

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
Аметов Кемран Ленверович

направления подготовки 09.03.04 «Программная инженерия»

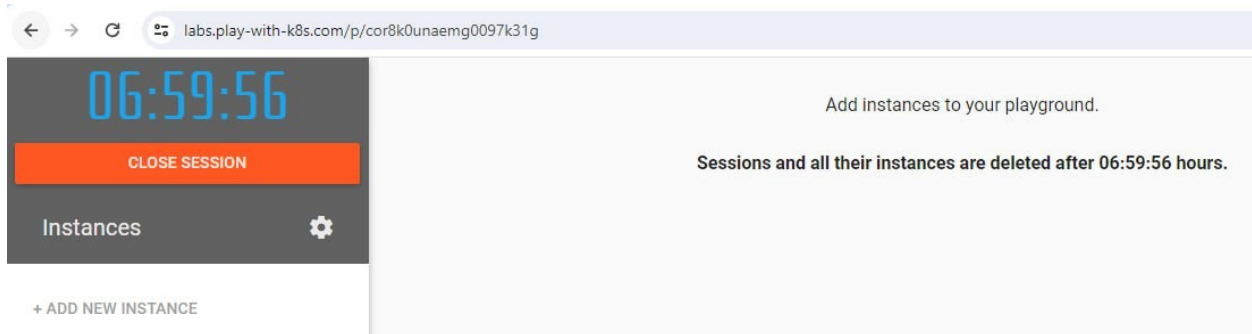
Симферополь, 2024

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

Ход выполнения задания:

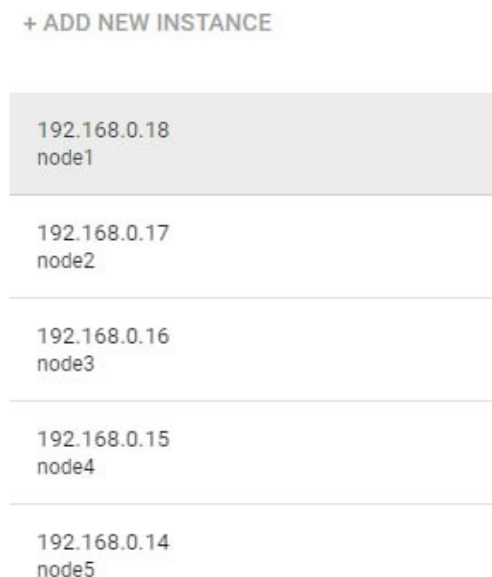
Объединение серверов в кластер

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) иницииируют 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-нодой (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).

```
[node1 ~]$ kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16
Initializing machine ID from random generator.
W0504 20:09:07.511971    780 initconfiguration.go:120] Usage of CRI endpoints without URL scheme is deprecated a
nd can cause kubelet errors in the future. Automatically prepending scheme "unix" to the "criSocket" with value "
/run/docker/containerd/containerd.sock". Please update your configuration!
I0504 20:09:07.850134    780 version.go:256] remote version is much newer: v1.30.0; falling back to: stable-1.27
[init] Using Kubernetes version: v1.27.13
[preflight] Running pre-flight checks
[WARNING Swap]: swap is enabled; production deployments should disable swap unless testing the NodeSwap f
eature gate of the kubelet
[preflight] The system verification failed. Printing the output from the verification:
KERNEL_VERSION: 4.4.0-210-generic
OS: Linux
CGROUPS_CPU: enabled
CGROUPS_CPUACCT: enabled
```

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"

```
[node1 ~]$ kubectl get pods -A
NAMESPACE      NAME                                     READY   STATUS    RES
TARTS   AGE
kube-system    coredns-5d78c9869d-gvtrb              1/1     Running   0
          2m16s
kube-system    coredns-5d78c9869d-jktgm              1/1     Running   0
          2m16s
kube-system    etcd-node1                             1/1     Running   0
          2m33s
kube-system    kube-apiserver-node1                  1/1     Running   0
          2m25s
kube-system    kube-controller-manager-node1         1/1     Running   0
          2m25s
kube-system    kube-proxy-hr9mm                      1/1     Running   0
          2m17s
kube-system    kube-router-xw52k                     1/1     Running   0
          56s
kube-system    kube-scheduler-node1                  1/1     Running   0
          2m33s
[node1 ~]$ kubectl get pods
No resources found in default namespace.
[node1 ~]$
```

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.

Instances

+ ADD NEW INSTANCE

	192.168.0.8 node1
	192.168.0.7 node2
	192.168.0.6 node3
	192.168.0.5 node4
	192.168.0.4 node5

14.15% (565.9MiB / 3.906GiB) 5.52%

URL
ip172-18-0-63-cp6d9k28dsgg00b2blh0.direct.labs.play-with-k8s.c

DELETE

2. Initialize cluster networking:

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

3. (Optional) Create an nginx deployment:

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

The PWK team.

```
[node5 ~]$ kubeadm join 192.168.0.8:6443 --token 0qofqb.jbna9mt0g2x6l3z9 \
> --discovery-token-ca-cert-hash sha256:79da08f19b0f604eb60e3b7c885b4e634443f6c9ef8805d84e63f604eed3c3f9
Initializing machine ID from random generator.
W0521 17:11:58.571358 1328 initconfiguration.go:120] Usage of CR
I endpoints without URL scheme is deprecated and can cause kubelet
errors in the future. Automatically prepending scheme "unix" to the
"criSocket" with value "/run/docker/containerd/containerd.sock". P
lease update your configuration!
[preflight] Running pre-flight checks
[preflight] The system verification failed. Printing the output fro
m the verification:
KERNEL_VERSION: 4.4.0-210-generic
OS: Linux
CGROUPS_CPU: enabled
CGROUPS_CPUACCT: enabled
```

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

kubectl get nodes

В результате видим, что в списке теперь 5 нод

```
[node1 ~]$ kubectl get pods
No resources found in default namespace.
[node1 ~]$ kubectl get nodes
NAME      STATUS    ROLES          AGE      VERSION
node1     Ready     control-plane   4m3s     v1.27.2
node2     Ready     <none>          55s      v1.27.2
node3     Ready     <none>          41s      v1.27.2
node4     Ready     <none>          39s      v1.27.2
node5     Ready     <none>          36s      v1.27.2
[node1 ~]$
```

10) Выполним команду:

kubectl describe nodes node1

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

```

ephemeral-storage 0 (0%)      0 (0%)
hugepages-1Gi     0 (0%)      0 (0%)
hugepages-2Mi     0 (0%)      0 (0%)
Events:
  Type            Reason              Age             From
                Message
  ----            -
Normal          Starting            4m4s           kube-pro
xy
Normal          NodeAllocatableEnforced 4m27s          kubelet
                Updated Node Allocatable limit across pods
Warning         InvalidDiskCapacity  4m27s          kubelet
                invalid capacity 0 on image filesystem
Normal          NodeHasSufficientMemory 4m27s (x8 over 4m27s) kubelet
                Node node1 status is now: NodeHasSufficientMemory
Normal          NodeHasNoDiskPressure  4m27s (x7 over 4m27s) kubelet
                Node node1 status is now: NodeHasNoDiskPressure
Normal          NodeHasSufficientPID   4m27s (x7 over 4m27s) kubelet
                Node node1 status is now: NodeHasSufficientPID
Normal          Starting            4m27s          kubelet
                Starting kubelet.
Normal          RegisteredNode        4m6s           node-con
troller Node node1 event: Registered Node node1 in Controller
Normal          Starting            2m3s           kubelet
                Starting kubelet.
Normal          Starting            118s           kubelet
                Starting kubelet.
Normal          Starting            112s           kubelet
                Starting kubelet.
Normal          Starting            34s            kubelet
                Starting kubelet.
Normal          Starting            24s            kubelet
                Starting kubelet.
[node1 ~]$ █

```

11) Выполним команду:

kubectll 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

NAME	AGE	IP	NODE	NOMINATED NODE	READINESS	GATES
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>	

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/kubernetes/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	NODE	NOMINATED NODE	READY	STATUS	RESTARTS	AGE	IP
my-nginx-cbdccf466-p4gkt	node3	<none>	1/1	Running	0	45s	10.5.2.2
my-nginx-cbdccf466-qss4t	node2	<none>	1/1	Running	0	45s	10.5.1.2
my-nginx-cbdccf466-z2nzx	node4	<none>	1/1	Running	0	45s	10.5.3.2

```
[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	AGE	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
kubernetes	5m44s	ClusterIP	10.96.0.1	<none>	443/TCP
my-nginx-svc	62s	LoadBalancer	10.106.49.248	<pending>	80:30977/TCP

```
[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 installed 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-14
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 apply 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

```
[node1 ~]$ kubectl get pods -o wide  
No resources found in default namespace.  
[node1 ~]$
```

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 производственных средах обычно используется не менее двух мастер-нод для обеспечения отказоустойчивости.