

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

**ДИСЦИПЛИНА:**

Интеграция и развертывание программного обеспечения с помощью  
контейнеров

**Лабораторная работа №3-2**

**Тема:**

«Развертывание приложения в Kubernetes»

Выполнил(а): Морозова Валерия АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

**Цель работы:** освоить процесс развертывания приложения в Kubernetes с использованием Deployments и Services.

**Задачи:**

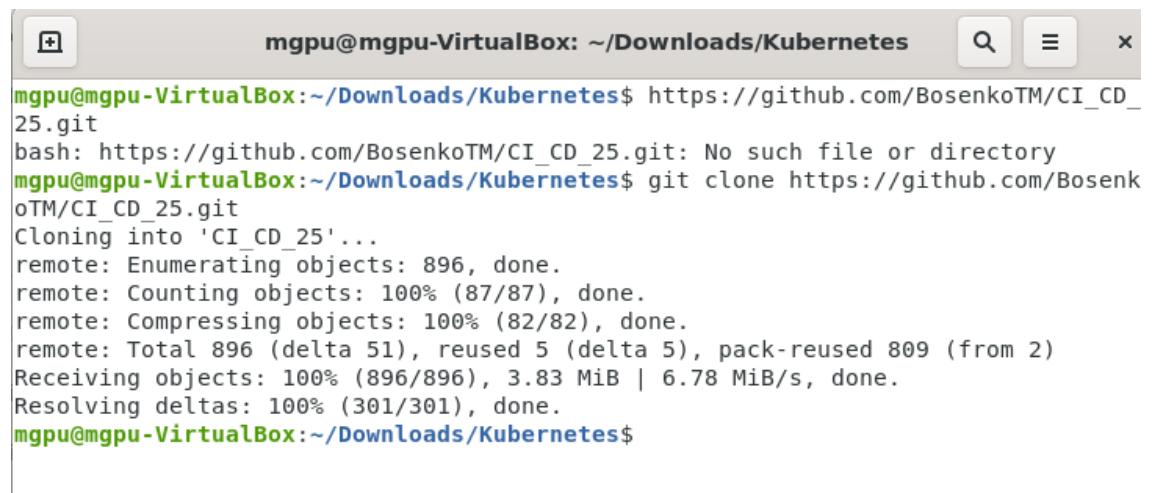
Создать Deployment для указанного приложения.

Создать Service для обеспечения доступа к приложению.

Проверить доступность приложения через созданный Service.

Выполнить индивидуальное задание.

**Вариант 10.** Разверните приложение на **Node.js**, использующее базу данных **MongoDB** и **Nginx** в качестве обратного прокси, в **Kubernetes**. Создайте **Deployment** для **Node.js**, **MongoDB** и **Nginx**, а также **Service** для доступа к приложению.



```
mgpu@mgpu-VirtualBox: ~/Downloads/Kubernetes
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes$ https://github.com/BosenkoTM/CI_CD_25.git
bash: https://github.com/BosenkoTM/CI_CD_25.git: No such file or directory
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes$ 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 | 6.78 MiB/s, done.
Resolving deltas: 100% (301/301), done.
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes$
```

Рисунок 1. Клонирование каталога

```
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
100 119M 100 119M 0 0 12.6M 0 0:00:09 0:00:09 --:--:-- 11.8M
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
[sudo] password for mgpu:
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$
```

Рисунок 2. Установка minikube

```
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$ sudo usermod -aG docker $USER && newgrp docker
```

Рисунок 3. Добавление пользователя в группу Docker

```
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$ sudo snap install kubectl --classic
kubectl 1.32.3 from Canonical✓ installed
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$
```

Рисунок 4. Установка kubectl

```

mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/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% 6.48 Mi
> gcr.io/k8s-minikube/kicbase...: 500.31 MiB / 500.31 MiB 100.00% 7.08 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: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
mgpu@mgpu-VirtualBox:~/Downloads/Kubernetes/CI_CD_25/practice/lab4_1$

```

Рисунок 5. Запуск

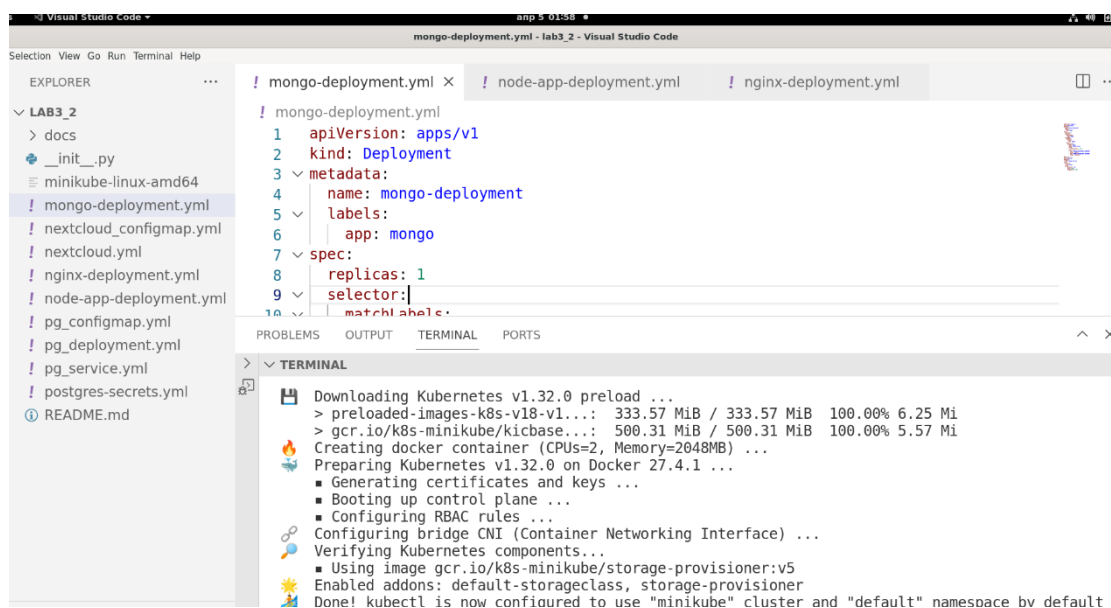


Рисунок 6. Создание трех деплоев для индивидуального задания

```

mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl create -f mongo-deployment.yml
deployment.apps/mongo-deployment created
service/mongo-service created
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl create -f node-app-deployment.yml
deployment.apps/node-app-deployment created
service/node-app-service created
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl create -f nginx-deployment.yml
deployment.apps/nginx-deployment created
configmap/nginx-config created

```

Рисунок 7. Создание ресурсов в Kubernetes кластере

```

mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-77d6b7c57d-9mt49   1/1     Running   0           59m
nginx-deployment-766c76446-mpb64    1/1     Running   0           55m
node-app-deployment-58d6b67b67-lkhfc 1/1     Running   0          5m44s
node-app-deployment-58d6b67b67-nf6vk 1/1     Running   0           5m8s

```

Рисунок 8. Состояние Pod в кластере Kubernetes

Как видно, все запустились.

```
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl get service
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes           ClusterIP      10.96.0.1        <none>            443/TCP           36m
mongo-service        ClusterIP      10.103.41.200    <none>            27017/TCP         12m
nginx-service        LoadBalancer  10.106.144.210   <pending>         80:30935/TCP      8m40s
node-app-service     ClusterIP      10.111.168.33    <none>            3000/TCP          11m
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$
```

Рисунок 9. Проверка сервисов

```
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mongo-deployment    1/1     1             1           60m
nginx-deployment    1/1     1             1           56m
node-app-deployment 2/2     2             2           59m
```

Рисунок 10. Проверка деплоев

```
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ kubectl describe pod mongo-deployment-77d6b7c57d-9mt49
Name:                mongo-deployment-77d6b7c57d-9mt49
Namespace:           default
Priority:              0
Service Account:     default
Node:                minikube/192.168.49.2
Start Time:          Sat, 05 Apr 2025 02:03:34 +0300
Labels:              app=mongo
                    pod-template-hash=77d6b7c57d
Annotations:         <none>
Status:              Running
IP:                  10.244.0.3
IPs:                 10.244.0.3
```

Рисунок 11. Просмотр первого пода Mongo

```
mgpu@mgpu-VirtualBox:~/Downloads/CI_CD_25/practice/lab3_2$ minikube service nginx-service --url
http://192.168.49.2:30935
```

Рисунок 12. OpenAPI

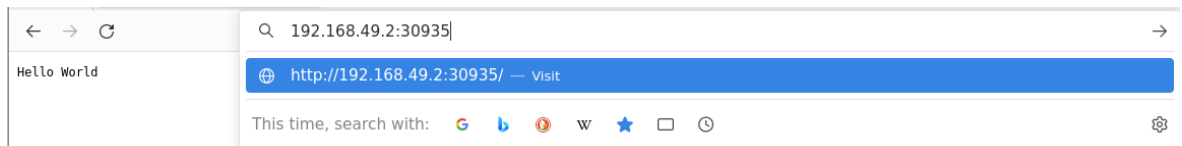


Рисунок 13. Проверка подключения проведена успешно

### Выводы:

1. Созданы 3 Deployment для указанного приложения.
2. Созданы Service для обеспечения доступа к приложению.
3. Проверена доступность приложения через созданный Service.
4. Выполнено индивидуальное задание.

### Контрольные вопросы.

**Kubernetes** — это платформа для оркестрации контейнеров в кластере. Она автоматически запускает приложения, регулирует число экземпляров и следит за их работоспособностью.

## 1. Что такое Pod, Deployment и Service в Kubernetes?

**Pod** — это минимальная единица работы в Kubernetes. Это как «оболочка» для одного или нескольких контейнеров, которые должны работать вместе. Например, приложение и его локальная база данных можно объединить в один Pod.

**Deployment** — это объект, который управляет запуском и обновлением Pod'ов. С его помощью можно указать, сколько экземпляров контейнера должно работать, и задать стратегию обновления (например, постепенное обновление или откат).

**Service** — это абстракция, которая объединяет Pod'ы в логическую группу и определяет политику доступа к ним. То есть что-то вроде маршрутизатора и балансировщика нагрузки между Pod'ами.

## 2. Каково назначение Deployment в Kubernetes?

Назначение Deployment в Kubernetes — управление жизненным циклом подов, обеспечение стабильного развёртывания, обновления и масштабирования приложений.

Некоторые функции Deployment:

**Обеспечение заданного количества подов.** Deployment следит за тем, чтобы всегда было запущено определённое количество подов, которое указано в конфигурации.

**Обновление версий приложений.** Deployment позволяет без простоя обновлять контейнеры до новых версий.

**Масштабирование.** С помощью Deployment можно легко увеличить или уменьшить количество запущенных подов в зависимости от нагрузки.

**Откаты (rollback).** В случае ошибки Deployment позволяет быстро вернуться к предыдущей версии приложения.

**Самовосстановление при сбое подов.** В таком случае Deployment автоматически запускает новые реплики, чтобы поддерживать необходимое количество работающих экземпляров.

## 3. Каково назначение Service в Kubernetes?

Назначение Service в Kubernetes — определение сетевого доступа к набору Pod-ов.

Поскольку Pod-ы в Kubernetes являются временными (они могут создаваться и удаляться при масштабировании или обновлении), обращаться к ним напрямую через IP-адреса неудобно и ненадёжно. Сервис решает эту проблему, создавая устойчивую точку доступа (имя + IP) для взаимодействия с Pod-ами. Он автоматически направляет трафик на актуальные живые Pod-ы, входящие в его группу.

Также сервис содержит политики доступа и отвечает за соблюдение этих политик для входящих запросов. Например, он может соединить внешний интерфейс приложения с внутренним, каждый из которых запускается в отдельном развёртывании внутри кластера.

#### 4. Как создать Deployment в Kubernetes?

Создание Deployment в Kubernetes происходит с помощью манифестов в формате YAML. **Манифесты** — это файлы, которые содержат описание ресурсов Kubernetes, таких как деплойменты, сервисы, конфигурационные карты и секреты.

Чтобы создать Deployment, нужно выполнить следующие шаги:

**Создать файл с манифестом.** В нём должны быть такие поля, как apiVersion, kind, metadata, spec, replicas, selector, template, metadata, spec, containers, name, image, ports.

**Применить манифест к кластеру Kubernetes.** Для этого используется команда kubectl. Нужно убедиться, что настроено окружение Kubernetes (локальный кластер или удалённый кластер). Затем выполнить команду

```
kubectl apply -f deployment.yaml
```

Она создаст деплоймент на основе файла манифеста, и Kubernetes начнёт создание подов для приложения.

**Проверить статус деплоймента.** Для этого используется команда

```
kubectl get deployments
```

Появится список всех деплоиментов в кластере и количество запущенных подов. Также можно проверить статус подов с помощью команды

```
kubectl get pods.
```

**Масштабировать деплоймент.** Чтобы изменить количество подов, можно отредактировать манифест YAML и изменить значение поля replicas.

## 5. Как создать Service в Kubernetes и какие типы Services существуют?

Чтобы создать Service в Kubernetes, нужно выполнить следующие шаги:

- Создать развёртывание. Для этого используют файл YAML или команду kubectl.
- Создать сервис. Сервис определяют в отдельном файле YAML или с помощью команды kubectl.
- Применить сервис. Для создания используют команду kubectl.
- Проверить статус сервиса. Для этого используют kubectl.