МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

СЕРГИЕНКО ЛЕВ ЭДУАРДОВИЧ

Отчет по
ЛАБОРАТОРНАЯ РАБОТА 4
ПО ДИСЦИПЛИНЕ
«Непрерывное интегрирование и сборка программного обеспечения»

Методы и средства оркестрации в многоконтейнерных системах

Преподаватель

Давидовская М.И.
Филиппов М.А.

2025

1. Цель работы.

Изучение создание кластера и работу с K8S.

2. Вариант задания.

3. Код приложений, конфигурационных файлов.

Задание 1. Подготовка окружения для локального кластера

```
PS C:\WINDOWS\system32> kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS       AGE
coredns-66bc5c9577-wbr8c           1/1     Running   0              6m6s
etcd-minikube                      1/1     Running   0              6m10s
kube-apiserver-minikube            1/1     Running   0              6m10s
kube-controller-manager-minikube   1/1     Running   0              6m10s
kube-proxy-cfj8h                   1/1     Running   0              6m6s
kube-scheduler-minikube            1/1     Running   0              6m10s
storage-provisioner                1/1     Running   1 (6m3s ago)   6m10s
PS C:\WINDOWS\system32> kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   6m50s   v1.34.0
PS C:\WINDOWS\system32> kubectl explain deployment
GROUP:      apps
KIND:       Deployment
VERSION:    v1

DESCRIPTION:
    Deployment enables declarative updates for Pods and ReplicaSets.

FIELDS:
  apiVersion    <string>
    APIVersion defines the versioned schema of this representation of an object.
    Servers should convert recognized schemas to the latest internal value, and
    may reject unrecognized values. More info:
    https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
```

Локальный кластер с использованием minikube

Задание 2. Знакомство с ресурсами kubernetes

Изучить список pods в kubernetes. Объяснить, зачем нужны эти компоненты и описать в отчете

```
PS C:\Users\lev> kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-66bc5c9577-wbr8c           1/1     Running   1 (26m ago)   34m
kube-system   etcd-minikube                      1/1     Running   1 (26m ago)   34m
kube-system   kube-apiserver-minikube            1/1     Running   1 (26m ago)   34m
kube-system   kube-controller-manager-minikube   1/1     Running   1 (26m ago)   34m
kube-system   kube-proxy-cfj8h                   1/1     Running   1 (26m ago)   34m
kube-system   kube-scheduler-minikube            1/1     Running   1 (26m ago)   34m
kube-system   storage-provisioner                1/1     Running   3 (65s ago)   34m
PS C:\Users\lev>
```

Создать том и продемонстрировать его монтирование в pod

```
kubectl apply -f volume-claim.yaml
persistentvolumeclaim/my-data-claim created
kubectl get pvc
NAME            STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
my-data-claim   Bound    pvc-c6eb21b2-0951-4cf4-9dfd-a2b7185ac59e   100Mi      RWO            standard       <unset>                 0s
kubectl apply -f pod-with-vol.yaml
pod/web-pod created
```

```
NAME       READY   STATUS    RESTARTS   AGE
web-pod    1/1     Running   0          61s
```

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2> kubectl exec -it pod/web-pod -- /bin/sh
/ # cd /data
/data # echo "Hello world!" > sample.txt
/data # exit
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2> kubectl delete pod/web-pod
pod "web-pod" deleted
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2> kubectl apply -f pod-with-vol.yaml
pod/web-pod created
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2> kubectl exec -it pod/web-pod -- /bin/sh -c "cat /data/sample.txt"
Hello world!
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2>
```

Продемонстрировать работу с ReplicaSet, коллекцией pods

```
kubectl apply -f replicaset.yaml
replicaset.apps/rs-web created
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2\rs> make info
kubectl get rs
NAME     DESIRED   CURRENT   READY   AGE
rs-web   3         3         3       5s
kubectl get pods -l app=web
NAME           READY   STATUS    RESTARTS   AGE
rs-web-4tpsq   1/1     Running   0          5s
rs-web-cfd7b   1/1     Running   0          5s
web-pod        1/1     Running   0          5m22s
kubectl describe rs/rs-web
Name:         rs-web
Namespace:    default
Selector:     app=web
Labels:       <none>
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=web
  Containers:
   nginx:
    Image:         nginx:alpine
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  6s    replicaset-controller  Created pod: rs-web-cfd7b
  Normal  SuccessfulCreate  6s    replicaset-controller  Created pod: rs-web-4tpsq
```

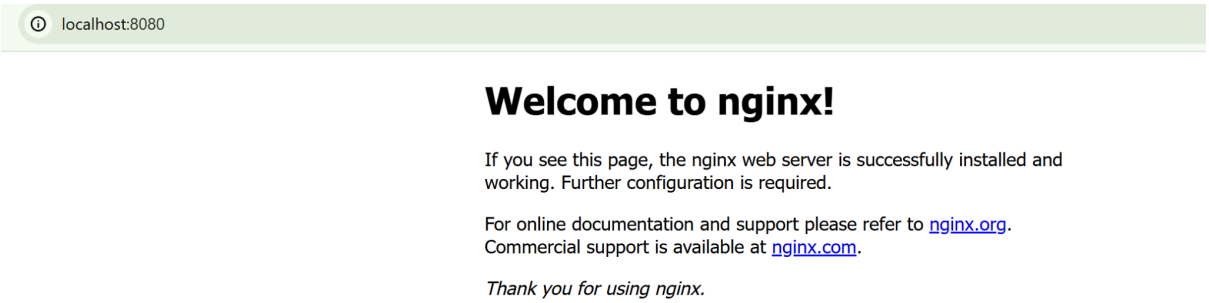
```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2\rs> kubectl delete pod rs-web-4tpsq
pod "rs-web-4tpsq" deleted
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task2\rs> make info
kubectl get rs
NAME     DESIRED   CURRENT   READY   AGE
rs-web   3         3         3       104s
kubectl get pods -l app=web
NAME           READY   STATUS    RESTARTS   AGE
rs-web-cfd7b   1/1     Running   0          104s
rs-web-txl6t   1/1     Running   0          3s
web-pod        1/1     Running   0          7m1s
kubectl describe rs/rs-web
Name:         rs-web
Namespace:    default
Selector:     app=web
Labels:       <none>
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=web
  Containers:
   nginx:
    Image:         nginx:alpine
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  104s  replicaset-controller  Created pod: rs-web-cfd7b
  Normal  SuccessfulCreate  104s  replicaset-controller  Created pod: rs-web-4tpsq
  Normal  SuccessfulCreate  3s    replicaset-controller  Created pod: rs-web-txl6t
```

Демонстрация восстановления пода при удалении

Задание 3. Работа с Kubernetes

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task3> kubectl apply -f web-deployment.yaml
>>     kubectl apply -f web-service.yaml
deployment.apps/web unchanged
service/web unchanged
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task3> kubectl get pods -l app=web
>>     kubectl describe deploy web
NAME                   READY   STATUS    RESTARTS   AGE
rs-web-cfd7b           1/1     Running   0          25m
rs-web-txl6t           1/1     Running   0          23m
web-75447dbc55-46mdp   1/1     Running   0          12m
web-75447dbc55-kf5ww   1/1     Running   0          12m
web-75447dbc55-m55pt   1/1     Running   0          12m
web-pod                1/1     Running   0          30m
Name:                   web
Namespace:              default
CreationTimestamp:      Sat, 04 Oct 2025 19:59:21 +0300
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=web
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=web
  Containers:
   nginx:
    Image:         nginx:alpine
    Port:          80/TCP
    Host Port:     0/TCP
```

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab4-foreverNP\task3> kubectl port-forward svc/web 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```



Демонстрация рабочего сервиса

4. Ответы на контрольные вопросы.

1. Что такое Kubernetes и зачем он нужен?

Kubernetes - это система оркестрации контейнеров. Нужен для автоматизации развёртывания, масштабирования и управления контейнерными приложениями в кластере.

2. Основные компоненты Kubernetes и как посмотреть их в kube-system

Ключевые компоненты:

- **kube-apiserver** - API сервер (входная точка).
- **etcd** - KV-хранилище кластера.
- **kube-controller-manager** - контроллеры (реплика, ноды и т.д.).
- **kube-scheduler** - планировщик подов на ноды.
- **kubelet** - агент на каждой ноде (поддерживает состояние подов).
- **kube-proxy** - сетевой прокси, реализует Service networking.
- **CoreDNS / kube-dns** -DNS для сервисов.

Команда, чтобы посмотреть поды в namespace kube-system:

kubectl get pods -n kube-system

```
PS C:\WINDOWS\system32> kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS        AGE
coredns-66bc5c9577-wbr8c           1/1     Running   0               4m19s
etcd-minikube                      1/1     Running   0               4m23s
kube-apiserver-minikube            1/1     Running   0               4m23s
kube-controller-manager-minikube   1/1     Running   0               4m23s
kube-proxy-cfj8h                   1/1     Running   0               4m19s
kube-scheduler-minikube            1/1     Running   0               4m23s
storage-provisioner                1/1     Running   1 (4m16s ago)   4m23s
PS C:\WINDOWS\system32>
```

3. Создание Pod из YAML -основные поля

```yaml
apiVersion: v1

kind: Pod

metadata:

  name: my-pod

  labels:

    app: demo

spec:

  containers:
```

```
- name: app
  image: nginx:stable
  ports:
  - containerPort: 80
restartPolicy: Always
```

4. Сервисы ClusterIP и Headless

- **ClusterIP** -внутренний виртуальный IP внутри кластера; обеспечивает балансировку (через kube-proxy) и DNS запись на имя сервиса. Применяется для большинства внутренних сервисов.
- **Headless Service** (clusterIP: None) -*без* виртуального IP; DNS возвращает записи A для каждого Endpoint (подходит для stateful-приложений, собственных балансировщиков на клиенте, прямых подключений). Используется вместе со StatefulSet или для сервис-дискавери.

5. Сервисы NodePort и LoadBalancer

- **NodePort** -открывает порт на каждой ноде (обычно 30000–32767). Внешний доступ через http://<NodeIP>:<NodePort>. NodePort фактически экспонирует ClusterIP сервис наружу.
- **LoadBalancer** - просит облачного провайдера создать внешний LB и привязывает его к ClusterIP/NodePort. На облаках автоматом выделяется публичный IP.

6. Ingress

Ingress - ресурс L7 маршрутизации, который организует host/path -> Service правила. Нужен Ingress Controller (nginx-ingress, Traefik, Istio Gateway и т.п.). Возможности: виртуальные хосты, path-based routing, TLS termination, аннотации для настроек.

7. Deployment, StatefulSet, DaemonSet

Deployment - для *стейтлес* приложений. Управляет ReplicaSet, поддерживает rolling updates, rollback, масштабирование.

StatefulSet -для приложений, которым нужны стабильные сетевые идентификаторы и постоянное хранилище (например, БД, Kafka). Обеспечивает порядковые создание/удаление, стабильные имена и PVC per pod.

DaemonSet -запускает один (или N) экземпляр пода на каждой ноде (или на выборочных нодах). Используется для логирования, мониторинга, сетевых агентов.

8. Job и CronJob

- **Job** - запускает один или несколько подов до успешного завершения (контролирует completions, parallelism). Хорошо для batch-задач.
- **CronJob** -планирует Jobs по cron-выражению; создаёт Jobs в заданное время/периодически.

9. ConfigMap и Secret

- **ConfigMap** - хранит неконфиденциальные конфиги (key-value, файлы). Использовать как переменные окружения (env / envFrom) или смонтировать как файл (volume).
- **Secret** -хранит чувствительные данные (пароли, токены). Значения base64-кодируются; рекомендуется включить шифрование etcd для защиты. Могут монтироваться как volume или использоваться в env.

10. Основные команды kubectl

```
# Просмотр ресурсов:

kubectl get pods

kubectl get all

kubectl get svc,deploy -n namespace

kubectl describe pod my-pod



# Создание/обновление/удаление:

kubectl apply -f resource.yaml
```

```
kubectl create -f resource.yaml

kubectl delete -f resource.yaml

kubectl edit deployment my-deploy

kubectl replace -f new.yaml



# Логи и интер:

kubectl logs pod-name

kubectl logs -f pod-name

kubectl exec -it pod-name -- /bin/sh

kubectl port-forward svc/my-svc 8080:80



# Масштабирование и rollout:

kubectl scale deployment/my-deploy --replicas=5

kubectl rollout status deployment/my-deploy

kubectl rollout undo deployment/my-deploy

# Конфигурация и контексты:

kubectl config get-contexts

kubectl config use-context my-cluster

# Диагностика и доп. утилиты:

kubectl top nodes

kubectl top pods

kubectl cp localfile pod:/path

kubectl port-forward pod/pod-name 8080:80

kubectl apply -k ./kustomization-dir

kubectl explain deployment
```