

# КУРС

## «Применение контейнеров для работы с данными»

# Лекция 4: Установка **minikube**

Если вы используете **Docker Desktop**, вам может вообще не понадобится **minikube**, поскольку первый уже обеспечивает готовую поддержку **Kubernetes**.

Если вы не можете использовать **Docker Desktop** или по какой-то причине у вас есть доступ только к более старой версии инструмента, которая еще не поддерживает **Kubernetes**, то рекомендуется установить **minikube**.

**minikube** предоставляет одноузловой кластер **Kubernetes** на вашей рабочей станции и доступен через **kubectl** — инструмент командной строки, используемый для работы с **Kubernetes**.

# Установка minikube на Linux, macOS и Windows



Чтобы установить **minikube** в **Linux**, **macOS** или **Windows**, перейдите по следующей ссылке.

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

# Тестирование minikube и kubectl

Попробуем получить доступ к нашему кластеру с помощью **kubectl**.  
Во-первых, нам нужно убедиться, что для **kubectl** выбран правильный контекст. Если вы ранее установили **Docker Desktop**, а теперь **minikube**, вы можете использовать следующую команду:

```
$ kubectl config get-contexts
```

```
• → The-Ultimate-Docker-Container-Book git:(main) ✕ kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO	NAMESPACE
	docker-desktop	docker-desktop	docker-desktop	
*	minikube	minikube	minikube	default

# Тестирование minikube и kubectl

Звездочка рядом с контекстом под названием minikube сообщает нам, что это текущий контекст. Таким образом, при использовании **kubectl** мы будем работать с новым кластером, созданным **minikube**.

Теперь посмотрим, сколько узлов в нашем кластере с помощью этой команды:

```
$ kubectl get nodes
```

```
● → The-Ultimate-Docker-Container-Book git:(main) ✕ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	control-plane	9m9s	v1.25.3

# Тестирование minikube и kubectl



Здесь у нас есть кластер с одним узлом. Роль узла — роль плоскости управления, что означает, что он является главным узлом. Типичный кластер **Kubernetes** состоит из нескольких главных узлов и множества рабочих узлов. Версия **Kubernetes**, с которой мы здесь работаем, — **v1.25.3**.

\$ kubectl get nodes

```
● → The-Ultimate-Docker-Container-Book git:(main) x kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	control-plane	9m9s	v1.25.3

# Тестирование minikube и kubectl

Скачать папку **setup**, содержащую файл **.yaml**, который будем использовать для теста.

1. Откройте новое окно терминала. Создайте модуль **pod** под управлением **Nginx** с помощью следующей команды:

```
$ kubectl apply -f setup/nginx.yaml
```

Вы должны увидеть этот вывод:

```
$ pod/nginx created
```



# Тестирование minikube и kubectl



Можно проверить, работает ли **pod**, с помощью **kubectl**:

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	0	11m

Чтобы получить доступ к серверу **Nginx**, нужно открыть приложение, работающее в модуле, с помощью следующей команды:

```
$ kubectl expose pod nginx --type=NodePort --port=80
```

# Тестирование minikube и kubectl



Можно использовать **kubectl** для вывода списка всех сервисов, определенных в нашем кластере:

```
$ kubectl get services
```

```
→ The-Ultimate-Docker-Container-Book git:(main) x kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	<u>10.96.0.1</u>	<none>	443/TCP	59m
nginx	NodePort	10.104.77.208	<none>	80:30373/TCP	11m

В выводе видим второй сервис под названием **Nginx**, который мы только что создали. Сервис относится к типу **NodePort**; порт 80 модуля был сопоставлен с портом **30373** узла нашего кластера **Kubernetes** в **minikube**.

# Тестирование minikube и kubectl

Теперь можем использовать **minikube**, чтобы создать туннель к кластеру и открыть браузер с правильным URL-адресом для доступа к веб-серверу Nginx. Используйте эту команду:

```
$ minikube service nginx
```

```
→ The-Ultimate-Docker-Container-Book git:(main) x minikube service nginx
```

NAMESPACE	NAME	TARGET PORT	URL
default	nginx	80	http://192.168.49.2:30373

```
🏃 Starting tunnel for service nginx.
```

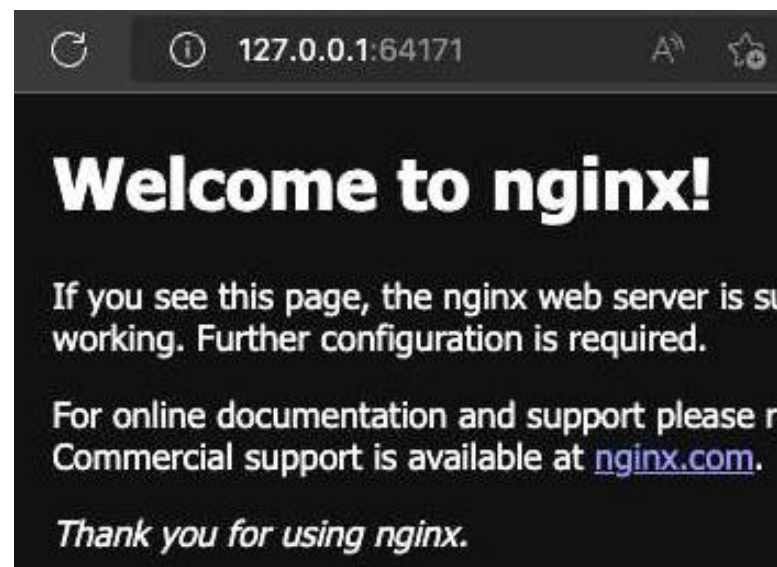
NAMESPACE	NAME	TARGET PORT	URL
default	nginx		http://127.0.0.1:64171

```
🔔 Opening service default/nginx in default browser...  
! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

# Тестирование minikube и kubectl

**minikube** создал туннель для службы **nginx**, который прослушивает порт уел с 30373 на порт 64171 на хосте, который находится на ПК.

Новая вкладка браузера должна была открыться автоматически и привести вас к **<http://127.0.0.1:64171>**. Вы должны увидеть экран приветствия **Nginx**:



# Тестирование minikube и kubectl

**minikube** создал туннель для службы **nginx**, который прослушивает порт уел с 30373 на порт 64171 на хосте, который находится на ПК.

Новая вкладка браузера должна была открыться автоматически и привести вас к **<http://127.0.0.1:64171>**. Вы должны увидеть экран приветствия **Nginx**:



Вы успешно запустили и получили доступ к веб-серверу **Nginx** в одноузловом кластере **Kubernetes** на **minikube**

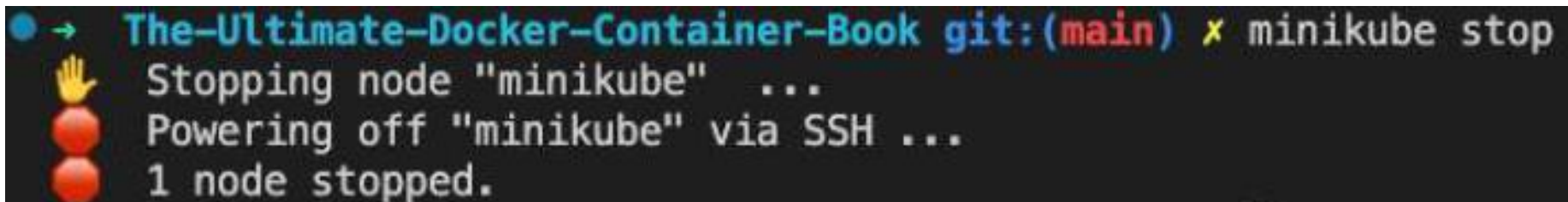
# Завершение работы minikube и kubectl

1. Остановите туннель к кластеру, нажав **Ctrl + C** в окне терминала.
2. Удалите службу **nginx** и модуль в кластере:

```
$ kubectl delete service nginx  
$ kubectl delete pod nginx
```

1. Остановите кластер с помощью следующей команды

```
$ minikube stop
```



```
• → The-Ultimate-Docker-Container-Book git:(main) ✕ minikube stop  
👋 Stopping node "minikube" ...  
🔴 Powering off "minikube" via SSH ...  
🔴 1 node stopped.
```



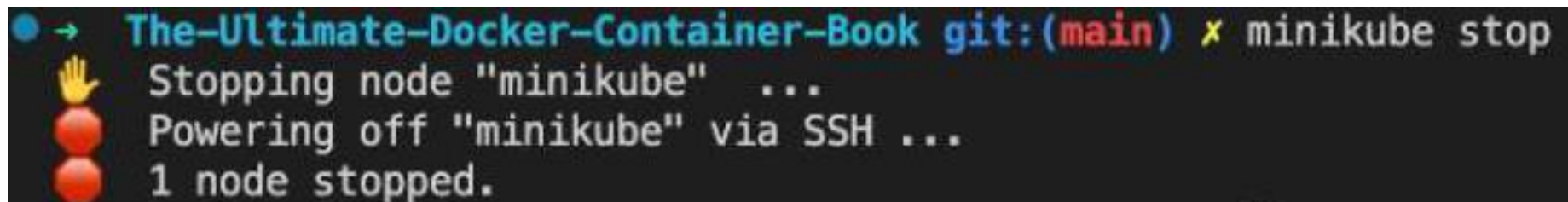
# Завершение работы minikube и kubectl

1. Остановите туннель к кластеру, нажав **Ctrl + C** в окне терминала.
2. Удалите службу **nginx** и модуль в кластере:

```
$ kubectl delete service nginx  
$ kubectl delete pod nginx
```

1. Остановите кластер с помощью следующей команды

```
$ minikube stop
```



```
• → The-Ultimate-Docker-Container-Book git:(main) ✕ minikube stop  
👋 Stopping node "minikube" ...  
🔴 Powering off "minikube" via SSH ...  
🔴 1 node stopped.
```

# Работа с многоузловым кластером **minikube**



# Завершение работы minikube и kubectl



Работа с кластером, состоящим из нескольких узлов в миникубе, использовать команду:

```
$ minikube start --nodes 3 -p demo
```

создается кластер с тремя узлами и называется демонстрационным. Используйте **kubectl** для получения списка всех узлов вашего кластера:

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
demo	Ready	control-plane	84s	v1.25.3
demo-m02	Ready	<none>	45s	v1.25.3
demo-m03	Ready	<none>	22s	v1.25.3

# Завершение работы minikube и kubectl



остановить кластер:

```
$ minikube stop -p demo
```

Удалите все кластеры в системе с помощью команды:

```
$ minikube delete --all
```

# Установка **Kind**

# Установка **Kind**



**Kind** — еще один популярный инструмент, который можно использовать для локального запуска многоузлового кластера **Kubernetes** на вашем компьютере. Его очень легко установить и использовать.

Для установки **Kind** используйте соответствующий менеджер пакетов для вашей платформы. Более подробную информацию о процессе установки, перейдите по следующей ссылке.

<https://kind.sigs.k8s.io/docs/user/quick-start/>

# Установка Kind

На компьютере с **Linux** использовать следующий сценарий для установки **Kind** из его двоичных файлов:

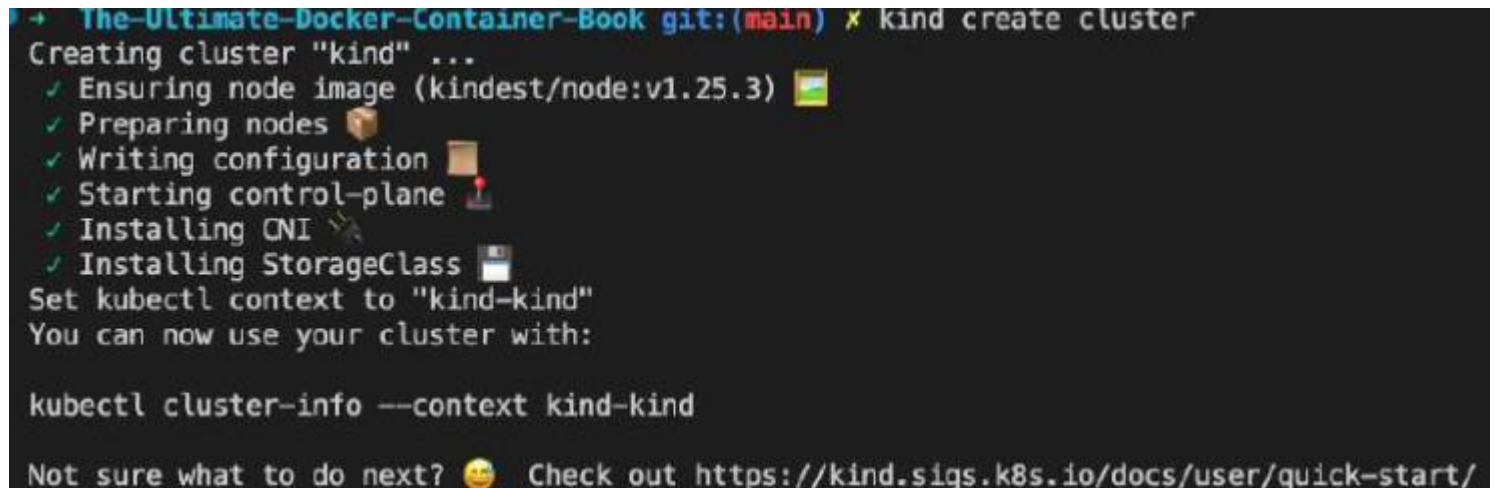
```
$ curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.17.0/kind-linux-amd64
```

```
$ chmod +x ./kind
```

```
$ sudo mv ./kind /usr/local/bin/kind
```

```
$ kind version
```

```
$ kind create cluster
```



```
+ The-Ultimate-Docker-Container-Book git:(main) x kind create cluster
Creating cluster "kind" ...
  ✓ Ensuring node image (kindest/node:v1.25.3)
  ✓ Preparing nodes
  ✓ Writing configuration
  ✓ Starting control-plane
  ✓ Installing CNI
  ✓ Installing StorageClass
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Not sure what to do next? 😊 Check out https://kind.sigs.k8s.io/docs/user/quick-start/
```

# Установка **Kind**



Чтобы убедиться, что кластер создан, используйте следующую команду:

```
$ kind get clusters
```

создать дополнительный кластер с другим именем, используя параметр **--name**:

```
$ kind create cluster --name demo
```

```
$ kind show clusters
```

# Testing Kind

**\$ kubectl config get-contexts**

```
● → The-Ultimate-Docker-Container-Book git:(main) x k config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO	NAMESPACE
	docker-desktop	docker-desktop	docker-desktop	
*	kind-demo	kind-demo	kind-demo	
	kind-kind	kind-kind	kind-kind	

Используйте следующую команду, чтобы сделать демонстрационный кластер вашим текущим кластером, если звездочка указывает, что текущим является другой кластер:

**\$ kubectl config use-context kind-demo**

**\$ kubectl get nodes**

```
● → The-Ultimate-Docker-Container-Book git:(main) x kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
demo-control-plane	Ready	control-plane	2m25s	v1.25.3

# Testing **Kind**

Теперь попробуем запустить первый контейнер в этом кластере. Использовать доверенный веб-сервер **Nginx**, как и раньше. Используйте следующую команду для его запуска:

```
$ kubectl apply -f setup/nginx.yaml
```

```
$ kubectl port-forward nginx 8080 80
```

```
Forwarding from 127.0.0.1:8080 -> 80  
Forwarding from [::1]:8080 -> 80
```

Откройте новую вкладку браузера и перейдите по адресу **<http://localhost:8080>**; вы должны увидеть экран приветствия **Nginx**.



# Testing **Kind**

Закончив работу с **Nginx**, используйте команду, чтобы удалить модуль из кластера:

```
$ kubectl delete -f setup/nginx.yaml
```

```
$ kind delete cluster --name kind
```

```
$ kind delete cluster --name demo
```

# ЗАДАНИЕ

1. Установить **minikube**. Создайте модуль **pod** под управлением **Nginx**.
2. Провести тестирование **minikube** и **kubectl**.
3. Развернуть многоузловой кластер **minikube**.
4. Установить **Kind**. Создайте модуль **pod** под управлением **Nginx**.
5. Провести тестирование **Kind**.

# ЛИТЕРАТУРА

1. Chocolatey – The Package Manager for Windows: <https://chocolatey.org/>
2. Run Docker on Hyper-V with Docker Machine: <http://bit.ly/2HGMPiI>
3. Developing inside a Container: <https://code.visualstudio.com/docs/remote/containers>

СПАСИБО ЗА ВНИМАНИЕ