МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
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-233
Иващенко Дениса Олеговича

направления подготовки 09.03.04 «Программная инженерия»

Симферополь, 2024

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

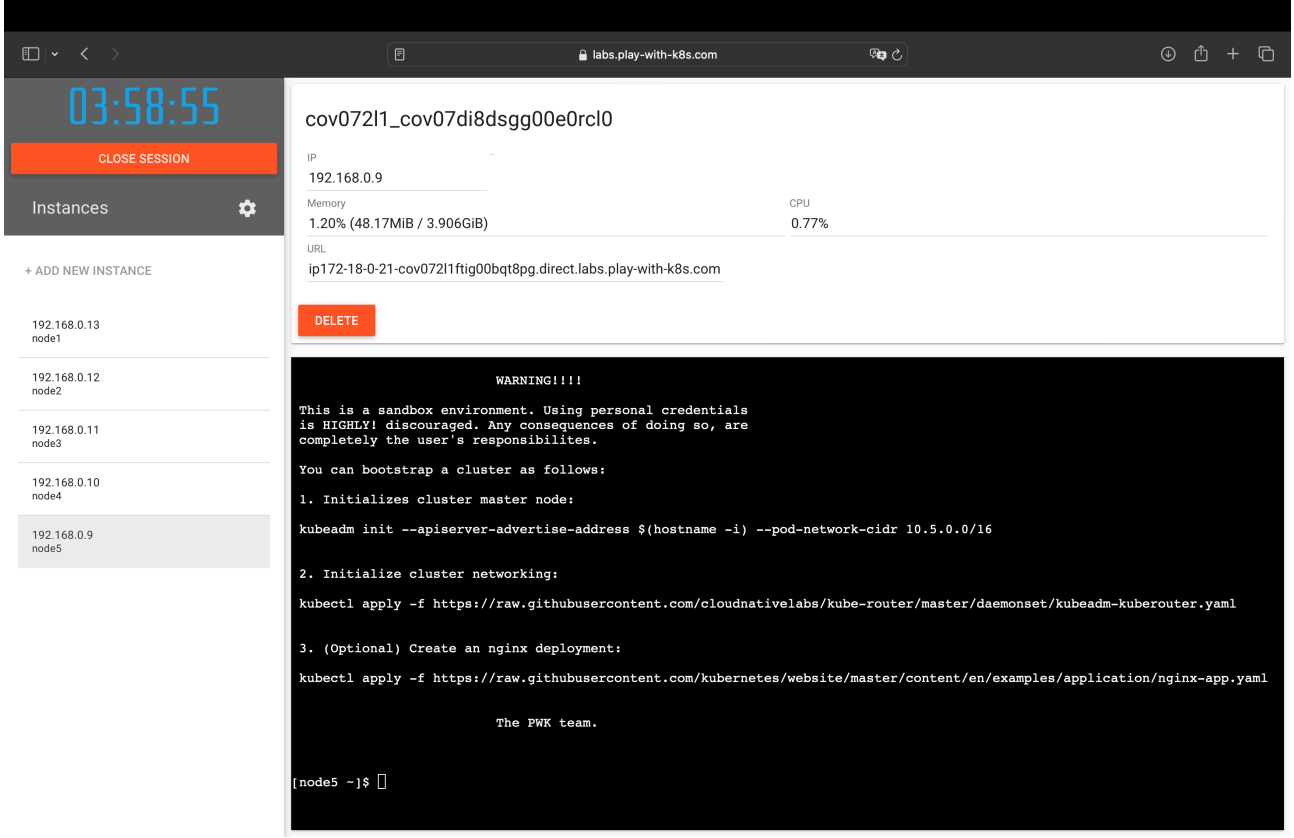
Ход выполнения работы:

Объединение серверов в кластер

1)Перейдем на сайт https://labs.play-with-k8s.com/ и залогиньтесь при помощи учётной записи gitHub или dockerHub.

2)После того, как мы нажмём на кнопку "Start" для вас будет создана 4-х часовая сессия, в пределах которой можно будет создать до 5-ти серверов (node) с уже установленными компонентами kubernetes.

Создайте 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 вы увидите 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.
kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16
Здесь же будет показана строка которую нужно будет запускать на рабочих узлах для подключения к кластеру.

```
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node node1 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node node1 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: llbags.07mlw6tn5fdtssfs
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.13:6443 --token llbags.07mlw6tn5fdtssfs \
        --discovery-token-ca-cert-hash sha256:5ef490c23b98d14796bb21920760ab9a3ce374f888fef2d9e689992453e474e0
Waiting for api server to startup
Warning: resource daemonsets/kube-proxy is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
daemonset.apps/kube-proxy configured
No resources found
[node1 ~]$
```

На мастер-ноде запустите команду:

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

```
[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 ~]$
```

6) Выполните команду:

kubectl get nodes

В результате вы увидите 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   7m43s   v1.27.2
```

7)Выполните команду:

kubectl get pods -A

а затем

kubectl get pods

В результате вы увидите 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    RESTARTS   AGE
kube-system   coredns-5d78c9869d-gfz7b            1/1     Running   0          8m17s
kube-system   coredns-5d78c9869d-k85sk            1/1     Running   0          8m17s
kube-system   etcd-node1                          1/1     Running   0          8m25s
kube-system   kube-apiserver-node1                1/1     Running   0          8m25s
kube-system   kube-controller-manager-node1       1/1     Running   0          8m32s
kube-system   kube-proxy-8bsrb                    1/1     Running   0          8m17s
kube-system   kube-router-sgd5x                   1/1     Running   0          107s
kube-system   kube-scheduler-node1                1/1     Running   0          8m24s
[node1 ~]$ kubectl get pods
No resources found in default namespace.
[node1 ~]$ kubectl get pods
No resources found in default namespace.
[node1 ~]$
```

8)Подключим остальные сервера к кластеру.Для этого на оставшихся 4-х нодах запустите команду kubeadm join с параметрами, которые вы копировали ранее в текстовый файл.

```
[node5 ~]$ kubeadm join 192.168.0.13:6443 --token llbags.07mlw6tn5fdtssfs \
>         --discovery-token-ca-cert-hash sha256:5ef490c23b98d14796bb21920760ab9a3ce374f888fef2d9e689992453e474e0
Initializing machine ID from random generator.
W0510 11:42:02.148940    3960 initconfiguration.go:120] Usage of CRI endpoints without URL scheme is deprecated and can cause kubelet errors in the future. Automatically prepe
nding scheme "unix" to the "criSocket" with value "/run/docker/containerd/containerd.sock". Please update your configuration!
[preflight] Running pre-flight checks
[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
CGROUPS_CPUSET: enabled
CGROUPS_DEVICES: enabled
CGROUPS_FREEZER: enabled
CGROUPS_MEMORY: enabled
CGROUPS_PIDS: enabled
CGROUPS_HUGETLB: enabled
CGROUPS_BLKIO: enabled
        [WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit status 1
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[node5 ~]$
```

9)Перейдите на мастер-ноду и снова выполните команду:

kubectl get nodes

В результате вы должны увидеть, что в списке теперь 5 нод.

```
[node1 ~]$ kubectl get nodes
NAME     STATUS     ROLES           AGE     VERSION
node1    Ready      control-plane   13m     v1.27.2
node2    NotReady   <none>          2m46s   v1.27.2
node3    NotReady   <none>          2m19s   v1.27.2
node4    NotReady   <none>          84s     v1.27.2
node5    NotReady   <none>          60s     v1.27.2
[node1 ~]$
```

10) Выполните команду:

kubectl describe nodes node1

Данная команда позволит вам получить подробную информацию по ноде с именем «node1».

```
[node1 ~]$ kubectl describe nodes node1
Name:               node1
Roles:              control-plane
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node1
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/control-plane=
                    node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///run/docker/cont
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Fri, 10 May 2024 11:29:56 +0000
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  node1
  AcquireTime:     <unset>
  RenewTime:       Fri, 10 May 2024 11:44:06 +0000
```

11)Выполните команду:

kubectl get pods -o wide -A

В результате вы должны получить список всех подов с расширенной информацией по каждому. Обратите внимание, что некоторые поды автоматически запустились на новых нодах.

```
[node1 ~]$ kubectl get pods -o wide -A
NAMESPACE     NAME                              READY   STATUS    RESTARTS       AGE     IP              NODE    NOMINATED NODE   READINESS GATES
kube-system   coredns-5d78c9869d-gfz7b          1/1     Running   0              14m     10.5.0.3        node1   <none>           <none>
kube-system   coredns-5d78c9869d-k85sk          1/1     Running   0              14m     10.5.0.2        node1   <none>           <none>
kube-system   etcd-node1                        1/1     Running   0              15m     192.168.0.13    node1   <none>           <none>
kube-system   kube-apiserver-node1              1/1     Running   0              15m     192.168.0.13    node1   <none>           <none>
kube-system   kube-controller-manager-node1     1/1     Running   1 (4m6s ago)   15m     192.168.0.13    node1   <none>           <none>
kube-system   kube-proxy-8bsrb                  1/1     Running   0              14m     192.168.0.13    node1   <none>           <none>
kube-system   kube-proxy-dl7mg                  1/1     Running   0              3m24s   192.168.0.10    node4   <none>           <none>
kube-system   kube-proxy-dtkz7                  1/1     Running   0              4m46s   192.168.0.12    node2   <none>           <none>
kube-system   kube-proxy-tr796                  1/1     Running   0              4m19s   192.168.0.11    node3   <none>           <none>
kube-system   kube-proxy-xs5bw                  1/1     Running   0              2m59s   192.168.0.9     node5   <none>           <none>
kube-system   kube-router-9fk7j                 1/1     Running   0              2m59s   192.168.0.9     node5   <none>           <none>
kube-system   kube-router-nk8st                 1/1     Running   0              4m19s   192.168.0.11    node3   <none>           <none>
kube-system   kube-router-sgd5x                 1/1     Running   1 (3m4s ago)   8m29s   192.168.0.13    node1   <none>           <none>
kube-system   kube-router-st2jf                 1/1     Running   0              4m46s   192.168.0.12    node2   <none>           <none>
kube-system   kube-router-wghzc                 1/1     Running   0              3m24s   192.168.0.10    node4   <none>           <none>
kube-system   kube-scheduler-node1              1/1     Running   1 (3m3s ago)   15m     192.168.0.13    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". Проверьте это при помощи команды kubectl get pods -o wide. В столбце "Status" должно быть указано «Running».(в моем случае в столбце «Status» указано «Pending» из-за нестабильной работы сервиса).

```
[node1 ~]$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml
service/my-nginx-svc unchanged
deployment.apps/my-nginx unchanged
[node1 ~]$ kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP       NODE     NOMINATED NODE   READINESS GATES
my-nginx-cbdccf466-9q4wb    0/1     Pending   0          2m9s    <none>   <none>   <none>           <none>
my-nginx-cbdccf466-djljs    0/1     Pending   0          2m9s    <none>   <none>   <none>           <none>
my-nginx-cbdccf466-jcpp4    0/1     Pending   0          2m10s   <none>   <none>   <none>           <none>
```

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.

kubectl get svc

Эта команда покажет вам список элементов типа "Service". Найдите в нём сервис типа "LoadBalancer" с названием "my-nginx-svc" и в столбце "Port" определите внешний порт (в моем случае это 32723)

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

```
[node1 ~]$ curl 192.168.0.13:32723
curl: (7) Failed connect to 192.168.0.13:32723; Connection refused
[node1 ~]$ curl 10.106.147.227:32723
```

15)Отключим одну или несколько нод на которых работают поды с "nginx". Процесс корректного отключения ноды от реального кластера может содержать больше шагов, чем приведены здесь:

1.Определите на каких нодах запущены поды с «nginx"

2.Введите команду (со своим именем ноды):

```
[node1 ~]$ kubectl drain node3 --ignore-daemonsets --delete-local-data
Flag --delete-local-data has been deprecated, This option is deprecated and will be deleted. Use --delete-emptydir-data.
node/node3 cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-d4zn9, kube-system/kube-router-85z4s
node/node3 drained
[node1 ~]$
```

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

3.Проверьте список доступных нод в кластере.

```
[node1 ~]$ kubectl get nodes
NAME    STATUS                        ROLES           AGE   VERSION
node1   Ready                         control-plane   41m   v1.27.2
node2   NotReady                      <none>          40m   v1.27.2
node3   NotReady,SchedulingDisabled   <none>          40m   v1.27.2
node4   NotReady                      <none>          40m   v1.27.2
node5   NotReady                      <none>          39m   v1.27.2
```

16)Остановим запущенное приложение "nginx". Для этого выполните команду запуска, но вместо applay укажите 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/kubernetes/website/master/content/en/examples/application/nginx-app.yaml
service "my-nginx-svc" deleted
deployment.apps "my-nginx" deleted
```

17)Проверьте список запущенных подов - он должен быть пустым.

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

Вывод: сегодня на практическом занятии по предмету «Современные технологии программирования» я ознакомился на практике с инструментом оркестрации контейнеризированных приложений Kubernetes.

Вопросы к практическому заданию

1)Как узнать ip-адрес ноды подключённой к кластеру по её имени?

Для того чтобы узнать IP-адрес ноды по её имени в кластере Kubernetes, можно воспользоваться командой kubectl get nodes -o wide. Эта команда покажет список всех нод в вашем кластере вместе с их IP-адресами.

2)Какой предварительный этап нужно выполнить, чтобы иметь возможность запустить приложение в кластере из исходников? Считаем, что кластер уже собран.

Прежде всего, необходимо настроить среду для разработки и сборки вашего приложения. Далее, нужно создать Docker-образ вашего приложения и загрузить его в репозиторий контейнеров (например, Docker Hub). Затем, вы можете создать манифесты Kubernetes (например, Deployment) для запуска вашего приложения в кластере.

3)Что нужно знать, чтобы подключить рабочую ноду к кластеру?

Для подключения рабочей ноды к кластеру Kubernetes, необходимо иметь доступ к кластеру (обычно через kubectl) и правильно настроить конфигурацию рабочей ноды для связи с мастер-нодами.

4)Может ли быть в одном кластере несколько мастер-нод?

Да, в Kubernetes можно иметь несколько мастер-нод для обеспечения отказоустойчивости и балансировки нагрузки. При этом, только одна из мастер-нод будет активной в определенный момент времени.

5)Может ли существовать кластер состоящий только из мастер ноды, которая одновременно является и рабочей нодой?

Технически это возможно, однако это не рекомендуется в продакшене из-за риска единой точки отказа. В реальной среде лучше иметь отдельные мастер- и рабочие ноды для лучшей отказоустойчивости и масштабируемости.