Minikube

Задание в процессе выполнения

Minikube — специализированная конфигурация Kubernetes, содержащая кластер со всего одним узлом. В основном предназначена для развёртывания на локальной машине разработчика и применяется для экспериментов над Kubernetes. Minikube реализован для всех основных ОС.

Единственный узел, содержащийся в Minikube, будет одновременно являться worker-node и master-node.

Minikube может работать с различными системами контейнеризации и виртуализации. Если мы хотим заменить CRI, использование Docker как среды выполнения Minikube запрещено. Minikube в таком случае использует CRI самого докера, и изменить это никак не получится, так что мы будем использовать систему виртуализации.

Все файлы конфигурации для этой работы хранятся в моём репозитории GitHub.

Особенности установки и запуска minikube на Windows

В Windows10 Pro по дефолту установлен HyperV, и при попытке указать в качестве драйвера VirtualBox, minikube выдаст следующую ошибку:

```
C:\>minikube start --driver=virtualbox
* minikube v1.33.1 на Microsoft Windows 10 Pro 10.0.19045.3803 Build 19045.3803
E0829 12:11:47.355494   15588 start.go:812] api.Load failed for minikube: filestore "minikube": Docker machine "minikube" does not exist. Use "dock
er-machine ls" to list machines. Use "docker-machine create" to add a new one.
E0829 12:11:47.356667   15588 start.go:812] api.Load failed for minikube: filestore "minikube": Docker machine "minikube" does not exist. Use "dock
er-machine ls" to list machines. Use "docker-machine create" to add a new one.
* Используется драйвер virtualbox на основе существующего профиля
* Starting "minikube" primary control-plane node in "minikube" cluster
* Creating virtualbox VM (CPUs=2, Memory=4000MB, Disk=20000MB) ...
! StartHost failed, but will try again: creating host: create: precreate: This computer is running Hyper-V. VirtualBox won't boot a 64bits VM when
Hyper-V is activated. Either use Hyper-V as a driver, or disable the Hyper-V hypervisor. (To skip this check, use --virtualbox-no-vtx-check)
* Creating virtualbox VM (CPUs=2, Memory=4000MB, Disk=20000MB) ...
* Failed to start virtualbox VM. Running "minikube delete" may fix it: creating host: create: precreate: This computer is running Hyper-V. VirtualB
ox won't boot a 64bits VM when Hyper-V is activated. Either use Hyper-V as a driver, or disable the Hyper-V hypervisor. (To skip this check, use --
virtualbox-no-vtx-check)

X Exiting due to PR_VBOX_HYPERV_64_BOOT: Failed to start host: creating host: create: precreate: This computer is running Hyper-V. VirtualBox won't
 boot a 64bits VM when Hyper-V is activated. Either use Hyper-V as a driver, or disable the Hyper-V hypervisor. (To skip this check, use --virtualb
ox-no-vtx-check)
* Предложение: VirtualBox and Hyper-V are having a conflict. Use '--driver=hyperv' or disable Hyper-V using: 'bcdedit /set hypervisorlaunchtype off
'
* Related issues:
  - https://github.com/kubernetes/minikube/issues/4051
  - https://github.com/kubernetes/minikube/issues/4783
```

Minikube не может запустить VM в VirtualBox при наличии работающего HyperV. Решение - отключить HyperV или использовать его в качестве драйвера. Так как мы, используя minikube, не взаимодействуем с гипервизором, его вендор не имеет значения.

Замена CRI/CNI при запуске Minikube

Minikube предлагает возможность заменить CRI (Container Runtime Interface) средствами самого Minikube без необходимости редактирования конфигураций вручную. Данный способ можно использовать только при слабой ауре разработчика, т.к. в Kubernetes для этого есть механизмы промышленного уровня.

Освоение основ работы с системой оркестрации на примере Minikube

Просмотр списка узлов

```
kubectl get nodes
```

Просмотр сведений об узле с именем "minikube"

```
kubectl describe node minikube
```

По умолчанию команды вывода информации о ресурсах кластера показывают ресурсы, находящиеся только в текущем пространстве имён.

Мы можем явно указывать пространство имён, по которому хотим видеть вывод команды с помощью опции --namespace. Например:

```
PS D:\k8s-education> kubectl get pods --namespace=kube-system
NAME                               READY   STATUS    RESTARTS        AGE
coredns-7db6d8ff4d-56wvd           1/1     Running   0               4h30m
coredns-7db6d8ff4d-87bcd           1/1     Running   0               4h30m
etcd-minikube                      1/1     Running   0               4h30m
kube-apiserver-minikube            1/1     Running   0               4h30m
kube-controller-manager-minikube   1/1     Running   0               4h30m
kube-proxy-lhb4k                   1/1     Running   0               4h30m
kube-scheduler-minikube            1/1     Running   0               4h30m
storage-provisioner                1/1     Running   1 (4h29m ago)   4h30m
```

Также можем запросить вывод команды по всем пространствам имён через опцию --all-namespaces

Очистить namespace от созданных нами ресурсов

```
kubectl delete pods,deployments,replicasets --all -n <namespace>
```

Посмотреть подробную информацию по всем ресурсам (в текущем namespace)

```
kubectl get all -o wide
```

Создание контейнера с сетевыми утилитами внутри кластера

```
kubectl run --rm -it --image amouat/network-utils test bash
```

Без каких-либо сетевых настроек поды имеют только внутренний IP, к которому мы можем обратиться, находясь в одной сети с ними. Чтобы отправить запрос на какой-либо под, мы можем запустить в одной с ними сети контейнер с сетевыми утилитами и подключиться к его командной оболочке.

Работа с Pod-ами

Запуск пода

```
kubectl apply -f pod.yaml
```

Получить список подов. Параметр -o указывает на подробный формат вывода

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

Вывести информацию по ресурсам (нодам/подам). В данном случае вывод инфы по поду с именем "my-pod"

```
kubectl describe pod "my-pod"
```

Удаляем под из кластера

```
kubectl delete pod "my-pod"
```

Удалить всё, что было описано в файле конфигурации

```
kubectl delete -f pod.yaml
```

Работа с ReplicaSet-ами

Запуск ReplicaSet-а

```
kubectl apply -f replicaset.yaml
```

Посмотрим вывод команды *get pods* - видим, что было создано 2 пода. Их имена состоят из имени replicaSet + уникальная последовательность символов

```
PS D:\k8s-education> kubectl get pods -o wide
NAME                 READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
my-replicaset-gqb5c  1/1     Running   0          39h   10.244.0.6   minikube   <none>           <none>
my-replicaset-p5hmc  1/1     Running   0          39h   10.244.0.5   minikube   <none>           <none>
```

```
PS D:\k8s-education> kubectl get replicasets -o wide
NAME            DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES        SELECTOR
my-replicaset   2         2         2       39h   nginx        nginx:1.12    app=my-app
```

Попробуем удалить один из подов. Kubernetes тотчас создаст ещё один, на замену удалённому, чтобы привести кластер к состоянию, указанному в replicaSet.

```
PS D:\k8s-education> kubectl delete pod my-replicaset-p5hmc
pod "my-replicaset-p5hmc" deleted
PS D:\k8s-education> kubectl get pods -o wide
NAME                 READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
my-replicaset-gqb5c  1/1     Running   0          39h   10.244.0.6   minikube   <none>           <none>
my-replicaset-kjv9q  1/1     Running   0          4s    10.244.0.7   minikube   <none>           <none>
```

⚠ Важно знать, что Kubernetes при изменении конфигурации ReplicaSet не обновляет содержащиеся в ней образы на уже работающих подах (на новых подах уже будет работать новая версия образов). ReplicaSet отвечает только за количество реплик.

Работа с Deployment

Deployment обеспечивает механизм обновления pod-ов с помощью ReplicaSet-ов.

Посмотрим на созданные деплойментом ресурсы:

Мы видим, что Deployment создал ReplicaSet. Поэтому имена подов состоят из имени "корневого" конфигурационного файла, id реплики и id пода. Если бы под был создан из реплики, в его названии присутствовало бы только имя репликасета и его id.

После обновления подов старые ReplicaSet-ы, созданные Deployment-ом остаются пустыми. Это нужно для хранения истории обновлений. В параметре 'revisionHistoryLimit' мы можем задать, сколько реплик стоит хранить. По умолчанию 10.

```
PS D:\k8s-education> kubectl get all -o wide
NAME                                 READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
pod/my-deployment-7fb49dd4d5-db7s7   1/1     Running   0          36s   10.244.0.12  minikube   <none>           <none>
pod/my-deployment-7fb49dd4d5-mnt87   1/1     Running   0          36s   10.244.0.11  minikube   <none>           <none>
                  id реплики  id пода

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE   SELECTOR
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   45h   <none>

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES       SELECTOR
deployment.apps/my-deployment      2/2     2            2           36s   nginx        nginx:1.12   name=my-app

NAME                                  id реплики   DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES       SELECTOR
replicaset.apps/my-deployment-7fb49dd4d5   2         2         2       36s   nginx        nginx:1.12   name=my-app,pod-template-hash=7fb49
```

Deployment: откат изменений

Для того, чтобы велась история изменений и была возможность откатиться, применять конфигурацию следует с параметром --record=true

```
kubectl apply -f .\deployment.yaml --record=true
```

Попробуем обновить версию образа (заведомо с ошибкой)

```
kubectl set image deployment.apps/my-deployment nginx=nginx:1.999 --record=true
```

Deployment был успешно применён, только наши поды не работают. Их статус - ErrImagePull.

```
PS D:\k8s-education> kubectl get all -o wide
NAME                                 READY   STATUS         RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
pod/my-deployment-576b854b9c-gdp5c   0/1     ErrImagePull   0          6s    10.244.0.23  minikube   <none>           <none>
pod/my-deployment-576b854b9c-w6jrj   0/1     ErrImagePull   0          6s    <none>       minikube   <none>           <none>

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE   SELECTOR
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   46h   <none>

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES        SELECTOR
deployment.apps/my-deployment      0/2     2            0           33m   nginx        nginx:1.999   name=my-app

NAME                                     DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES        SELECTOR
replicaset.apps/my-deployment-576b854b9c   2         2         0       6s    nginx        nginx:1.999   name=my-app,pod-template-hash=576b854b9c
replicaset.apps/my-deployment-7fb49dd4d5   0         0         0       33m   nginx        nginx:1.12    name=my-app,pod-template-hash=7fb49dd4d5
```

С помощью команды describe мы можем подробнее узнать, какая ошибка произошла в поде.

```
kubectl describe pod my-deployment-576b854b9c-qsnln
```

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  16m                default-scheduler  Successfully assigned default/my-deployment-576b854b9c-qsnln to minikube
  Normal   Pulling    14m (x4 over 16m)  kubelet            Pulling image "nginx:1.999"
  Warning  Failed     14m (x4 over 16m)  kubelet            Failed to pull image "nginx:1.999": Error response from daemon: manifest for nginx:1.999 not found: manifest unknown: manifest unknown
  Warning  Failed     14m (x4 over 16m)  kubelet            Error: ErrImagePull
  Warning  Failed     14m (x6 over 16m)  kubelet            Error: ImagePullBackOff
  Normal   BackOff    57s (x64 over 16m) kubelet            Back-off pulling image "nginx:1.999"
```

Мы также можем посмотреть историю изменения конфигурации

Команда просмотра истории изменения конфигурации

```
kubectl rollout history deployment my-deployment
```

```
PS D:\k8s-education> kubectl rollout history deployment my-deployment
deployment.apps/my-deployment
REVISION  CHANGE-CAUSE
1         kubectl.exe apply --filename=.\deployment.yaml --record=true
2         kubectl.exe set image deployment.apps/my-deployment nginx=nginx:1.999 --record=true
```

По каждой из ревизий мы можем посмотреть более подробную информацию

```
kubectl rollout history deployment.apps/my-deployment --revision=4
```

```
PS D:\k8s-education> kubectl rollout history deployment.apps/my-deployment --revision=4
deployment.apps/my-deployment with revision #4
Pod Template:
  Labels:       name=my-app
        pod-template-hash=576b854b9c
  Annotations:  kubernetes.io/change-cause: kubectl.exe set image deployment.apps/my-deployment nginx=nginx:1.999 --record=true
  Containers:
   nginx:
    Image:      nginx:1.999
    Port:       80/TCP
    Host Port:  0/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

Теперь посмотрим текущее состояние развёртывания и его статус.

```
kubectl rollout status deployment my-deployment
```

```
PS D:\k8s-education> kubectl apply -f .\deployment.yaml --record=true
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/my-deployment configured
PS D:\k8s-education> kubectl rollout status deployment.apps/my-deployment
deployment "my-deployment" successfully rolled out
PS D:\k8s-education> kubectl set image deployment.apps/my-deployment nginx=nginx:1.999 --record=true
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/my-deployment image updated
PS D:\k8s-education> kubectl rollout status deployment.apps/my-deployment
Waiting for deployment "my-deployment" rollout to finish: 0 of 2 updated replicas are available...
```

Мы можем откатиться на предыдущую ревизию с помощью следующей команды (или на произвольную ревизию с использованием флага --version):

```
kubectl rollout undo deployment.apps/my-deployment
```

Работа с Service

Service обеспечивает сетевой доступ к приложению. Его задача - заставлять прокси настраиваться для пересылки запросов на набор контейнеров.

Service - это абстракция для предоставления сетевого доступа к приложению, работающему на группе pod'ов, а также для балансировки запросов к этим pod'ам.

Получить список сервисов

```
kubectl get services
```

Сервис имеет собственный IP-адрес

Сервис с автоматически создает еще один тип сущности - endpoints. Endpoints- это те конечные точки, на которые сервис будет перенаправлять приходящий на него трафик

Посмотреть список эндпоинтов

```
kubectl get endpoints
```

```
PS D:\k8s-education> kubectl get endpoints
NAME                   ENDPOINTS                       AGE
kubernetes             192.168.59.101:8443             2d1h
my-service-cluster-ip  10.244.0.31:80,10.244.0.32:80   38m
```

Давайте попробуем отправить HTTP GET-запрос на IP-адрес кластера (к сервису мы можем также обращаться по доменному имени). В каждом ответе мы можем видеть имя pod-а, на который сервис переадресовал запрос

```
root@test:/# curl 10.111.42.196
Hello from my-deployment-86cfc646db-vq7sw
root@test:/# curl 10.111.42.196
Hello from my-deployment-86cfc646db-vq7sw
root@test:/# curl 10.111.42.196
Hello from my-deployment-86cfc646db-qdkhn
root@test:/# curl 10.111.42.196
Hello from my-deployment-86cfc646db-qdkhn
root@test:/# curl my-service-cluster-ip
Hello from my-deployment-86cfc646db-vq7sw
```

Основные типы сервисов следующие:

- Без селектора
- ClusterIP
- NodePort
- LoadBalancer

Работа с Container probes

Здесь пока приведём теоретическую справку, что такое пробы, и зачем они нужны.

Container probes - проверки доступности контейнера.

Существуют 2 основных вида проверок:

- readynessProbe - проверка готовности контейнера
- livenessProbe - проверка работоспособности контейнера