



МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Интеллектуальные системы информационной безопасности»

Администрирование баз данных

Практическая работа № 5

ОТЧЁТ

Выполнил студент группы
БСБО-07-20
Любовский С.В.

Москва 2023

Выполнение задания.

1. Запуск кластера осуществился с помощью команды minikube start

< | Module 1 - 55:36

Create a Kubernetes cluster

Step 1 of 3

OK, we can see that minikube is in place.

Start the cluster, by running the `minikube start` command:

minikube start

Great! You now have a running Kubernetes cluster in your online terminal. Minikube started a virtual machine for you, and a Kubernetes cluster is now running in that VM.

CONTINUE

Terminal +

Kubernetes Bootcamp Terminal

\$ minikube version
minikube version: v1.18.0
commit: ec61815d60f66a6e4f6353030a40b12362557caa-dirty
\$ minikube start
* minikube v1.18.0 on Ubuntu 18.04 (amd64)
* Using the none driver based on existing profile

X The requested memory allocation of 2200MiB does not leave room for system overhead (total system memory: 2460MiB). You may face stability issues.
* Suggestion: Start minikube with less memory allocated: 'minikube start --memory=2200mb'

* Starting control plane node minikube in cluster minikube
* Running on localhost (CPUs=2, Memory=2460MB, Disk=194868MB)
...
* OS release is Ubuntu 18.04.5 LTS
* Preparing Kubernetes v1.20.2 on Docker 19.03.13 ...
- kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring local host environment ...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
\$

Команда `kubectl get nodes` показала все узлы, которые можно использовать для размещения наших приложений. Теперь у нас есть только одна нода, и мы видим, что ее статус готов.

< | Module 1 - 50:55

Create a Kubernetes cluster

Step 3 of 3

kubectl cluster-info

During this tutorial, we'll be focusing on the command line for deploying and exploring our application. To view the nodes in the cluster, run the `kubectl get nodes` command:

kubectl get nodes

This command shows all nodes that can be used to host our applications. Now we have only one node, and we can see that its status is ready (it is ready to accept applications for deployment).

CONTINUE

Terminal +

* Running on localhost (CPUs=2, Memory=2460MB, Disk=194868MB)
...
* OS release is Ubuntu 18.04.5 LTS
* Preparing Kubernetes v1.20.2 on Docker 19.03.13 ...
- kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring local host environment ...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
\$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.4", GitCommit:"e87da0bd6e03c3fea7933c4b5263d151aafd07c", GitTreeState:"clean", BuildDate:"2021-02-18T16:12:00Z", GoVersion:"go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.2", GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56", GitTreeState:"clean", BuildDate:"2021-01-13T13:20:00Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
\$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.8:8443
KubeDNS is running at https://10.0.0.8:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
\$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
minikube Ready control-plane,master 6m51s v1.20.2

2. Мы видим доступные узлы (в нашем случае 1). Kubernetes выберет, где развернуть приложение, исходя из доступных ресурсов Node.

Module 2 - 57:01

Deploy an app

Step 1 of 3

OK, kubectl is installed and you can see both the client and the server versions.

To view the nodes in the cluster, run the

`kubectl get nodes` command:

Here we see the available nodes (1 in our case). Kubernetes will choose where to deploy our application based on Node available resources.

CONTINUE

Terminal

```

an integer or a string.
--template='': Template string or path to template file
to use when -o=go-template, -o=go-template-file. The template
format is golang templates [http://golang.org/pkg/text/templat
e/#pkg-overview].
-w, --watch=false: After listing/getting the requested objec
t, watch for changes. Uninitialized objects are excluded if no
object name is provided.
--watch-only=false: Watch for changes to the requested o
bject(s), without listing/getting first.

Usage:
  kubectl get [(-o|--output=json|yaml|wide|custom-columns=...
|custom-columns-file=...|go-template=...|go-template-file=...|
jsonpath=...|jsonpath-file=...) (TYPE[.VERSION][.GROUP] [NAME
|-l label] | TYPE[.VERSION][.GROUP]/NAME ...) [flags] [option
s]

Use "kubectl options" for a list of global command-line option
s (applies to all commands).
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion
:"v1.20.4", GitCommit:"e87da0bd6e03ec3fea7933c4b5263d151aafd07
c", GitTreeState:"clean", BuildDate:"2021-02-18T16:12:00Z", Go
Version:"go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion
:"v1.20.2", GitCommit:"fae6196815e248d3ecfb03c680a4507229c2a5
6", GitTreeState:"clean", BuildDate:"2021-01-13T13:20:00Z", Go
Version:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
minikube      Ready     control-plane,master   68s   v1.20.2
$

```

С помощью команды `kubectl create deployment` мы развернули приложение. Мы видим, что существует 1 развертывание с одним экземпляром приложения. Экземпляр работает внутри контейнера Docker на нашем узле.

```

$ kubectl get deployments
NAME          READY    UP-TO-DATE    AVAILABLE    AGE
kubernetes-bootcamp  1/1      1              1            65s

```

Команда `kubectl` может создать прокси-сервер, который будет перенаправлять сообщения в частную сеть всего кластера.

```

$ echo -e "\n\n\n[e]Starting Proxy. After starting it will
e first Terminal Tab\n"; ease click the

Starting Proxy. After starting it will not output a response.
Please click the first Terminal Tab

$ kubectl proxy
Starting to serve on 127.0.0.1:8001

```

Теперь у нас есть соединение между нашим хостом и кластером Kubernetes. Мы можем запросить версию через API:

```
curl http://localhost:8001/version
```

Можно получить доступ к поду через API, запустив:

```
curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME/
```

```
$ curl http://localhost:8001/version
{
  "major": "1",
  "minor": "20",
  "gitVersion": "v1.20.2",
  "gitCommit": "faecb196815e248d3ecfb03c680a4507229c2a56",
  "gitTreeState": "clean",
  "buildDate": "2021-01-13T13:20:00Z",
  "goVersion": "gol.15.5",
  "compiler": "gc",
  "platform": "linux/amd64"
}$ export POD_NAME=$(kubectl get pods -o go-template --templat
'{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-57978f5f5d-wlwft
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$P
OD_NAME/
{
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "name": "kubernetes-bootcamp-57978f5f5d-wlwft",
    "generateName": "kubernetes-bootcamp-57978f5f5d-",
    "namespace": "default",
    "uid": "789abcc0-094b-4936-b6ae-4d73d90f117a",
    "resourceVersion": "716",
```

3. Чтобы найти существующие поды, запускает kubectl get pods

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTART
kubernetes-bootcamp-fb5c67579-bsvmf	1/1	Running	0

Далее просматриваем, какие контейнеры находятся внутри этого пода и какие image используются для создания этих контейнеров

```
$ kubectl describe pods
Name: kubernetes-bootcamp-fb5c67579-bsvmf
Namespace: default
Priority: 0
Node: minikube/10.0.0.6
Start Time: Tue, 04 Apr 2023 15:07:38 +0000
Labels: app=kubernetes-bootcamp
pod-template-hash=fb5c67579
Annotations: <none>
Status: Running
IP: 172.18.0.3
IPs:
  IP: 172.18.0.3
Controlled By: ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID: docker://alecf9c49187f5b0d1948122c0b40105e
b8e0768c2cc8302915c816c9ee38bd8
    Image: gcr.io/google-samples/kubernetes-bootcamp:
v1
    Image ID: docker-pullable://jocatalin/kubernetes-boo
tcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f
00e279c8fcc64af
    Port: 8080/TCP
    Host Port: 0/TCP
    State: Running
      Started: Tue, 04 Apr 2023 15:07:40 +0000
    Ready: True
    Restart Count: 0
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from defau
```

Запускаем прокси во втором терминале, после выводим POD_NAME и вывод приложения

```
$ export POD_NAME=$(kubectl get pods -o go-template --template
'{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-bsvmf
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME/proxy/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-bsvmf | v=1
```

Получение логов

```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2023-04-04T15:07:40.270Z
Running On: kubernetes-bootcamp-fb5c67579-bsvmf

Running On: kubernetes-bootcamp-fb5c67579-bsvmf | Total Requests: 1 | App Uptime: 411.163 seconds | Log Time: 2023-04-04T15:14:31.433Z
```

Выполнение команды непосредственно в контейнере после запуска пода

```
$ kubectl exec $POD_NAME -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=kubernetes-bootcamp-fb5c67579-knpm7
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=6.3.1
HOME=/root
```

Исходный код приложения находится в файле server.js

```
$ kubectl exec -ti $POD_NAME -- bash
root@kubernetes-bootcamp-fb5c67579-knpm7:/# cat server.js
var http = require('http');
var requests=0;
var podname= process.env.HOSTNAME;
var startTime;
var host;
var handleRequest = function(request, response) {
  response.setHeader('Content-Type', 'text/plain');
  response.writeHead(200);
  response.write("Hello Kubernetes bootcamp! | Running on: ");
  response.write(host);
  response.end(" | v=1\n");
  console.log("Running On:" ,host, "| Total Requests:", ++requests, "| App Uptime:", (new Date() - startTime)/1000 , "seconds", "| Log Time:",new Date());
}
var www = http.createServer(handleRequest);
www.listen(8080,function () {
  startTime = new Date();
  host = process.env.HOSTNAME;
  console.log ("Kubernetes Bootcamp App Started At:",startTime, "| Running On: " ,host, "\n" );
});
```

Проверить, запущено ли приложение

```
root@kubernetes-bootcamp-fb5c67579-knpm7:/# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-knpm7 | v=1
```

4. Сервисы кластера


```
$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)
AGE
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP
33s
```

Чтобы создать новый сервис и открыть его для внешнего трафика, используем команду expose с NodePort в качестве параметра

```
$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
service/kubernetes-bootcamp exposed
```

Теперь у нас есть сервис под названием kubernetes-bootcamp

```
$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)
AGE
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP
2m35s
kubernetes-bootcamp NodePort      10.100.27.113 <none>         8080:32132/TCP
85s
```

Чтобы узнать, какой порт был открыт, запустим команду describe

```
$ 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:            10.100.27.113
IPs:           10.100.27.113
Port:          <unset> 8080/TCP
TargetPort:    8080/TCP
NodePort:      <unset> 32132/TCP
Endpoints:     172.18.0.2:8080
Session Affinity: None
External Traffic Policy: Cluster
Events:        <none>
```

Создает переменную среды с именем NODE_PORT, которой присвоено значение порта узла

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=32132
```

Теперь мы можем проверить, доступно ли приложение за пределами кластера

```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-x6cq4 | v=1
```

Используем label, вызвав description, для запроса списка подов

```
$ kubectl get pods -l app=kubernetes-bootcamp
NAME          READY    STATUS    RESTARTS
AGE
kubernetes-bootcamp-fb5c67579-x6cq4 1/1      Running   0
6m45s
```

Получите имя пода и сохраните его в переменной POD_NAME

```
$ export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-x6cq4
```

Чтобы применить label

```
$ kubectl label pods $POD_NAME version=v1
pod/kubernetes-bootcamp-fb5c67579-x6cq4 labeled
```

Теперь можно запросить список подов, используя label

```
$ kubectl get pods -l version=v1
```

NAME	READY	STATUS	RESTARTS
kubernetes-bootcamp-fb5c67579-x6cq4	1/1	Running	0

Чтобы удалить сервер используется

```
$ kubectl delete service -l app=kubernetes-bootcamp
service "kubernetes-bootcamp" deleted
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP

Больше нельзя подключиться

```
$ curl $(minikube ip):$NODE_PORT
curl: (7) Failed to connect to 10.0.0.10 port 32132: Connection refused
```

Приложение все еще работает внутри пода

```
$ kubectl exec -ti $POD_NAME -- curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-x6cq4 | v=1
```

5. Список развернутых подов

```
$ kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kubernetes-bootcamp	1/1	1	1	44s

Чтобы увидеть ReplicaSet

```
$ kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
kubernetes-bootcamp-fb5c67579	1	1	1	11s

Масштабирование до 4 реплик

```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
```

```
$ kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kubernetes-bootcamp	4/4	4	4	4m32s

Теперь есть 4 пода с разными IP

```
$ kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS
kubernetes-bootcamp-fb5c67579-h97xz	1/1	Running	0
kubernetes-bootcamp-fb5c67579-htv5d	1/1	Running	0
kubernetes-bootcamp-fb5c67579-kz6v2	1/1	Running	0
kubernetes-bootcamp-fb5c67579-q5f69	1/1	Running	0

Находим открытые IP и порты

```
$ 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:        10.105.153.205
IPs:       10.105.153.205
Port:      <unset> 8080/TCP
TargetPort: 8080/TCP
NodePort:   <unset> 31666/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>
```

Создание переменной среды с именем NODE_PORT, которая имеет значение порта узла

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp
-o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=31666
```

С каждым запросом мы попадаем в разные поды. Это свидетельствует о том, что балансировка нагрузки работает

```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-kz6v2 | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-htv5d | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-h97xz | v=1
```

Уменьшение масштаба до 3 реплик

```
$ 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            15m
$ kubectl get pods -o wide
NAME                READY    STATUS    RESTARTS   TS    AGE    IP            NODE                NOMINATED NODE    READINESS GATES
kubernetes-bootcamp-fb5c67579-htv5d 1/1      Running    0           15m   15m    172.18.0.2    minikube            <none>             <none>
kubernetes-bootcamp-fb5c67579-q5f69 1/1      Running    0           12m   12m    172.18.0.9    minikube            <none>             <none>
```

6. Развернутые поды


```
$ kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kubernetes-bootcamp	4/4	4	4	6m30s

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
kubernetes-bootcamp-fb5c67579-2wks5	1/1	Running	0	6m23s
kubernetes-bootcamp-fb5c67579-69dck	1/1	Running	0	6m23s
kubernetes-bootcamp-fb5c67579-8jx26	1/1	Running	0	6m23s
kubernetes-bootcamp-fb5c67579-rrgfz	1/1	Running	0	6m23s

Обновление image

```
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
deployment.apps/kubernetes-bootcamp image updated
```

Статус новых подов и старых, завершающих работу

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
kubernetes-bootcamp-7d44784b7c-b429b	1/1	Running	0	33s
kubernetes-bootcamp-7d44784b7c-c9sz2	1/1	Running	0	33s
kubernetes-bootcamp-7d44784b7c-j5m8n	1/1	Running	0	35s
kubernetes-bootcamp-7d44784b7c-lhs8k	1/1	Running	0	35s
kubernetes-bootcamp-fb5c67579-2wks5	0/1	Terminating	0	10m
kubernetes-bootcamp-fb5c67579-69dck	0/1	Terminating	0	10m
kubernetes-bootcamp-fb5c67579-8jx26	1/1	Terminating	0	10m
kubernetes-bootcamp-fb5c67579-rrgfz	0/1	Terminating	0	10m

IP и ports

```
$ 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:                 10.103.75.241
IPs:                10.103.75.241
Port:               <unset> 8080/TCP
TargetPort:         8080/TCP
NodePort:           <unset> 30146/TCP
Endpoints:          172.18.0.10:8080,172.18.0.11:8080,172.18.0.12:8080 + 1 more...
Session Affinity:   None
External Traffic Policy: Cluster
Events:            <none>
```

Создание переменной среды с именем NODE_PORT, которой присвоено значение порта узла

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp
-o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=30146
```

Curl

```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7d44784b7c-c9sz2 | v=2
```

Каждый раз, когда запускается команда curl, мы попадаем в другой под. На всех подах установлена последняя версия (v2). Выполнив команду rollout status, подтверждаем обновление

```
$ kubectl rollout status deployments/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
```

Апдейт до 10 версии

```
$ kubectl set image deployments/kubernetes-bootcamp kubernetes
-bootcamp=gcr.io/google-samples/kubernetes-bootcamp:v10
deployment.apps/kubernetes-bootcamp image updated
```

Не все поды были загружены

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kubernetes-bootcamp	3/4	2	3	34m

Некоторые поды имеют статус ImagePullBackOff

```
$ kubectl get pods
NAME                                READY   STATUS
kubernetes-bootcamp-59b7598c77-cd8snf 0/1     ImagePullBackOf
kubernetes-bootcamp-59b7598c77-rxwm2f 0/1     ImagePullBackOf
kubernetes-bootcamp-7d44784b7c-c9sz2 1/1     Running
kubernetes-bootcamp-7d44784b7c-j5m8n 1/1     Running
kubernetes-bootcamp-7d44784b7c-lhs8k 1/1     Running
```

Версия образа v10 не существует в репозитории

```
Events:
Type      Reason      Age   From          Message
----      -
Normal    Scheduled   25m   default-scheduler Successfully assigned default/kubernetes-bootcamp-7d44784b7c-lhs8k to minikube
Normal    Pulled      25m   kubelet       Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
Normal    Created     25m   kubelet       Created container kubernetes-bootcamp
Normal    Started     25m   kubelet       Started container kubernetes-bootcamp
```

Откатываем изменения

```
$ kubectl rollout undo deployments/kubernetes-bootcamp
deployment.apps/kubernetes-bootcamp rolled back
```

Rollback прошел успешно

```
$ kubectl get pods
NAME                                READY   STATUS   RESTARTS   AGE
kubernetes-bootcamp-7d44784b7c-c9sz2 1/1     Running  0          28m
kubernetes-bootcamp-7d44784b7c-j5m8n 1/1     Running  0          28m
kubernetes-bootcamp-7d44784b7c-lhs8k 1/1     Running  0          28m
kubernetes-bootcamp-7d44784b7c-scstb 1/1     Running  0          97s
```