

Московский Физико-Технический Институт  
(государственный университет)

---

## Реферат на тему: Мой опыт работы с kubernetes

---

Сибгатуллин Булат, ФРКТ

# Содержание

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Что такое kubernetes?</b>                         | <b>3</b> |
| 1.1      | Концепции kubernetes . . . . .                       | 3        |
| 1.2      | Архитектура kubernetes . . . . .                     | 4        |
| 1.2.1    | Нода kubernetes . . . . .                            | 5        |
| 1.2.2    | Kubelet . . . . .                                    | 5        |
| 1.2.3    | Kube-Proxy . . . . .                                 | 5        |
| 1.2.4    | Компоненты управления Kubernetes . . . . .           | 5        |
| 1.2.5    | etcd . . . . .                                       | 5        |
| 1.2.6    | Kubernetes API Server . . . . .                      | 6        |
| 1.2.7    | Scheduler . . . . .                                  | 6        |
| 1.2.8    | Kubernetes Controller Manager Server . . . . .       | 6        |
| <b>2</b> | <b>Установка Minikube на локальную машину</b>        | <b>7</b> |
| 2.1      | Установка дополнительного ПО . . . . .               | 7        |
| 2.1.1    | Установка kubectl . . . . .                          | 7        |
| 2.1.2    | Установка Hypervisor . . . . .                       | 8        |
| 2.2      | Установка Minikube с помощью прямой ссылки . . . . . | 8        |
| <b>3</b> | <b>Работа с тестовой программой</b>                  | <b>8</b> |
| 3.1      | Подготовка к работе . . . . .                        | 8        |
| 3.2      | Создание кластера Minikube . . . . .                 | 9        |
| 3.3      | Создание Deployment . . . . .                        | 10       |

|     |                                 |    |
|-----|---------------------------------|----|
| 3.4 | Создание сервиса . . . . .      | 13 |
| 3.5 | Добавление дополнений . . . . . | 15 |
| 4   | Завершение работы               | 16 |
| 5   | Обсуждение результатов          | 16 |

# 1 Что такое kubernetes?

Kubernetes является проектом с открытым исходным кодом, предназначенным для управления кластером контейнеров Linux как единой системой. Kubernetes управляет и запускает контейнеры Docker на большом количестве хостов, а так же обеспечивает совместное размещение и репликацию большого количества контейнеров.

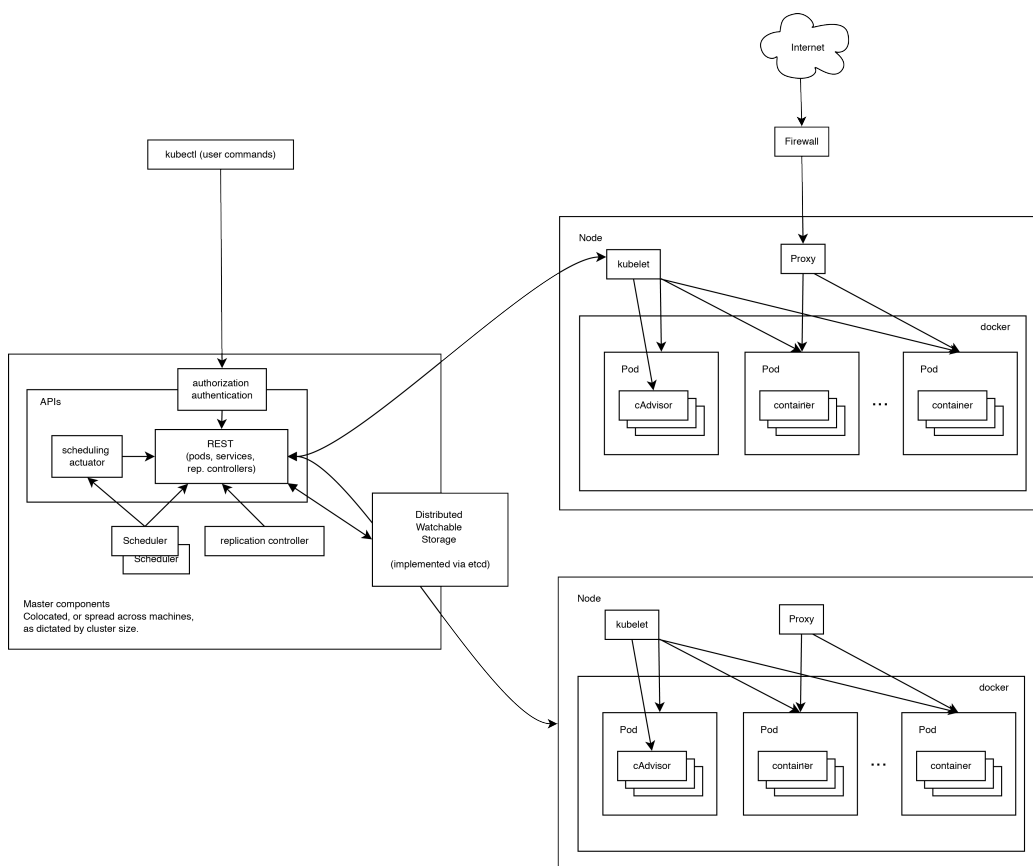
Проект преследует две цели. Если вы пользуетесь контейнерами Docker, возникает следующий вопрос о том, как масштабировать и запускать контейнеры сразу на большом количестве хостов Docker, а также как выполнять их балансировку. В проекте предлагается высокоуровневый API, определяющее логическое группирование контейнеров, позволяющее определять пулы контейнеров, балансировать нагрузку, а также задавать их размещение.

## 1.1 Концепции kubernetes

- Nodes ([nodes](#)): Нода это машина в кластере Kubernetes.
- Pods ([pods](#)): Pod это группа контейнеров с общими разделами, запускаемых как единое целое.
- Replication Controllers ([replication controller](#)): replication controller гарантирует, что определенное количество «реплик» pod'ы будут запущены в любой момент времени.
- Services ([service](#)): Сервис в Kubernetes это абстракция которая определяет логический объединённый набор pod и политику доступа к ним.
- Volumes ([volumes](#)): Volume(раздел) это директория, возможно, с данными в ней, которая доступна в контейнере.
- Labels ([labels and selectors](#)): Label'ы это пары ключ/значение которые прикрепляются к объектам, например pod'ам. Label'ы могут быть использованы для создания и выбора наборов объектов.
- Kubectl Command Line Interface ([kubectl](#)): kubectl интерфейс командной строки для управления Kubernetes.

## 1.2 Архитектура kubernetes

Работающий кластер Kubernetes включает в себя агента, запущенного на нодах (kubelet) и компоненты мастера (APIs, scheduler, etc), поверх решения с распределённым хранилищем. Приведённая схема показывает желаемое, в конечном итоге, состояние, хотя все ещё ведётся работа над некоторыми вещами, например: как сделать так, чтобы kubelet (все компоненты, на самом деле) самостоятельно запускался в контейнере, что сделает планировщик на 100% подключаемым.



Рассмотрим архитектуру кластера детально:

### 1.2.1 Нода `kubernetes`

При взгляде на архитектуру системы мы можем разбить его на сервисы, которые работают на каждой ноде и сервисы уровня управления кластера. На каждой ноде `Kubernetes` запускаются сервисы, необходимые для управления нодой со стороны мастера и для запуска приложений. Конечно, на каждой ноде запускается `Docker`. `Docker` обеспечивает загрузку образов и запуск контейнеров.

### 1.2.2 `Kubelet`

`Kubelet` управляет `pod`'ами их контейнерами, образами, разделами, etc.

### 1.2.3 `Kube-Proxy`

Также на каждой ноде запускается простой проху-балансировщик. Этот сервис запускается на каждой ноде и настраивается в `Kubernetes API`. `Kube-Proxy` может выполнять простейшее перенаправление потоков TCP и UDP (round robin) между набором бэкендов.

### 1.2.4 Компоненты управления `Kubernetes`

Система управления `Kubernetes` разделена на несколько компонентов. В данный момент все они запускаются на мастер-ноде, но в скором времени это будет изменено для возможности создания отказоустойчивого кластера. Эти компоненты работают вместе, чтобы обеспечить единое представление кластера.

### 1.2.5 `etcd`

Состояние мастера хранится в экземпляре `etcd`. Это обеспечивает надёжное хранение конфигурационных данных и своевременное оповещение прочих компонентов об изменении состояния.

### 1.2.6 Kubernetes API Server

Kubernetes API обеспечивает работу api-сервера. Он предназначен для того, чтобы быть CRUD сервером со встроенной бизнес-логикой, реализованной в отдельных компонентах или в плагинах. Он, в основном, обрабатывает REST операции, проверяя их и обновляя соответствующие объекты в etcd (и событийно в других хранилищах).

### 1.2.7 Scheduler

Scheduler привязывает незапущенные pod'ы к нодам через вызов /binding API. Scheduler подключаем; планируется поддержка множественных scheduler'ов и пользовательских scheduler'ов.

### 1.2.8 Kubernetes Controller Manager Server

Все остальные функции уровня кластера представлены в Controller Manager. Например, ноды обнаруживаются, управляются и контролируются средствами node controller. Эта сущность в итоге может быть разделена на отдельные компоненты, чтобы сделать их независимо подключаемыми.

ReplicationController — это механизм, основывающийся на pod API. В конечном счете планируется перевести её на общий механизм plug-in, когда он будет реализован.

## 2 Установка Minikube на локальную машину

Для вводного знакомства с Kubernetes установим [Minikube](#), инструмент для запуска одноузлового кластера Kubernetes на виртуальной машине в персональном компьютере.

### 2.1 Установка дополнительного ПО

Чтобы проверить, поддерживается ли виртуализация в Windows, выполним следующую команду:

```
systeminfo
```

Видим следующий вывод:

```
Hyper-V Requirements: VM Monitor Mode Extensions: Yes
                      Virtualization Enabled In Firmware: Yes
                      Second Level Address Translation: Yes
                      Data Execution Prevention Available: Yes
```

значит виртуализация поддерживается в Windows.

#### 2.1.1 Установка kubectl

Инструмент командной строки Kubernetes kubectl позволяет запускать команды для кластеров Kubernetes. Он нужен для запуска и работы с кластером в нашем проекте.

Установим его с этой ссылки [kubectl](#). Все установки будем производить в папку `C:\kubernetes`. Добавим эту папку в переменную Path в Windows.

Убедимся, что kubectl установлен с помощью:

```
kubectl version --client
```



### 2.1.2 Установка Hypervisor

Также Minikube требует установленный гипервизор. Его можно не устанавливать и просто запускать Minikube с опцией `--vm-driver=none`. Тогда компоненты Kubernetes будут запускаться на хосте, а не виртуальной машине. Но мне интересно опробовать Kubernetes именно так, как я бы использовал его в инфраструктуре компании (то есть на виртуальной машине).

Также учитывайте: драйвера виртуальной машины none может привести к проблемам безопасности и потери данных. Перед использованием `--vm-driver=none` обратитесь к [этой документации](#) для получения дополнительной информации.

В качестве гипервизора установим [Virtualbox](#). Выберем вариант для Windows и произведем установку с настройками по умолчанию.

## 2.2 Установка Minikube с помощью прямой ссылки

Перейдем по ссылке [minikube github](#) и выберем файл с названием `minikube-windows-amd64.exe`, сохраним его папке `C:\kubernetes` и переименуем в `minikube.exe`. Папка уже находится в переменной Path, поэтому просто проверим успешность установки написав в Windows PowerShell:

```
minikube version
```

## 3 Работа с тестовой программой

### 3.1 Подготовка к работе

Создадим директорию для работы, в ней напишем программу на JavaScript (`server.js`):

```
var http = require('http');
```

```
var handleRequest = function(request, response) {
  console.log('Получен запрос на URL: ' + request.url);
  response.writeHead(200);
  response.end('Hello World!');
};
var www = http.createServer(handleRequest);
www.listen(8080);
```

И образ контейнера (Dockerfile):

```
FROM node:6.14.2
EXPOSE 8080
COPY server.js .
CMD [ "node", "server.js" ]
```

## 3.2 Создание кластера Minikube

1) Создадим новую сессию в терминале и запустим там minikube при помощи команды:

```
minikube start
```

Потом откроем веб-панель Kubernetes в браузере:

```
minikube dashboard
```

В браузере откроется вот такая страница. В данный момент никакие задания не исполняются:

2) Также проверим состояниекомпонент kubernetes:

```
kubectl get componentstatuses
```

Получим информация о том, что все они в порядке:

| NAME  | STATUS | MESSAGE |
|-------|--------|---------|
| ERROR |        |         |

|                    |         |                                  |
|--------------------|---------|----------------------------------|
| controller-manager | Healthy | ok                               |
| scheduler          | Healthy | ok                               |
| etcd-0             | Healthy | {"health": "true", "reason": ""} |

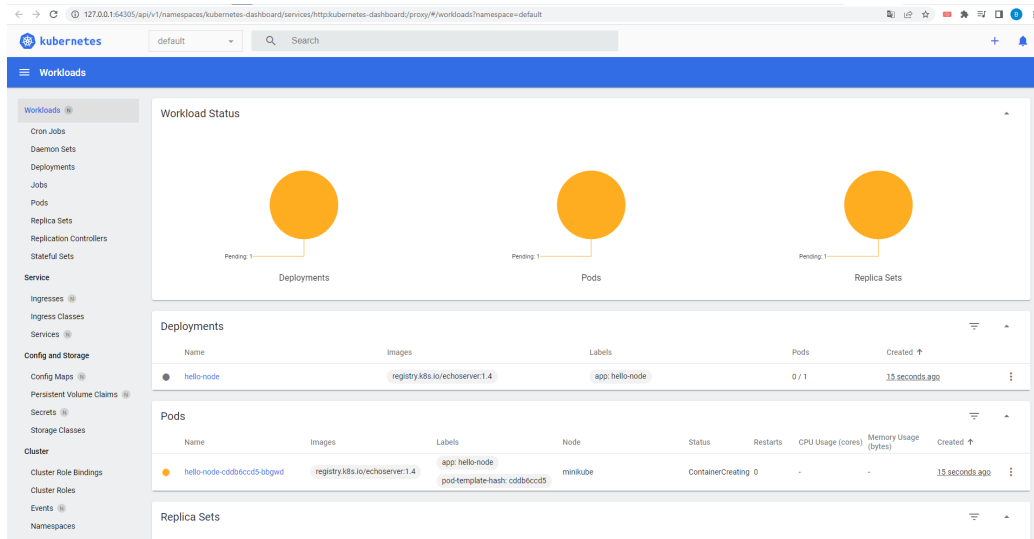
### 3.3 Создание Deployment

Kubernetes - это группа из одного или более контейнеров, связанных друг с другом с целью администрирования и организации сети. Deployment в Kubernetes проверяет здоровье пода и перезагружает контейнер пода в случае его отказа.

1) Используем команду `kubectl create` для создания деплоймента для управления подом. Под запускает контейнер на основе предоставленного Docker образа.

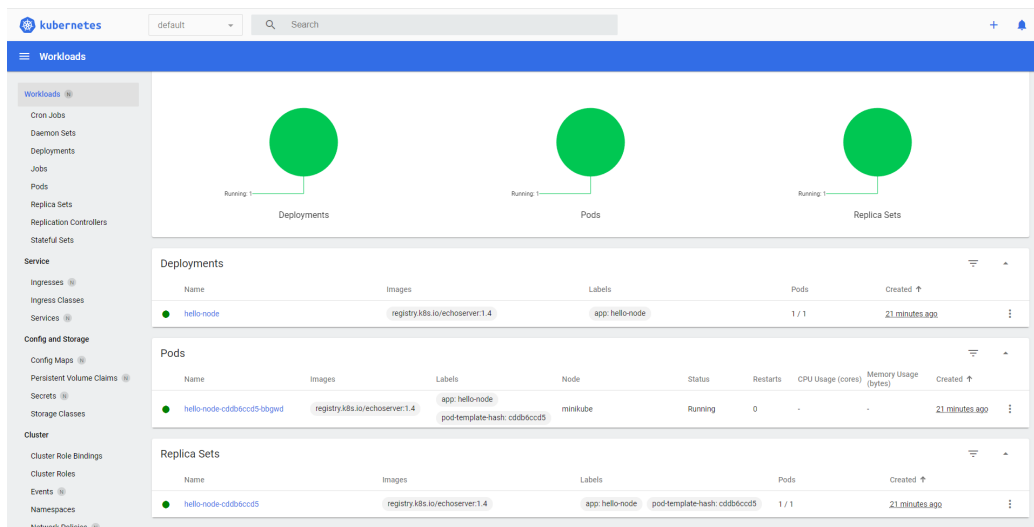
```
kubectl create deployment hello-node --image=registry.k8s.io/echoserver:1.4
```

Получим вот такой результат:



Статус подсвечен желтым, так как для данного Deployment-а еще не успел создаться контейнер. Вот какая картина будет когда Deployment получит статус running:

2) Теперь мы можем посмотреть информацию о Deployment:



`kubectl get deployments`

В ответ получим:

| NAME       | READY | UP-TO-DATE | AVAILABLE | AGE   |
|------------|-------|------------|-----------|-------|
| hello-node | 1/1   | 1          | 1         | 2m14s |

3) Также запросим информацию о поде:

`kubectl get pods`

В ответ получим:

| NAME                       | READY | STATUS  | RESTARTS | AGE |
|----------------------------|-------|---------|----------|-----|
| hello-node-cddb6ccd5-bbgwd | 1/1   | Running | 0        | 24m |

4) Посмотрим на события кластера:

`kubectl get events`

Вывод:

5) И наконец конфигурация кластера можно получить при помощи команды:

|     |        |                         |                                 |   |
|-----|--------|-------------------------|---------------------------------|---|
| 24m | Normal | Scheduled               | pod/hello-node-cddb6ccd5-bbgwd  | Successfully assigned default/hello-node-cddb6ccd5-bbgwd to minikube  |
| 24m | Normal | Pulling                 | pod/hello-node-cddb6ccd5-bbgwd  | Pulling image "registry.k8s.io/echoserver:1.4"  |
| 24m | Normal | Pulled                  | pod/hello-node-cddb6ccd5-bbgwd  | Successfully pulled image "registry.k8s.io/echoserver:1.4" in 18.940150124s (18.940161695s including waiting) |
| 24m | Normal | Created                 | pod/hello-node-cddb6ccd5-bbgwd  | Created container echoserver  |
| 24m | Normal | Started                 | pod/hello-node-cddb6ccd5-bbgwd  | Started container echoserver  |
| 24m | Normal | SuccessfulCreate        | replicaset/hello-node-cddb6ccd5 | Created pod: hello-node-cddb6ccd5-bbgwd   |
| 24m | Normal | ScalingReplicaSet       | deployment/hello-node           | Scaled up replica set hello-node-cddb6ccd5 to 1   |
| 28m | Normal | Starting                | node/minikube                   | Starting kubelet.   |
| 28m | Normal | NodeAllocatableEnforced | node/minikube                   | Updated node Allocatable limit across pods  |
| 28m | Normal | NodeHasSufficientMemory | node/minikube                   | Node minikube status is now: NodeHasSufficientMemory  |
| 28m | Normal | NodeHasNoDiskPressure   | node/minikube                   | Node minikube status is now: NodeHasNoDiskPressure  |
| 28m | Normal | NodeHasSufficientPID    | node/minikube                   | Node minikube status is now: NodeHasSufficientPID   |
| 28m | Normal | NodeReady               | node/minikube                   | Node minikube status is now: NodeReady  |
| 28m | Normal | RegisteredNode          | node/minikube                   | Node minikube event: Registered Node minikube in Controller   |
| 28m | Normal | Starting                | node/minikube                   |   |

kubectl config view

Получаем:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: C:\Users\GAMER\.minikube\ca.crt
    extensions:
    - extension:
        last-update: Thu, 25 May 2023 02:18:46 MSK
        provider: minikube.sigs.k8s.io
        version: v1.30.1
        name: cluster_info
        server: https://192.168.59.100:8443
    name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Thu, 25 May 2023 02:18:46 MSK
        provider: minikube.sigs.k8s.io
        version: v1.30.1
        name: context_info
    namespace: default
    user: minikube
    name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: C:\Users\GAMER\.minikube\profiles\minikube\client.crt
    client-key: C:\Users\GAMER\.minikube\profiles\minikube\client.key
```

Можем увидеть здесь ip-адрес кластера, момент последнего изменения, сертификаты и т.д.

Теперь, научившись создавать Deployment и смотреть на состояние кластера, можем переходить к следующему этапу.

## 3.4 Создание сервиса

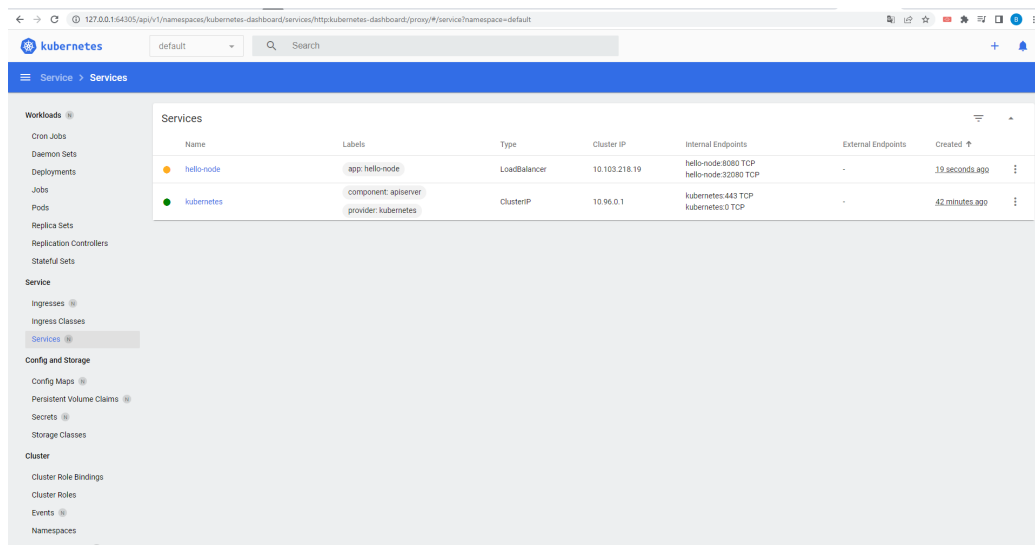
По-умолчанию под доступен только при обращении по его внутреннему IP адресу внутри кластера Kubernetes. Чтобы сделать контейнер hello-node доступным вне виртуальной сети Kubernetes, необходимо представить под как сервис Kubernetes.

1) Сделаем под доступным для публичной сети Интернет можно с помощью команды `kubectl expose`:

```
kubectl expose deployment hello-node --type=LoadBalancer --port=8080
```

Флаг `--type=LoadBalancer` показывает, что сервис должен быть виден вне кластера.

Во вкладке Services в веб-браузере можем увидеть появившийся сервис:



| Name       | Labels                                       | Type         | Cluster IP    | Internal Endpoints                          | External Endpoints | Created        |
|------------|--|--------------|---------------|---|--------------------|----------------|
| hello-node | app: hello-node                              | LoadBalancer | 10.103.218.19 | hello-node:8080 TCP<br>hello-node:32080 TCP | -                  | 19 seconds ago |
| kubernetes | component: apiserver<br>provider: kubernetes | ClusterIP    | 10.96.0.1     | kubernetes:443 TCP<br>kubernetes:0 TCP      | -                  | 42 minutes ago |

2) Чтобы посмотреть информация о только что созданном сервисе, напишем:

```
kubectl get services
```

Вывод:

| NAME<br>AGE         | TYPE         | CLUSTER-IP    | EXTERNAL-IP | PORT(S)        |
|---------------------|--------------|---------------|-------------|----------------|
| hello-node<br>2m25s | LoadBalancer | 10.103.218.19 | <pending>   | 8080:32080/TCP |
| kubernetes<br>44m   | ClusterIP    | 10.96.0.1     | <none>      | 443/TCP        |

Для облачных провайдеров, поддерживающих балансировщики нагрузки, для доступа к сервису будет предоставлен внешний IP адрес. В Minikube тип **LoadBalancer** делает сервис доступным при обращении с помощью команды **minikube service**.

3) Выполним следующую команду:

```
minikube service hello-node
```

Получим вывод в терминале:

```
|-----|-----|-----|-----|
| NAMESPACE | NAME      | TARGET PORT | URL                      |
|-----|-----|-----|-----|
| default    | hello-node | 8080        | http://192.168.59.100:32080 |
|-----|-----|-----|-----|
Opening service default/hello-node in default browser...
```

Он переадресует нас в браузер:

```

← → ↻ Не защищено | 192.168.59.100:32080

CLIENT VALUES:
client_address=10.244.0.1
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://192.168.59.100:8080/

SERVER VALUES:
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
accept=text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
accept-encoding=gzip, deflate
accept-language=ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
connection=keep-alive
host=192.168.59.100:32080
upgrade-insecure-requests=1
user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
BODY:
-no body in request-
```

### 3.5 Добавление дополнений

В Minikube есть набор встроенных дополнений, которые могут быть включены, выключены и открыты в локальном окружении Kubernetes.

1) Например включим дополнение `metrics-server`:

```
minikube addons enable metrics-server
```

Получим:

```
metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
```

```
You can view the list of minikube maintainers at:
```

```
https://github.com/kubernetes/minikube/blob/master/OWNERS
```

```
Using image registry.k8s.io/metrics-server/metrics-server:v0.6.3
```

```
The 'metrics-server' addon is enabled
```

2) Посмотрим Pod и Service, которые мы только что создали:

```
kubectl get pod,svc -n kube-system
```

Увидим:

```
PS D:\Proga\kubernetes> kubectl get pod,svc -n kube-system
>>
NAME                                READY   STATUS             RESTARTS   AGE
pod/coredns-787d4945fb-74bhx        1/1     Running            0           53m
pod/etcd-minikube                    1/1     Running            0           53m
pod/kube-apiserver-minikube          1/1     Running            0           53m
pod/kube-controller-manager-minikube 1/1     Running            0           53m
pod/kube-proxy-75x68                 1/1     Running            0           53m
pod/kube-scheduler-minikube          1/1     Running            0           53m
pod/metrics-server-6588d95b98-82ccx 0/1     ContainerCreating  0           7s
pod/storage-provisioner              1/1     Running            1 (52m ago) 53m

NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
service/kube-dns                     ClusterIP      10.96.0.10      <none>        53/UDP,53/TCP,9153/TCP  53m
service/metrics-server                ClusterIP      10.101.60.0     <none>        443/TCP                  7s
```

4) Чтобы отключить сервис просто напомним:

```
minikube addons disable metrics-server
```



## 4 Завершение работы

При завершении работы нужно освободить ресурсы созданного нами кластера:

```
kubectl delete service hello-node  
kubectl delete deployment hello-node
```

Также нужно остановить выполнение виртуальной машины (minikube) и удалить ее:

```
minikube stop  
minikube delete
```

## 5 Обсуждение результатов

В самом начале изучили принципы работы Kubernetes, узнали из каких модулей он состоит и для чего используется. Потом используя документацию с официального сайта установили Minikube и все дополнительные приложения. Также при помощи руководства с официального сайта и Minikube создали приложение на Kubernetes и разобрались как оно работает (посмотрели на его конфигурацию и состояния, создали Deployment и Service).

## Список литературы

- [1] demonight [Основы Kubernetes \(habr\)](#).
- [2] ADV-IT [Поднятие простого Локального K8s Cluster на Windows](#).
- [3] Minikube official documentation [Тестовое приложение: «Привет, Minikube»](#).