#### Создание пространства имен (namespace) в Kubernetes

Пространства имен в Kubernetes позволяют организовать ресурсы и управлять ими изолированно.

Запустите терминал. Убедитесь, что ваш kubectl настроен правильно и подключен к вашему кластеру Kubernetes. Вы можете проверить это с помощью команды:

kubectl cluster-info

```
C:\Windows\system32>kubectl cluster-info
Kubernetes control plane is running at https://kubernetes.docker.internal:6443
CoreDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Если команда отобразит информацию о вашем кластере, вы подключены правильно.

Создайте пространство имен:

kubectl create namespace my-namespace

Проверьте созданные пространства имен:

Чтобы убедиться, что пространство имен было создано, выполните команду:

kubectl get namespaces

```
C:\Windows\system32>kubectl get namespace
NAME
                 STATUS
                          AGE
default
                 Active
                          26h
kube-node-lease Active
                          26h
                Active
kube-public
                          26h
                 Active
kube-system
                          26h
                 Active
                          25h
test
```

Вы должны увидеть ваше новое пространство имен в списке.

C:\Windows\system32>kubectl config set-context --current --namespace=test
Context "docker-desktop" modified.

C:\Windows\system32>kubectl get pod
No resources found in test namespace.

#### 1. Pod

*Модуль* - группа контейнеров, которые представляют собой основной строительный блок к8с состоящий из одного или нескольких тесносвязанных контейнеров, которые будут выполняться вместе на 1 рабочем узле и 1 пространстве имен. Каждый модуль подобен отдельно логической машине с собственными процессами, ір. Все контейнеры работают только на 1 рабочем узле WorkNode).

Модуль при содержании нескольких контейнеров всегда работаtт на 1 узле.

Далее проверим, какие РОДы у нас вообще есть (все фразы будут подразумевать созданный namespace - test). В данном случае - никаких подов нет.

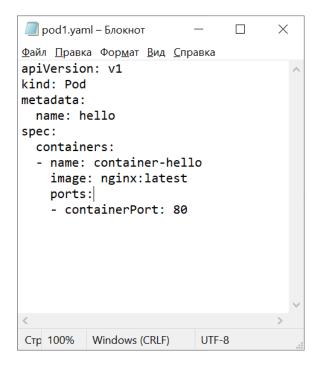
kubectl get pod

mnagapetyan@mnagapetyan-ThinkPad-P1-Gen-6:~\$ kubectl get pod
No resources found in test namespace.

Рассмотрим несколько примеров POD:

<u>Пример 1:</u> создадим POD для nginx. Сперва создаем pod1.yaml, далее "активируем" файл, после чего проверяем созданный под. Через describe посмотрим характеристики пода. Там есть важная информация: как POD называется, в каком namespace находится, когда был создан, по какому образу, id контейнера, события и т.п.

spec - спецификация контейнера



C:\Windows\system32>cd c:\temp
c:\Temp>kubectl apply -f pod1.yaml
pod/hello created

c:\Temp>kubectl get pods NAME READY STATUS RESTARTS AGE hello 1/1 Running 0 45s

```
::\Temp>kubectl describe pod hello
            hello
test
Name:
                 test
0
Namespace:
Service Account: default
Node: docker-desktop/192.168.65.3
Start Time: Thu, 07 Nov 2024 12:20:47 +0300
Labels: <none>
Annotations: <none>
Status:
Status:
                   Running
IP:
                   10.1.0.14
IPs:
 IP: 10.1.0.14
Containers:
 container-hello:
   Container ID: docker://5ab53648f66547e945aa70b109a618e7db120f1b042889668556fa285f150c90
    Image:
                    nginx:latest
    Image ID:
                    docker-pullable://nginx@sha256:28402db69fec7c17e179ea87882667f1e054391138f77ff
af0c3eb388efc3ffb
    Port:
                    80/TCP
    Host Port:
                   0/TCP
                   Running
   State:
     Started: Thu, 07 Nov 2024 12:20:53 +0300
    Ready:
                     True
    Restart Count: 0
   Environment:
                     <none>
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-h7frx (ro)
Conditions:
  Type
                                Status
  PodReadyToStartContainers
                                True
  Initialized
                                True
                                True
  ContainersReady
                               True
```

Чтобы проверить, что контейнер nginx работает, делаем port-forward с перенаправлением портов, например, на 5555 (можно взять любой). Далее пишем в браузере localhost:5555 (в minikube вместо localhost возможно нужно написать ір ноды (введите minikube ір)) и проверяем. Все работает, все прекрасно. Удалим созданный под.

```
c:\Temp>kubectl port-forward hello 5555:80
Forwarding from 127.0.0.1:5555 -> 80
Forwarding from [::1]:5555 -> 80
Handling connection for 5555
Handling connection for 5555
```

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at <a href="mailto:nginx.com">nginx.com</a>.

Thank you for using nginx.

```
c:\Temp>kubectl get pods
NAME
        READY
                STATUS
                           RESTARTS
                                      AGE
hello
                Running
        1/1
                           0
                                      53m
c:\Temp>kubectl delete pod hello
pod "hello" deleted
c:\Temp>kubectl get pods
No resources found in test namespace.
```

Пример 2: Создадим POD с 2-я контейнерами: nginx и tomcat. Создадим pod2.yaml. Добавим label как некоторые метаданные. Через describe посмотрим характеристики подов. Потом опять перенаправим порты и посмотрим, что все работает

```
*pod2.yaml – Блокнот
                                          <u>Ф</u>айл <u>П</u>равка Фор<u>м</u>ат <u>В</u>ид <u>С</u>правка
apiVersion: v1
kind: Pod
metadata:
 name: app-2
 labels:
   env: prod
   app: main
   tier: frontend
   owner: user
spec:
 containers:
 - name: container-hello
   image: nginx:latest
   ports:
     - containerPort: 80
 - name: container-tomcat
   image: tomcat:8.5.38
    ports:
     - containerPort: 8080
Стр 22, стлб 1 100% Windows (CRLF)
```

```
c:\Temp>kubectl get pods
No resources found in test namespace.
c:\Temp>kubectl apply -f pod2.yaml
pod/app-2 created
c:\Temp>kubectl get pods
NAME READY STATUS RESTARTS AGE
app-2 2/2 Running 0 75s
```

```
::\Temp>kubectl describe pod app-2
Name:
                 app-2
Namespace:
                 test
Priority:
                 0
Service Account: default
Node:
                 docker-desktop/192.168.65.3
Start Time:
                 Thu, 07 Nov 2024 13:21:39 +0300
                 app=main
Labels:
                 env=prod
                 owner=user
                 tier=frontend
Annotations:
                 <none>
Status:
                 Running
IP:
                 10.1.0.16
IPs:
 IP: 10.1.0.16
Containers:
 container-hello:
   Container ID:
                   docker://a9a770ec0f4a48697a984da96bd68afe4d0d01969b9224668e328c3c16711573
    Image:
                    nginx:latest
   Image ID:
                   docker-pullable://nginx@sha256:28402db69fec7c17e179ea87882667f1e054391138f77ff
af0c3eb388efc3ffb
   Port:
                   80/TCP
   Host Port:
                   0/TCP
                   Running
   State:
                   Thu, 07 Nov 2024 13:21:45 +0300
     Started:
   Ready:
                    True
   Restart Count: 0
   Environment:
                   <none>
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4g5gw (ro)
  container-tomcat:
   Container ID:
                  docker://639057916a69e5bc2e5a43ee74ca553570c44c67e9b5124ea2d9a55cf04b2e4f
```

```
c:\Temp>kubectl port-forward app-2 5555:80
Forwarding from 127.0.0.1:5555 -> 80
Forwarding from [::1]:5555 -> 80
Handling connection for 5555
Handling connection for 5555
c:\Temp>
c:\Temp>kubectl port-forward app-2 5555:8080
Forwarding from 127.0.0.1:5555 -> 8080
Forwarding from [::1]:5555 -> 8080
Handling connection for 5555
```

(i) localhost:5555

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to <u>nginx.org</u>. Commercial support is available at <u>nginx.com</u>.

Thank you for using nginx.



Home Documentation Configuration Examples Wiki Mailing Lists

#### Apache Tomcat/8.5.38



### If you're seeing this, you've successfully installed Tomcat. Congratulations!



**Recommended Reading:** 

Security Considerations HOW-TO

Manager Application HOW-TO

Clustering/Session Replication HOW-TO

Server S

Manager

Host Ma

#### **Developer Quick Start**

Tomcat Setup First Web Application Realms & AAA

JDBC DataSources

**Examples** 

Servlet Specifications
Tomcat Versions

#### Удаляем все

c:\Temp>kubectl delete pod app-2
pod "app-2" deleted

c:\Temp>kubectl get pods
No resources found in test namespace.

## 2. Deployment

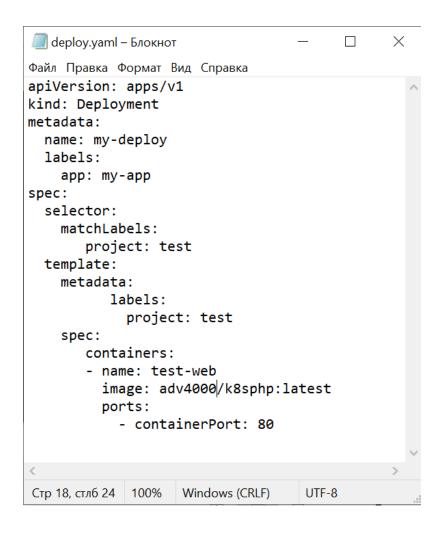
Ресурс более высокого уровня, предназначенный для развертывания приложения и их обновления декларативным образом. Деплоймент (Deployment) — это объект Kubernetes, который управляет развертыванием подов. Он описывает желаемое состояние приложения и обеспечивает автоматическое обновление и масштабирование подов.

- *Pods* это самая базовая единица развёртывания в Kubernetes, в то время как *Deployment* это более сложные абстракции высокого уровня для управления наборами *Pods*.
- Из-за недолговечности *модулей* управление их жизненным циклом по отдельности требует значительных усилий. *Deployment* обеспечивают автоматическое управление жизненным циклом содержащихся в них модулей, снижая нагрузку на пользователя.
- В то время как *модули* могут быть отключены в случае сбоя узла или чрезмерного использования ресурсов, *Deployment* непрерывно отслеживают и заменяют отключенные модули для поддержания желаемого состояния.
- *Модули* определяются независимо друг от друга; *Deployment*, однако, объединяют определения модулей, обеспечивая дополнительный уровень управления.
- *Pods* требуют ручного обновления и вмешательства для развёртывания и масштабирования. *Deployment* позволяет автоматически обновлять, откатывать и масштабировать, обеспечивая улучшенное управление кластером.

<u>Пример 1:</u> Напишем файл, который загружает образ из интернета . Проверяем, что был создан *Pod* и *Deployment*, а также работоспособность.

(этот файл описывает развертывание приложения с именем ту-deploy, которое управляет подами, содержащими контейнер test-web. Контейнер будет использовать образ adv/k8sphp:latest и будет прослушивать 80-й порт. Поды будут иметь метку project: test для идентификации и выборки с помощью селектора.)

selector - с какими подами будет работать. Работа с подами: template - создание подов (введение label), spec - спецификация с какими контейнерами



Создаем и проверяем наличие deployment, pod

```
c:\Temp>kubectl apply -f deploy.yaml
deployment.apps/my-deploy created
c:\Temp>kubectl get deploy
NAME
         READY UP-TO-DATE
                               AVAILABLE
                                          AGE
my-deploy 0/1
                                           13s
c:\Temp>kubectl get pod
                           READY
                                   STATUS
                                                                 AGE
                                                      RESTARTS
my-deploy-6c46db9c6c-7ts8d
                           0/1
                                   ContainerCreating
                                                                 19s
```

Делаем перенаправление портов пода на 5555 порт, чтоб можно было в браузере открыть. Проверяем

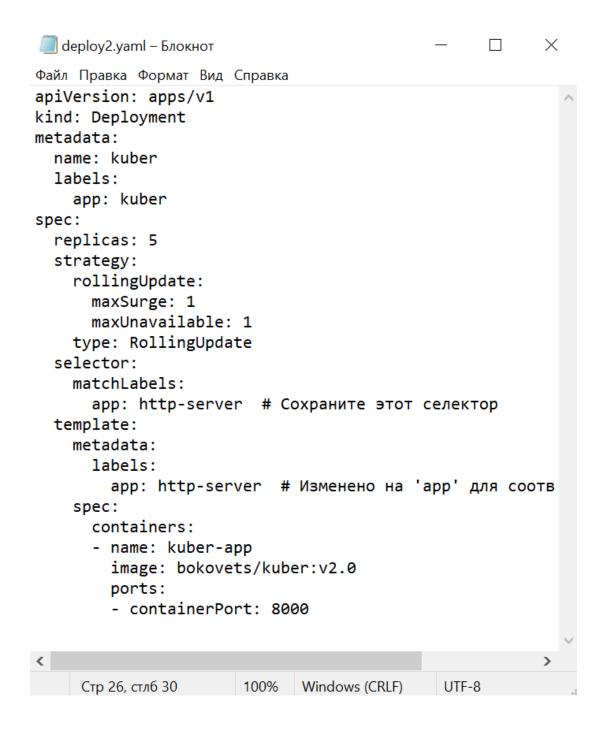
```
c:\Temp>kubectl port-forward my-deploy-6c46db9c6c-7ts8d 5555:80
Forwarding from 127.0.0.1:5555 -> 80
Forwarding from [::1]:5555 -> 80
Handling connection for 5555
Handling connection for 5555
```



# Hello from Kubernetes

Server IP Address is: 127.0.0.1

<u>Пример 3:</u> deployment можно усложнять, например, добавляя стратегии (будем изменять запуск)



Создаем 5 реплик (копий подов), RollingUpdate - плавное обновление подов, maxSurge - максимальный всплеск (сколько подов дополнительно создается....один создается, один удаляется),

minReadySeconds - через 10 секунд каждый под будет доступен