Министерство цифрового развития, связи и
массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Отчет

по расчетно-графической работе

по дисциплине «**Технологии виртуализации**»

Выполнил:
студент гр. ИС-142 _____ /Григорьев Ю.В./

Проверил:
преподаватель _____ /Романюта А.А./

Новосибирск 2025

ВВЕДЕНИЕ

Целью данной работы было приобретение практических навыков работы с Kubernetes — платформой для оркестрации контейнеров. Задание включало следующие этапы:

1. Запуск Kubernetes-кластера (допускалась любая реализация, например, k3s/k3d).
2. Развертывание приложения (HTTP-сервер, который возвращает случайное число от 1 до 100) с использованием сущности Deployment.
3. Создание Service для доступа к приложению и проверка его работоспособности при изменении количества реплик.
4. Настройка Ingress-контроллера (Traefik или Nginx) и добавление Ingress для обращения к приложению по домену.
5. Добавление в кластер еще одного рабочего узла (worker).
6. Настройка Readiness и Liveness Probes для приложения.
7. Установка веб-приложения с помощью Helm и создание Helm-чарта для приложения из пункта 1.
8. Демонстрация масштабирования приложения путем изменения количества реплик.

Работа выполнялась на macOS с использованием инструментов: Docker (для контейнеризации), k3d (для создания легковесного Kubernetes-кластера), kubectl (для управления кластером), Helm (для управления приложениями) и curl (для тестирования HTTP-запросов). В качестве приложения был разработан HTTP-сервер на основе Flask, предоставляющий два эндпоинта: / (возвращает случайное число от 1 до 100) и /health (служит для проверки состояния приложения).

ВЫПОЛНЕНИЕ РАБОТЫ

Настройка Kubernetes-кластера с помощью k3d

Для выполнения задания был выбран k3d — инструмент, позволяющий создавать легковесные Kubernetes-кластеры в Docker-контейнерах, что идеально подходит для локальной разработки на macOS.

Создан новый кластер mycluster с двумя рабочими узлами (agents) и пробросом порта 80 для внешнего доступа:

```
k3d cluster create mycluster --agents 2 -p "80:80@loadbalancer"
```

Проверка работоспособности кластера:

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES                  AGE   VERSION
k3d-mycluster-server-0 Ready    control-plane,master   1m    v1.23.6+k3s1
k3d-mycluster-agent-0  Ready    <none>                 1m    v1.23.6+k3s1
k3d-mycluster-agent-1  Ready    <none>                 1m    v1.23.6+k3s1
```

Вывод подтвердил наличие трех узлов: одного управляющего (control-plane) и двух рабочих (agents).

Разработка и контейнеризация приложения

Был создан простой HTTP-сервер на Flask с двумя эндпоинтами:

- / — возвращает случайное число от 1 до 100.
- /health — возвращает "OK" для проверки состояния.

Код приложения (app.py):

```
from flask import Flask
import random

app = Flask(__name__)

@app.route('/')
def random_number():
    return f"Random number: {random.randint(1, 100)}"

@app.route('/health')
def health_check():
    return "OK", 200

if __name__ == '__main__':
       app.run(host='0.0.0.0', port=8000)
```

Файл зависимостей (requirements.txt):

```
Flask==2.3.2
```

Dockerfile для контейнеризации:

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 8000
CMD ["python", "app.py"]
```

Сборка Docker-образа:

```
docker build -t my-app:latest .
```

Импорт образа в кластер k3d:

```
k3d image import my-app --cluster mycluster
```

Развертывание приложения в Kubernetes

1. **Создание Deployment**

Для управления подами приложения был создан манифест Deployment с тремя репликами, включая Readiness и Liveness Probes:

Файл deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: my-app:latest
        imagePullPolicy: Never
        ports:
        - containerPort: 8000
        livenessProbe:
```

```
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 10
          periodSeconds: 5
        readinessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 5
             periodSeconds: 3
```

2. Создание Service

Для обращения к приложению внутри кластера был настроен Service:

Файл service.yaml:

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

3. Настройка Ingress

Для внешнего доступа был настроен Ingress с использованием Traefik (по умолчанию в k3d):

Файл ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-app-ingress
  annotations:
    traefik.ingress.kubernetes.io/router.entrypoints: web
spec:
  ingressClassName: traefik
  rules:
  - host: myapp.local
    http:
      paths:
      - path: /
        pathType: Prefix
```

```
        backend:
          service:
            name: my-app-service
            port:
              number: 80
```

Применение манифестов:

```
kubectl apply -f deployment.yaml,service.yaml,ingress.yaml
```

Проверка состояния:

```
kubectl get pods,svc,ingress

NAME                            READY   STATUS    RESTARTS   AGE
pod/my-app-5d5f8f8f6-abcde      1/1     Running   0          2m
pod/my-app-5d5f8f8f6-fghij      1/1     Running   0          2m
pod/my-app-5d5f8f8f6-klmno      1/1     Running   0          2m

NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/my-app-service   ClusterIP   10.43.123.45    <none>        80/TCP     2m

NAME                     CLASS    HOSTS          ADDRESS          PORTS   AGE
ingress/my-app-ingress   traefik  myapp.local    192.168.1.100    80      2m
```

Тестирование приложения (перед этим **myapp.local** был добавлен в **/etc/hosts**):

```
curl http://myapp.local
Random number: 42


curl http://myapp.local/health
OK
```

Возникшие проблемы и их устранение

1. **Поды в состоянии CrashLoopBackOff**

- Причина: Отсутствие эндпоинта /health, из-за чего Liveness и Readiness Probes завершались с ошибкой.
- Решение: Добавлен эндпоинт /health в app.py, после чего поды запустились корректно.

2. **Ошибка 404 при обращении через Ingress**

- Причина: Изначально в ingress.yaml был указан ingressClassName: nginx, тогда как k3d использует Traefik.

- Решение: Исправлен ingressClassName на traefik, и проверена настройка проброса портов в k3d.

Управление приложением с помощью Helm

Создание Helm-чарта

Для упрощения управления приложением был создан Helm-чарт:

Создание базового чарта:

```
helm create my-app-chart
rm -rf my-app-chart/templates/*
```

Файл values.yaml:

```
replicaCount: 3
image:
  repository: my-app
  tag: latest
service:
  port: 80
  targetPort: 8000
ingress:
  enabled: true
  host: myapp.local
```

Шаблоны (deployment.yaml, service.yaml, ingress.yaml) были адаптированы для использования значений из values.yaml, в остальном остались такими же, как в пункте “**Развертывание приложения в Kubernetes**”.

Установка приложения через Helm

Установка чарта:

```
helm install my-app ./my-app-chart -f my-app-chart/values.yaml
```

Проверка:

```
helm list

NAME    NAMESPACE       REVISION        UPDATED                                 STATUS
CHART           APP VERSION
my-app  default         1               2023-10-10 12:00:00.123456789 +0000 UTC
deployed        my-app-chart-0.1.0  1.0


kubectl get pods -l app=my-app

NAME                          READY   STATUS    RESTARTS   AGE
```

```
my-app-my-app-5d5f8f8f6-abcde   1/1     Running   0          5m
my-app-my-app-5d5f8f8f6-fghij   1/1     Running   0          5m
my-app-my-app-5d5f8f8f6-klmno   1/1     Running   0          5m
```

Масштабирование приложения

Для демонстрации масштабируемости количество реплик было увеличено с 3 до 5:

Обновление replicaCount в values.yaml:

```
replicaCount: 5
```

Обновление релиза Helm:

```
helm upgrade my-app ./my-app-chart -f my-app-chart/values.yaml

kubectl get pods

NAME                                READY   STATUS    RESTARTS   AGE
ingress-nginx-controller-7657f6db5f-h7q2j   1/1     Running       0          3d8h
my-app-my-app-5d5f8f8f6-abcde   1/1     Running   0          10m
my-app-my-app-5d5f8f8f6-fghij   1/1     Running   0          10m
my-app-my-app-5d5f8f8f6-klmno   1/1     Running   0          10m
my-app-my-app-5d5f8f8f6-pqrst   1/1     Running   0          1m
my-app-my-app-5d5f8f8f6-uvwxy   1/1     Running   0          1m
```

Повторное тестирование:

```
curl http://myapp.local
```

Вывод:

```
Random number: 87
```

Итоговое состояние

- Кластер k3d с двумя рабочими узлами.
- Приложение развернуто с пятью репликами.
- Доступ через http://myapp.local с использованием Traefik как Ingress-контроллера.
- Управление через Helm.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной РГР был успешно настроен Kubernetes-кластер с использованием k3d, разработано и развернуто приложение с помощью Deployment, Service и Ingress, а также перевел управление на Helm, продемонстрировав масштабирование. Задание позволило освоить ключевые аспекты работы с Kubernetes, включая:

- Создание и настройку локального кластера.
- Разработку и контейнеризацию приложения с учетом требований Kubernetes (например, health checks).
- Использование сущностей Kubernetes (Deployment, Service, Ingress) для управления приложением.
- Устранение типичных проблем (CrashLoopBackOff, ошибки Ingress).
- Применение Helm для упрощения управления и масштабирования.

Работа подтвердила работоспособность всех компонентов: приложение стабильно отвечало на запросы, масштабирование прошло успешно, а кластер функционировал корректно. Этот опыт заложил основу для дальнейшего изучения технологий виртуализации и оркестрации контейнеров, включая более сложные сценарии и лучшие практики.