Федеральное государственное автономное образовательное учреждение
высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И.
ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №7
«Знакомство с Kubernetes»

Практическая работа
по дисциплине «Современные технологии программирования»
студента 1 курса группы ПИ-б-о-231
Покидько Максим Сергеевич

направления подготовки 09.03.04 «Программная инженерия»

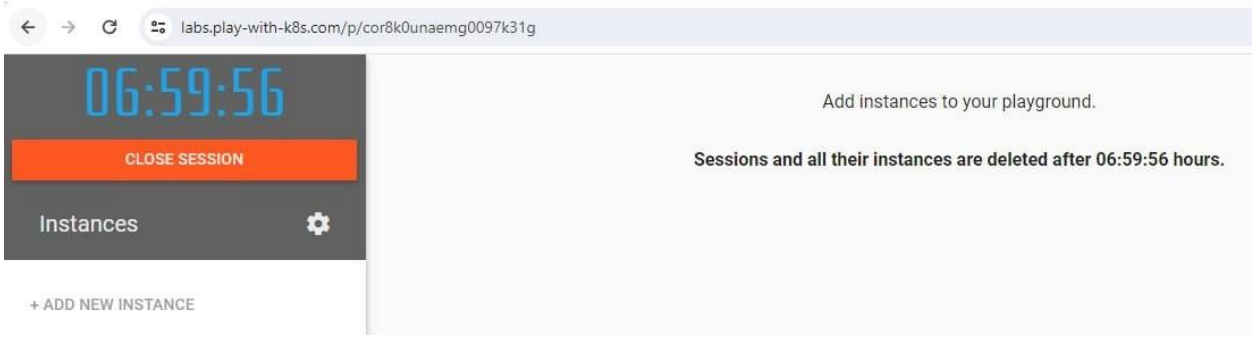
Симферополь, 2024

Цель: Ознакомиться на практике с инструментом оркестрации контейнерирированных приложений Kubernetes.

Ход выполнения задания:

Объединение серверов в кластер

1) Перейдём на сайт https://labs.play-with-k8s.com/ и залогинимся при помощи учётной записи GitHub



2) После того, как мы нажали на кнопку "Start" для нас создана 7-и часовая сессия, в пределах которой можно будет создать до 5-ти серверов (node) с уже установленными компонентами kubernetes.

Создадим максимум нод при помощи кнопки "Add new instance"

3) Каждую ноду можно воспринимать как отдельный сервер подключённый к подсети 192.168.0.0. Несмотря на то, что это локальная сеть (т.е. недоступная из интернета) каждой ноде присвоено доменное имя (url) на поддомене

direct.labs.play-with-k8s.com по которому уже можно будет достучаться до сервера

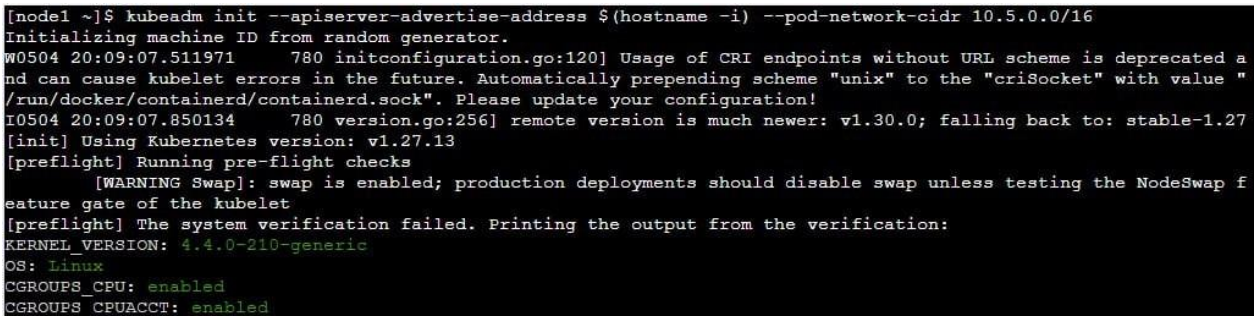
4) На каждой ноде мы увидим одинаковое приветствие с предложением выполнить несколько команд, которые (1) инициируют на машине мастер-ноду, (2) создадут сеть внутри кластера и (3) установят приложение "nginx"

5) В данном случае мы будем создавать кластер с одной мастер-нодай (хотя можно и больше) и 4-мя рабочими нодами. Поэтому:

На первой ноде запустим команду (Ctrl + Shift + C для вставки):

kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

Это приведёт к инициализации на текущей машине управляющих компонент кластера (Control Plane).



После завершения процесса инициализации нас попросят указать путь к конфиг-файлу кластера. По умолчанию, kubernetes ищет конфиг в рабочем каталоге пользователя в папке ~/.kube/config или в переменной окружения KUBECONFIG.

Мы пропускаем этот этап, т.к. kubernetes сам создал в нашем рабочем каталоге файл с конфигом, но это произошло потому, что мы запустили процедуру установки из под этого пользователя, и если бы мы хотели передать возможность работать с нашим кластером другому пользователю, то конфиг нужно было бы скопировать ему в рабочий каталог. Т.е. под выбранным для управления кластером пользователем нужно было бы выполнить команды:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Здесь же будет показана строка которую нужно будет запускать на рабочих узлах для подключения к кластеру. Найдём её и скопируем в текстовый файл, она нам ещё пригодится. В моём случае она выглядит так:

```
kubeadm join 192.168.0.8:6443 --token 0qofqb.jbna9mt0g2x6l3z9 \
        --discovery-token-ca-cert-hash sha256:79da08f19b0f604eb60e3b7c885b4e634443f6c9ef8805d84e63f604eed3c3f9
```

На мастер-ноде запустим команду:

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

Она приведёт к инициализации сети внутри кластера

```
[node1 ~]$ kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
configmap/kube-router-cfg created
daemonset.apps/kube-router created
serviceaccount/kube-router created
clusterrole.rbac.authorization.k8s.io/kube-router created
clusterrolebinding.rbac.authorization.k8s.io/kube-router created
[node1 ~]$
```

Теперь у нас есть полностью готовая к работе мастер-нода

6) Выполним команду:

kubectl get nodes

В результате мы увидим список подключённых к кластеру нод. В нашем случае, в списке, пока что доступна только одна нода

```
[node1 ~]$ kubectl get nodes
NAME    STATUS   ROLES           AGE     VERSION
node1   Ready    control-plane   2m14s   v1.27.2
[node1 ~]$
```

7) Выполним команду:

kubectl get pods -A

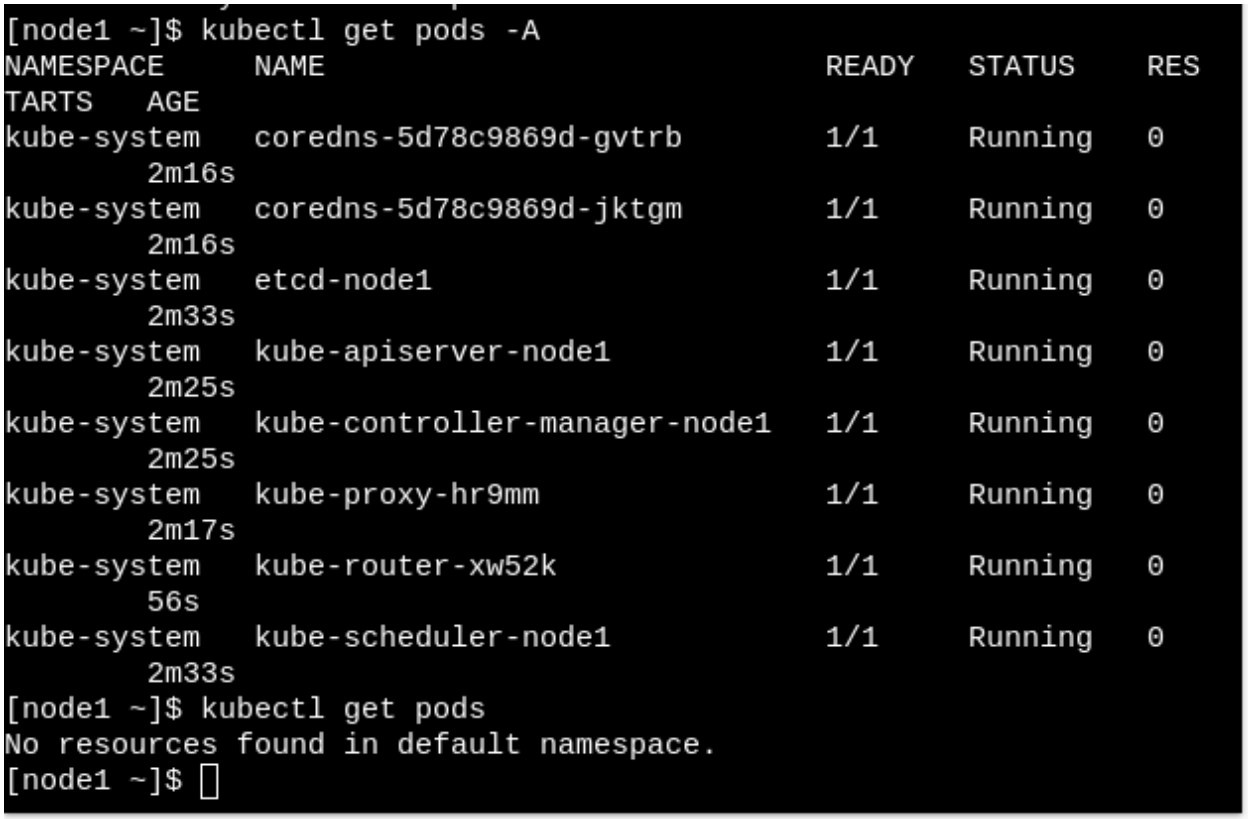
а затем

kubectl get pods

В результате мы увидим полный (-A) список подов запущенных в кластере и только те поды, которые находятся в пространстве имён "default". В первом

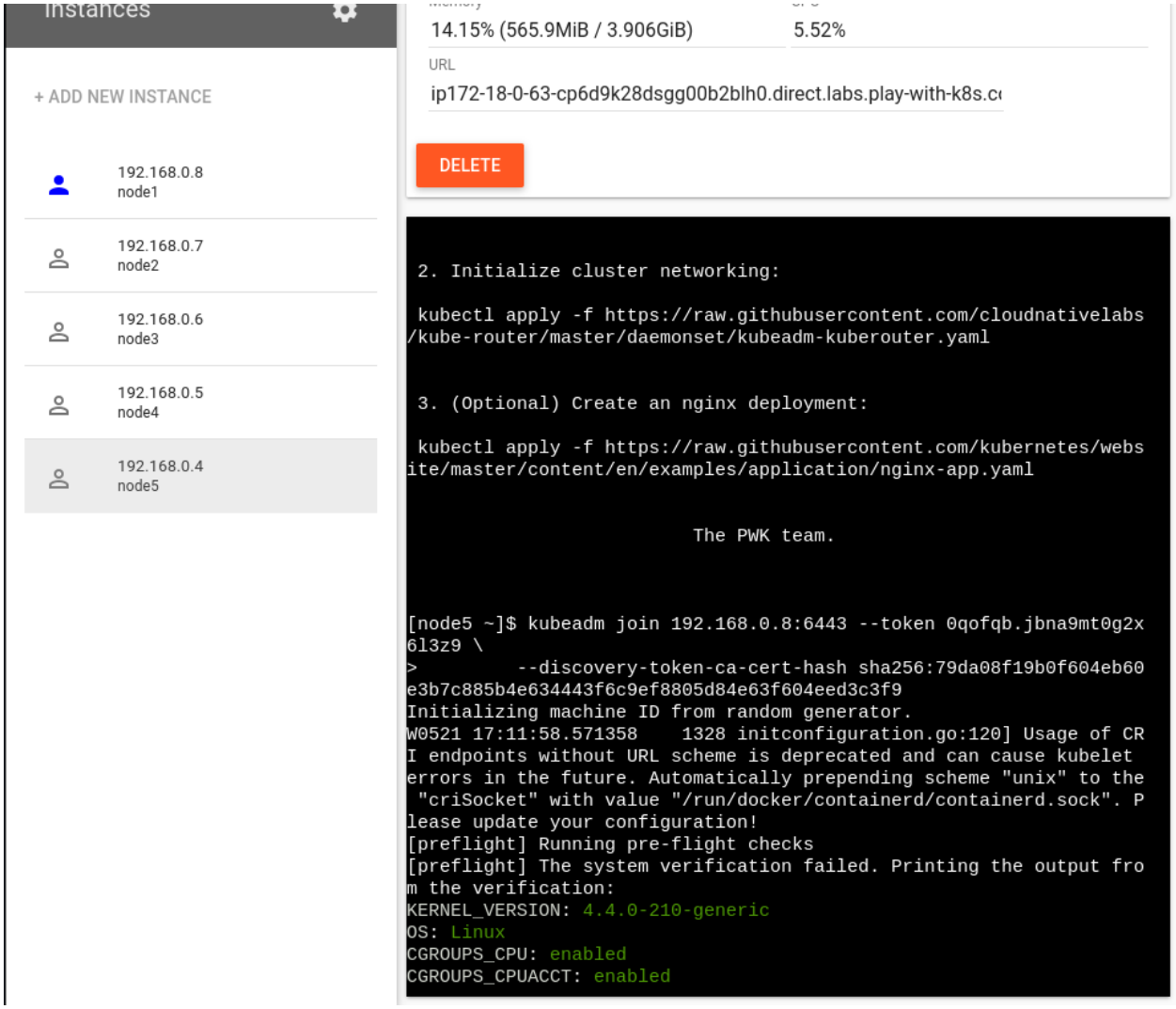
приближении поды можно воспринимать как аналог процессов на обычном компьютере.

Все поды распределяются по пространствам имён (namespaces) полный список которых можно посмотреть при помощи команды: kubectl get namespaces. Если команды выполняется без явного указания пространства имён (ключ --namespace), то подразумевается пространство имён "default"



8) Подключим остальные сервера к кластеру.

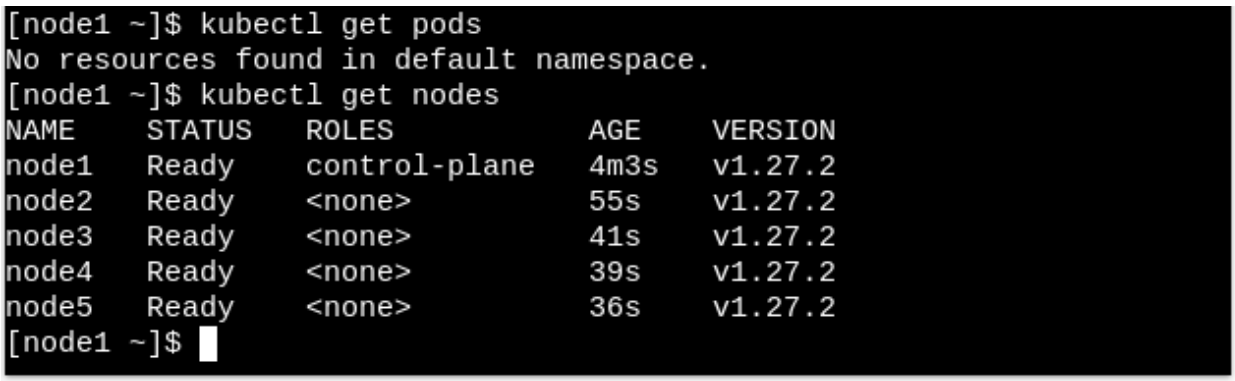
Для этого на оставшихся 4-х нодах запустим команду kubeadm join с параметрами, которые мы копировали ранее в текстовый файл.



9) Перейдём на мастер-ноду и снова выполним команду:

kubectl get nodes

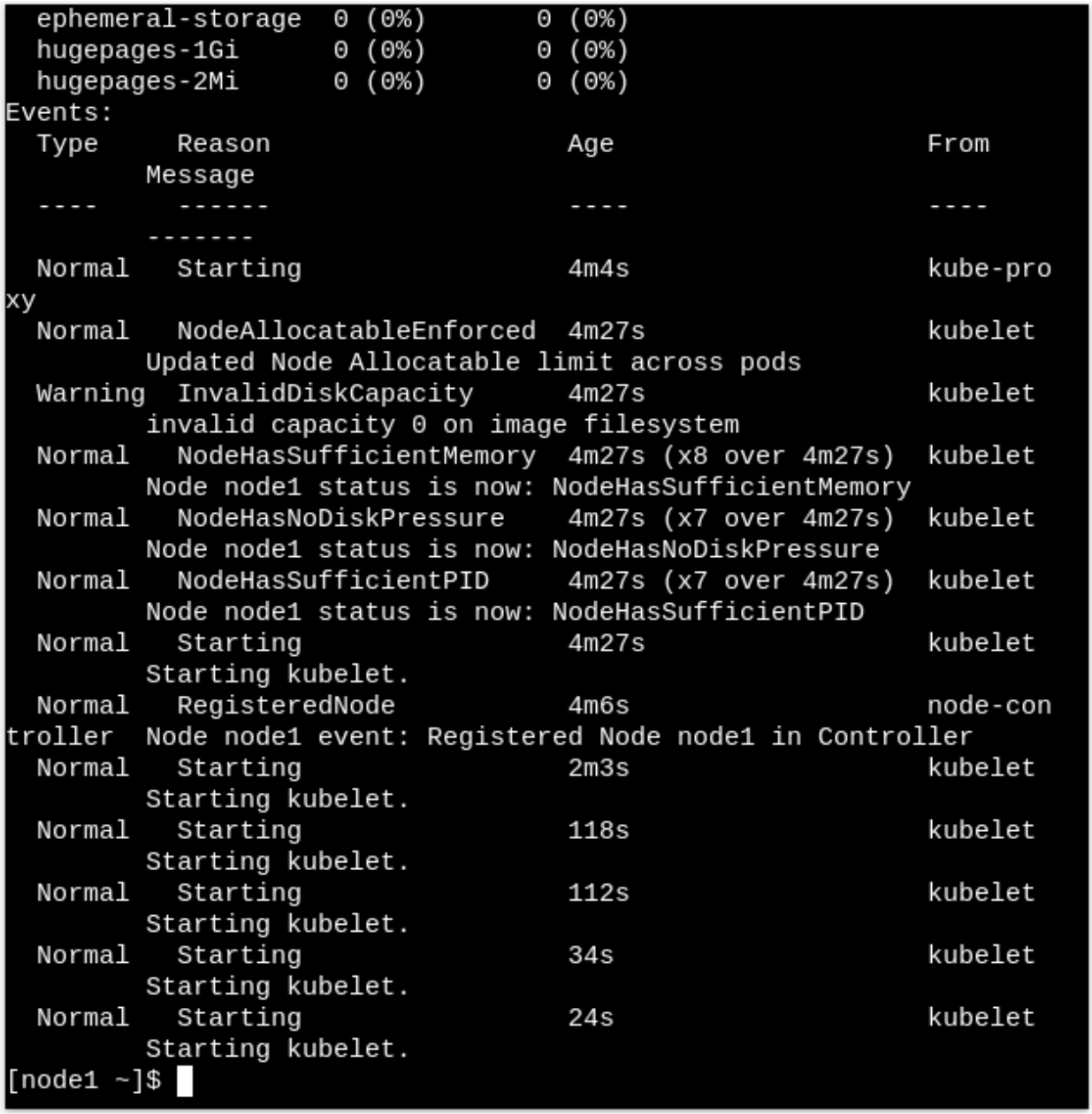
В результате видим, что в списке теперь 5 нод



10) Выполним команду:

kubectl describe nodes node1

Данная команда позволит нам получить подробную информацию по ноде с именем "node1"



11) Выполним команду:

kubectl get pods -o wide -A

В результате получим список всех подов с расширенной информацией по каждому. Обратим внимание, что некоторые поды автоматически запустились на новых нодах

```
TARTS   AGE     IP           NODE    NOMINATED NODE   READINESS GA
TES
kube-system   coredns-5d78c9869d-gvtrb        1/1     Running   0
        4m15s   10.5.0.3     node1   <none>           <none>
kube-system   coredns-5d78c9869d-jktgm        1/1     Running   0
        4m15s   10.5.0.4     node1   <none>           <none>
kube-system   etcd-node1                      1/1     Running   0
        4m32s   192.168.0.8  node1   <none>           <none>
kube-system   kube-apiserver-node1            1/1     Running   0
        4m24s   192.168.0.8  node1   <none>           <none>
kube-system   kube-controller-manager-node1   1/1     Running   0
        4m24s   192.168.0.8  node1   <none>           <none>
kube-system   kube-proxy-7fs2j                1/1     Running   0
        85s     192.168.0.7  node2   <none>           <none>
kube-system   kube-proxy-dr998                1/1     Running   0
        71s     192.168.0.6  node3   <none>           <none>
kube-system   kube-proxy-hr9mm                1/1     Running   0
        4m16s   192.168.0.8  node1   <none>           <none>
kube-system   kube-proxy-l4hzb                1/1     Running   0
        69s     192.168.0.5  node4   <none>           <none>
kube-system   kube-proxy-qh5rg                1/1     Running   0
        66s     192.168.0.4  node5   <none>           <none>
kube-system   kube-router-4bfbc               1/1     Running   0
        69s     192.168.0.5  node4   <none>           <none>
kube-system   kube-router-n6cqr               1/1     Running   0
        66s     192.168.0.4  node5   <none>           <none>
kube-system   kube-router-x698t               1/1     Running   0
        71s     192.168.0.6  node3   <none>           <none>
kube-system   kube-router-xw52k               1/1     Running   0
        2m55s   192.168.0.8  node1   <none>           <none>
kube-system   kube-router-zqxlp               1/1     Running   0
        85s     192.168.0.7  node2   <none>           <none>
kube-system   kube-scheduler-node1            1/1     Running   0
        4m32s   192.168.0.8  node1   <none>           <none>
[node1 ~]$
```

12) Выполните команду:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml

В результате будет запущено 3 пода с сервером "nginx". Проверим это при помощи команды kubectl get pods -o wide. В столбце "Status" указано "Running"

```
[node1 ~]$ kubectl apply -f https://raw.githubusercontent.com/kuber
netes/website/master/content/en/examples/application/nginx-app.yaml
service/my-nginx-svc created
deployment.apps/my-nginx created
[node1 ~]$ kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP
      NODE    NOMINATED NODE   READINESS GATES
my-nginx-cbdccf466-p4gkt    1/1     Running   0          45s   10.5.
2.2   node3   <none>           <none>
my-nginx-cbdccf466-qss4t    1/1     Running   0          45s   10.5.
1.2   node2   <none>           <none>
my-nginx-cbdccf466-z2nzx    1/1     Running   0          45s   10.5.
3.2   node4   <none>           <none>
[node1 ~]$
```

13) Обычно поды "заперты" во внутренней сети кластера и снаружи к ним доступа нет. Но командой выше мы не только запустили 3 пода с "nginx", но и создали элемент кластера "Service", который пробросил некоторый внешний порт на 80й порт внутри кластера. Т.к. "nginx" - это веб-сервер то он слушает как раз 80й порт. Определим, какой же порт проброшен.
Для этого введём команду:

kubectl get svc

Эта команда покажет список элементов типа "Service". Найдём в нём сервис типа "LoadBalancer" с названием "my-nginx-svc" и в столбце "Port" определите внешний порт (у меня это 31231)

```
[node1 ~]$ kubectl get svc
NAME           TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
       AGE
kubernetes     ClusterIP      10.96.0.1       <none>        443/TCP
       5m44s
my-nginx-svc   LoadBalancer   10.106.49.248   <pending>     80:3097
7/TCP   62s
[node1 ~]$
```

14) Проверим, что "nginx" работает при помощи утилиты curl:

curl 192.168.0.18:31231

Здесь 192.168.0.18 - это ip-адрес моей мастер-ноды, а 31231 - проброшенный внутрь порт.

В результате видим код html-страницы с приветствием от "nginx"

```
[node1 ~]$ curl 192.168.0.8:30977
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully insta
lled and
working. Further configuration is required.</p>

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

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[node1 ~]$
```

15) Отключим одну или несколько нод на которых работают поды с "nginx". Определим на каких нодах запущены поды с "nginx" (у меня это node4, node5, node2).

Введём команду (со своим именем ноды):

kubectl drain node4 --ignore-daemonsets --delete-local-data

Данная команда проинформирует ноду о том, что нужно завершить все свои поды и в результате они будут автоматически запущены на оставшихся нодах.

```
[node1 ~]$ kubectl drain node4 --ignore-daemonsets --delete-local-d
ata
Flag --delete-local-data has been deprecated, This option is deprec
ated and will be deleted. Use --delete-emptydir-data.
node/node4 cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-l4
hzb, kube-system/kube-router-4bfbc
evicting pod default/my-nginx-cbdccf466-z2nzx
pod/my-nginx-cbdccf466-z2nzx evicted
node/node4 drained
[node1 ~]$ kubectl drain node3 --ignore-daemonsets --delete-local-d
ata
Flag --delete-local-data has been deprecated, This option is deprec
ated and will be deleted. Use --delete-emptydir-data.
node/node3 cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-dr
998, kube-system/kube-router-x698t
evicting pod default/my-nginx-cbdccf466-p4gkt
pod/my-nginx-cbdccf466-p4gkt evicted
node/node3 drained
[node1 ~]$ kubectl drain node2 --ignore-daemonsets --delete-local-d
ata
Flag --delete-local-data has been deprecated, This option is deprec
ated and will be deleted. Use --delete-emptydir-data.
node/node2 cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-7f
s2j, kube-system/kube-router-zqxlp
evicting pod default/my-nginx-cbdccf466-qss4t
pod/my-nginx-cbdccf466-qss4t evicted
node/node2 drained
[node1 ~]$
```

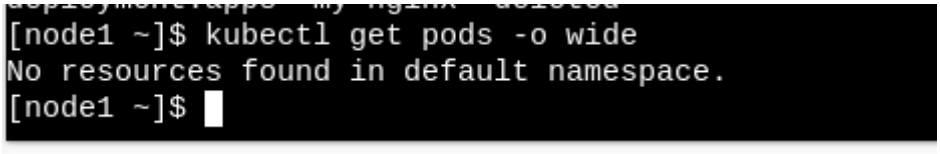
16) Остановим запущенное приложение "nginx". Для этого выполним команду запуска, но вместо applay укажем delete, т.е.:

kubectl delete –f

https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml

```
[node1 ~]$ kubectl delete -f https://raw.githubusercontent.com/kube
rnetes/website/master/content/en/examples/application/nginx-app.yam
l
service "my-nginx-svc" deleted
deployment.apps "my-nginx" deleted
[node1 ~]$
```

17) Проверим список запущенных подов - он пуст



Вопросы:

1. Чтобы узнать IP-адрес ноды по её имени в кластере, вы можете воспользоваться командой `kubectl get pods -o wide`, если используете Kubernetes. Эта команда покажет IP-адреса всех подов в вашем кластере, включая их имена.

2. Предварительный этап для запуска приложения из исходных кодов в кластере может включать настройку среды разработки, установку необходимых зависимостей и инструментов сборки, а также создание манифестов Kubernetes для вашего приложения.

3. Для подключения рабочей ноды к кластеру нужно знать IP-адреса мастер-нод, учетные данные для аутентификации (например, сертификаты или токены доступа) и соответствующие разрешения (RBAC), если они используются в вашем кластере.

4. Да, в одном кластере может быть несколько мастер-нод. Это обеспечивает отказоустойчивость и распределение нагрузки.

5. Да, кластер может состоять только из мастер-ноды, которая одновременно является и рабочей нодой. Это часто используется в небольших или тестовых средах, где нет необходимости в выделенных рабочих нодах. Однако в продукционных средах обычно используется не менее двух мастер-нод для обеспечения отказоустойчивости.

6.