), возвращающее по запросу строку «Hello World!» и адрес хоста. Код реализации представлен в приложении

1.2 Развертывание приложения при помощи сущности Deployment

Для того, чтобы развернуть приложение при помощи сущности Kubernetes Deployment, необходимо написать следующий текст манифеста:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: server-app
          image: selvnv/vtech:1.3
          ports:
            - containerPort: 8000
```

В нем указано, что необходимо поддерживать 3 реплики приложения (пода с меткой my-app), созданных из образа, загруженного на Docker Hub, прослушивающих соединение на порту 8000

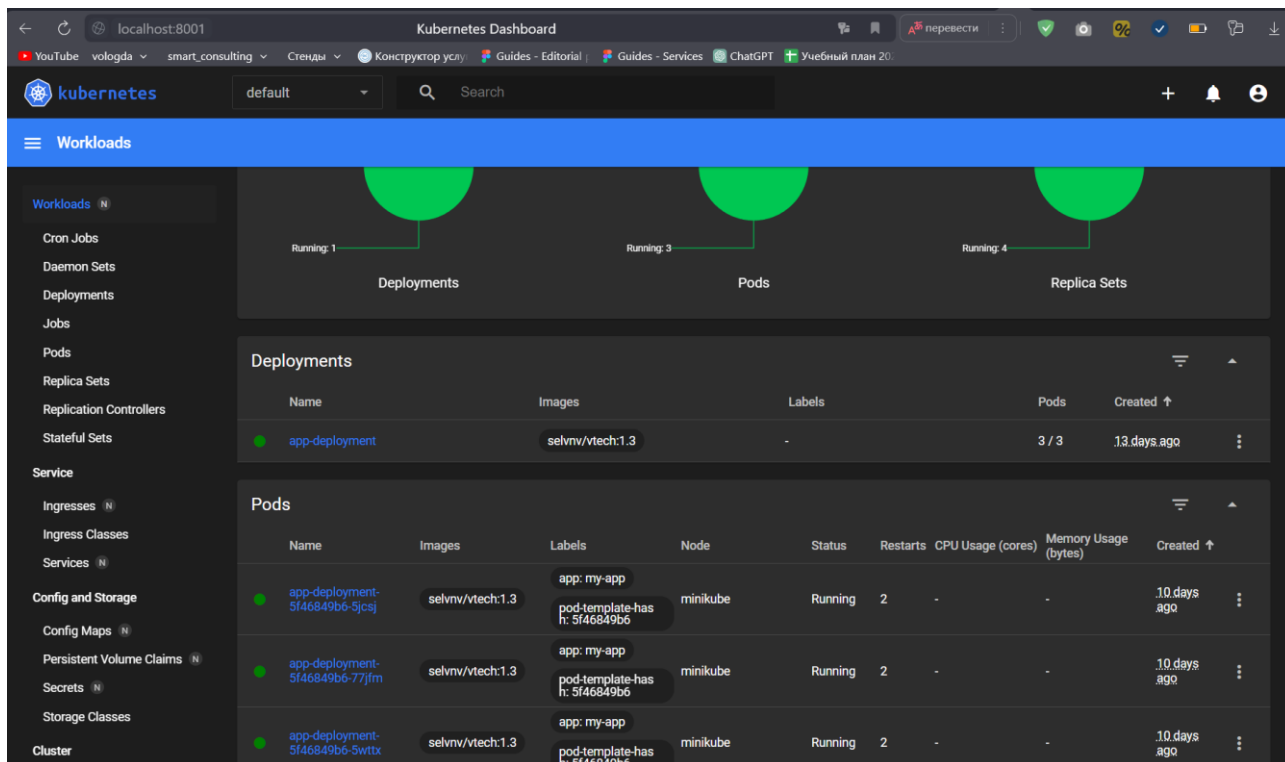


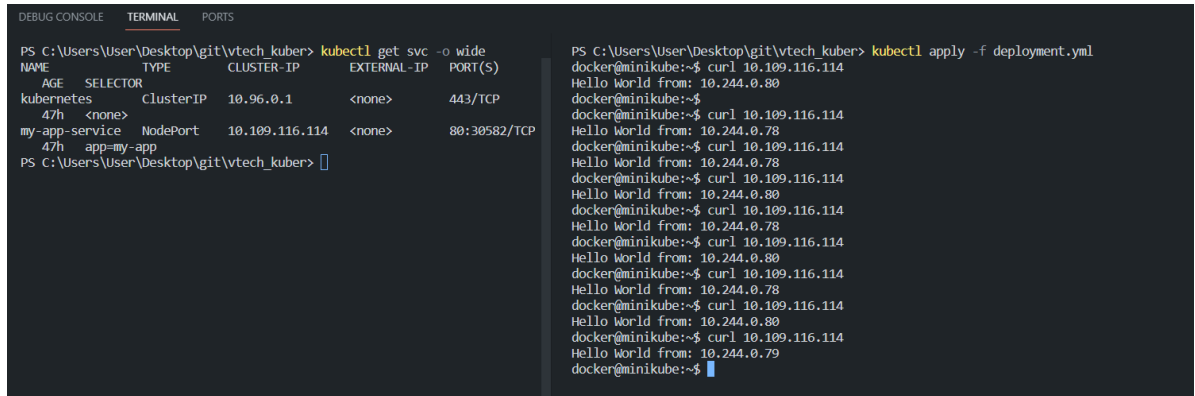
Рисунок 7 – демонстрация развернутого приложения при помощи интерфейса
Kubernetes Dashboard

1.3 Обращение к приложению через сервис

Для обращения к приложению через сервис, необходимо применить манифест следующего содержания:

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
  type: NodePort
```


В нем указано, что при обращении к объекту сервиса, запрос будет перенаправляться к подам, имеющим метку my-app, на порт 8000. В результате, при выполнении запроса при помощи утилиты curl на адрес сервиса, реплики приложения возвращают в ответ строку «Hello World!» и адрес хоста. Адреса хоста разные из-за балансировки нагрузки

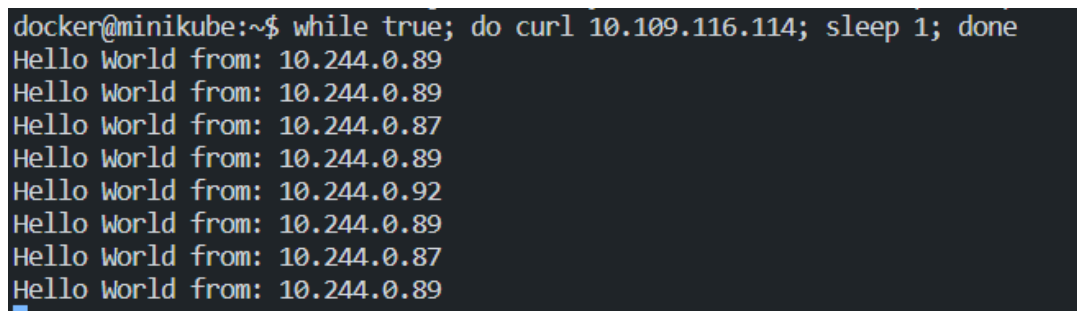


```
DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\User\Desktop\git\vttech_kuber> kubectl get svc -o wide
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)
kubernetes ClusterIP   10.96.0.1     <none>        443/TCP
my-app-service NodePort    10.109.116.114 <none>        80:30582/TCP
47h      app=my-app
PS C:\Users\User\Desktop\git\vttech_kuber>

PS C:\Users\User\Desktop\git\vttech_kuber> kubectl apply -f deployment.yml
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.80
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.78
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.78
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.80
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.78
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.80
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.78
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.80
docker@minikube:~$ curl 10.109.116.114
Hello World from: 10.244.0.79
docker@minikube:~$
```

Рисунок 8 – проверка обращения к приложению через сервис



```
docker@minikube:~$ while true; do curl 10.109.116.114; sleep 1; done
Hello World from: 10.244.0.89
Hello World from: 10.244.0.89
Hello World from: 10.244.0.87
Hello World from: 10.244.0.89
Hello World from: 10.244.0.92
Hello World from: 10.244.0.89
Hello World from: 10.244.0.87
Hello World from: 10.244.0.89
```

Рисунок 9 – проверка обращения к приложению через сервис с использованием цикла while

ЗАКЛЮЧЕНИЕ

В результате выполнения работы развернуто приложение в minikube, а также реализовано обращение к приложению через объект Kubernetes Service. Продемонстрирована работа приложения в Kubernetes

ПРИЛОЖЕНИЕ

Код приложения

```
from flask import Flask
import socket

app = Flask(__name__)

@app.route('/')
def hello():
    # Возвращаем контент в виде строки
    ip_address = socket.gethostbyname(socket.gethostname())
    return f'Hello World from: {ip_address}\n'

if __name__ == '__main__':
    # Запуск сервера. Порт 8000, перезагрузка и вывод ошибок на
    # странице в режиме отладки
    app.run(host='0.0.0.0', port=8000, debug=True)
```

Текст Dockerfile

```
# Используем базовый образ Python версии 3.9
FROM python:3.9-slim

RUN pip install Flask

# Копируем файлы вашего приложения в текущую директорию
COPY src/app.py /

# Определяем команду для запуска приложения при старте контейнера
CMD ["python", "app.py"]
```

Текст манифеста deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: server-app
          image: selvnv/vtech:1.3
          ports:
            - containerPort: 8000
```

Текст манифеста service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

```
type: NodePort
```