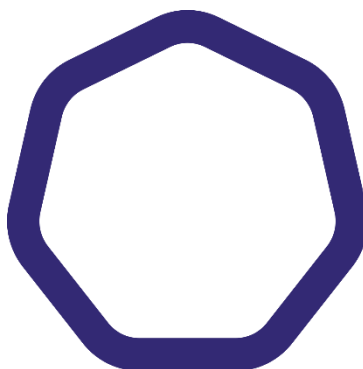


Министерство цифрового развития, связи и массовых коммуникаций  
Российской Федерации  
Ордена Трудового Красного Знамени федеральное государственное  
бюджетное образовательное учреждение высшего образования  
«Московский технический университет связи и информатики»  
(МТУСИ)



Кафедра: Сетевые информационные технологии и сервисы

Отчёт по лабораторной работе №7  
«Масштабирование приложения»

Выполнил:

Студенты 1 курса

Группы М092401(75)

Цыганков Р.О.

Проверил:

к.т.н., Шалагинов А. В.

---

## **Задание**

Научиться масштабировать приложения с помощью kubectl.

### **Выполнение работы**

Для выполнения лабораторных работ необходимо:

- Зайти на сайт: <https://kubernetes.io/ru/docs/tutorials/kubernetes-basics/scale/scale-intro/>;
- Изучить теоретическую часть;
- Нажать на кнопку «Начать интерактивный урок».

#### **Теоретическая часть работы:**

В случае масштабирования развёртывания создаются новые поды, которые распределяются по узлам с доступными ресурсами. Масштабирование увеличит количество подов в соответствии с указанным требуемым состоянием. Kubernetes также поддерживает автоматическое масштабирование подов. Кроме этого, возможно масштабирование до нуля, тогда завершается работа всех подов в развёртывании.

При запуске нескольких экземпляров приложения нужно правильно распределить трафик между ними. У сервисов есть встроенный балансировщик нагрузки, который распределяет сетевой трафик всех подов в открытом извне развёртывании. Сервисы постоянно отслеживают запущенные поды через их конечные точки, чтобы направлять трафик только на доступные поды.

Масштабирование выполняется с помощью изменения количества реплик в развёртывании.

Имея несколько работающих экземпляров приложения можно выполнять плавающие обновления (rolling updates) без простоев.

## Практическая часть работы:

### 1. Масштабирование деплоя:

< | Module 5 - Scale up your app

Step 1 of 3 ▶

Step 1: Scaling a deployment

To list your deployments use the `get deployments` command: `kubectl get deployments` ✓

The output should be similar to:

NAME	AVAILABLE	AGE	READY	UP-TO-DATE
kubernetes-bootcamp	1	11m	1/1	1

We should have 1 Pod. If not, run the command again. This shows:

- `NAME` lists the names of the Deployments in the cluster.
- `READY` shows the ratio of CURRENT/DESIRED

Terminal +

Selector: app=kubernetes-bootcamp  
Replicas: 4 desired | 4 updated | 4 total | 4 available | 0 unavailable  
StrategyType: RollingUpdate  
MinReadySeconds: 0  
RollingUpdateStrategy: 25% max unavailable, 25% max surge  
Pod Template:  
Labels: app=kubernetes-bootcamp  
Containers:  
kubernetes-bootcamp:  
Image: gcr.io/google-samples/kubernetes-bootcamp:v1  
Port: 8080/TCP  
Host Port: 0/TCP  
Environment: <none>  
Mounts: <none>  
Volumes: <none>  
Conditions:  
Type Status Reason  
-----  
Progressing True NewReplicaSetAvailable  
Available True MinimumReplicasAvailable  
OldReplicaSets: <none>  
NewReplicaSet: kubernetes-bootcamp-fb5c67579 (4/4 replicas created)  
Events:  
Type Reason Age From Message  
-----  
Normal ScalingReplicaSet 27s deployment-controller Scaled up replica set kubernetes-bootcamp-fb5c67579 to 1  
Normal ScalingReplicaSet 5s deployment-controller Scaled up replica set kubernetes-bootcamp-fb5c67579 to 4  
\$

### 2. Баланировка нагрузки:

< | Module 5 - Scale up your app

Step 2 of 3 ▶

Step 2: Load Balancing

Let's check that the Service is load-balancing the traffic. To find out the exposed IP and Port we can use the describe service as we learned in the previous Module:

`kubectl describe services/kubernetes-bootcamp` ✓

Create an environment variable called `NODE_PORT` that has a value as the Node port:

`export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')`  
`echo NODE_PORT=$NODE_PORT` ✓

Next, we'll do a `curl` to the exposed IP and port.

Terminal +

Events:  
Type Reason Age From Message  
-----  
Normal ScalingReplicaSet 27s deployment-controller Scaled up replica set kubernetes-bootcamp-fb5c67579 to 1  
Normal ScalingReplicaSet 5s deployment-controller Scaled up replica set kubernetes-bootcamp-fb5c67579 to 4  
\$ kubectl describe services/kubernetes-bootcamp  
Name: kubernetes-bootcamp  
Namespace: default  
Labels: app=kubernetes-bootcamp  
Annotations: <none>  
Selector: app=kubernetes-bootcamp  
Type: NodePort  
IP Families: <none>  
IP: 10.109.3.27  
IPs: 10.109.3.27  
Port: <unset> 8080/TCP  
TargetPort: 8080/TCP  
NodePort: <unset> 31634/TCP  
Endpoints: 172.18.0.2:8080,172.18.0.7:8080,172.18.0.8:8080 + 1 more...  
Session Affinity: None  
External Traffic Policy: Cluster  
Events: <none>  
\$ export NODE\_PORT=\$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')  
\$ echo NODE\_PORT=\$NODE\_PORT  
NODE\_PORT=31634  
\$ curl \$(minikube ip):\$NODE\_PORT  
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-9ckh8 | v=1  
\$

### 3. Уменьшение масштабируемости сервиса (Scale down):

< | Module 5 - Scale up your app

Step 3 of 3

Step 3: Scale Down

To scale down the Service to 2 replicas, run again the `scale` command:

```
kubectl scale deployments/kubernetes-bootcamp --replicas=2 ✓
```

List the Deployments to check if the change was applied with the `get deployments` command:

```
kubectl get deployments ✓
```

The number of replicas decreased to 2. List the number of Pods, with `get pods`:

```
kubectl get pods -o wide ✓
```

This confirms that 2 Pods were terminated.

Terminal

```
Port: <unset> 8080/TCP
TargetPort: 8080/TCP
NodePort: <unset> 31634/TCP
Endpoints: 172.18.0.2:8080,172.18.0.7:8080,172.18.0.8:8080 + 1 more...
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=31634
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-9ckh8 | v=1
$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME READY UP-TO-DATE AVAILABLE AGE
kubernetes-bootcamp 2/2 2 2 83s
$ kubectl get pods -o wide
NAME READY STATUS RESTARTS AGE IP
kubernetes-bootcamp-fb5c67579-9ckh8 1/1 Terminating 0 49s 172.18.0.7
inikube <none> <none>
kubernetes-bootcamp-fb5c67579-ftk6s 1/1 Running 0 49s 172.18.0.9
inikube <none> <none>
kubernetes-bootcamp-fb5c67579-msj7v 1/1 Running 0 70s 172.18.0.2
inikube <none> <none>
kubernetes-bootcamp-fb5c67579-v68rg 1/1 Terminating 0 49s 172.18.0.8
inikube <none> <none>
$
```

### Закключение

В результате работы мы получили практические навыки работы с масштабированием приложения с помощью `kubectl`.