使用 kubeadm 在 CentOS 7.9 部署 K8s 集群

作者：lingyundu

原文链接：https://ld246.com/article/1647921788023

来源网站：链滴

集群规划

使用 VMware® Workstation 16 Pro 创建三台虚拟机。

主机名	IP	角色	操作系统
node1.k8s.com	192.168.153.21	master	entOS 7.9-Minimal
node2.k8s.com	192.168.153.22	worker	entOS 7.9-Minimal
node3.k8s.com	192.168.153.23	worker	entOS 7.9-Minimal

环境配置

每台虚拟机都要配置，使用 root 用户。

1、关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

2、同步时间

```
yum -y install ntpdate
ntpdate ntp5.aliyun.com
```

3、配置主机名

```
vi /etc/hostname
```

按照规划，将三台虚拟机的主机名分别设置为：node1.k8s.com、node2.k8s.com 和 node3.k8s.com。

4、配置 hosts 文件

```
vi /etc/hosts
```

添加下面的内容：

```
192.168.153.21 node1 node1.k8s.com
192.168.153.22 node2 node1.k8s.com
192.168.153.23 node3 node1.k8s.com
```

5、禁用 Swap

Kubernetes v1.8+ 要求关闭系统 Swap，否则 kubelet 可能工作不正常（kubeadm 初始化系统时会检查 swap 是否关闭）

```
swapoff -a && sed -i '/ swap / s/^/#/' /etc/fstab
```

6、允许 iptables 检查桥接流量

确保 br_netfilter 模块被加载：

```
cat <<EOF | tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

modprobe br_netfilter
```

将 net.bridge.bridge-nf-call-iptables 设置为 1：

```
cat <<EOF | tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sysctl --system
```

7、禁用 SELinux

将 SELinux 设置为 permissive 模式（相当于将其禁用）

```
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

8、创建用户并添加到 wheel 组

```
useradd k8s
passwd k8s
usermod -aG wheel k8s
```

安装 Docker 和 kubeadm

Kubernetes 支持多种容器运行时（Container Runtime），我这里将会安装 Docker。

从 v1.14.0 开始， kubeadm 会自动检测 Linux 节点的容器运行环境，该检测基于 Domain Socket。

三种运行时及其对应的 Domain Socket：

Runtime	Domain Socket
Docker	/var/run/docker.sock
contained	/run/containerd/containerd.sock
CRI0-O	/var/run/crio/crio.sock

如果同时检测到 Docker 和 containerd，则优先采用 Docker。

安装 Docker

每个节点都要安装，使用 k8s 用户。

1、安装 yum-utils，并设置 Docker 的 stable 仓库

```
sudo yum install -y yum-utils
sudo yum-config-manager \
    --add-repo \
```

https://download.docker.com/linux/centos/docker-ce.repo

2、安装 Docker 最新版本

```
sudo yum -y install docker-ce docker-ce-cli containerd.io
```

确认安装成功：

```
[k8s@node1 ~]$ docker --version
Docker version 20.10.13, build a224086
```

3、配置 cgroupfs 驱动

需要确保容器运行时和 kubelet 所使用的是相同的 cgroup 驱动，否则 kubelet 进程会失败。

由于 kubeadm 把 kubelet 视为一个系统服务来管理，所以对基于 kubeadm 的安装， 官方推荐使用 systemd 驱动，不推荐 cgroupfs 驱动。

新建 /etc/docker/daemon.json 文件：

```
sudo vi /etc/docker/daemon.json
```

在文件中添加下面的启动项参数：

```
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

4、启用并启动 Docker

```
sudo systemctl enable --now docker
```

5、确认 Docker 启动成功

```
[k8s@node1 docker]$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-03-22 08:43:14 CST; 24s ago
     Docs: https://docs.docker.com
 Main PID: 12340 (dockerd)
    Tasks: 9
   Memory: 34.2M
   CGroup: /system.slice/docker.service
           └─12340 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.368688243+08:
0" level=info msg="ccResolverWrapper: sending update to cc: {[{unix:///run/co...odule=grpc
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.368707820+08:
0" level=info msg="ClientConn switching balancer to \"pick_first\"" module=grpc
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.376663870+08:
0" level=info msg="[graphdriver] using prior storage driver: overlay2"
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.378856509+08:
0" level=info msg="Loading containers: start."
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.485042302+08:
0" level=info msg="Default bridge (docker0) is assigned with an IP address 17...P address"
```

```
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.526021249+08:
0" level=info msg="Loading containers: done."
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.544232229+08:
0" level=info msg="Docker daemon" commit=906f57f graphdriver(s)=overlay2 version=20.10
13
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.544300079+08:
0" level=info msg="Daemon has completed initialization"
Mar 22 08:43:14 node1.k8s.com systemd[1]: Started Docker Application Container Engine.
Mar 22 08:43:14 node1.k8s.com dockerd[12340]: time="2022-03-22T08:43:14.562599360+08:
0" level=info msg="API listen on /var/run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
[k8s@node1 docker]$
```

6、确认 Cgroup 驱动配置成功

```
[k8s@node1 docker]$ sudo docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.8.0-docker)
  scan: Docker Scan (Docker Inc., v0.17.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.13
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 1
.....
```

安装 kubeadm、kubelet 和 kubectl

Kubeadm 是社区推出的一个用来部署 Kubernetes 集群的工具。

Kubelet 主要负责同容器运行时（比如：Docker）打交道，它是 Kubernetes 集群的核心组件。

Kubectl 则是 Kubernetes 集群的命令行工具。

每个节点都要安装，使用 k8s 用户。

1、配置 Yum 源

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
```

```
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.
om/kubernetes/yum/doc/rpm-package-key.gpg
EOF
```

此处使用的是阿里云的 Kubernetes 镜像。

2、安装

```
sudo yum install -y --nogpgcheck kubelet kubeadm kubectl
```

由于 Kubernetes 官网未开放同步方式, 使用阿里云镜像可能会有索引 gpg 检查失败的情况，此处指定 -nogpgcheck 参数跳过了检查。

3、启用并启动 kubelet

```
sudo systemctl enable --now kubelet
```

4、查看 kubelet 状态

```
[k8s@node1 docker]$ systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: disabled)
  Drop-In: /usr/lib/systemd/system/kubelet.service.d
           └─10-kubeadm.conf
   Active: activating (auto-restart) (Result: exit-code) since Tue 2022-03-22 09:33:52 CST; 3s ag

     Docs: https://kubernetes.io/docs/
  Process: 12721 ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFI
_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS (code=exited, status=1/FAILURE)
 Main PID: 12721 (code=exited, status=1/FAILURE)

Mar 22 09:33:52 node1.k8s.com systemd[1]: Unit kubelet.service entered failed state.
Mar 22 09:33:52 node1.k8s.com systemd[1]: kubelet.service failed.
[k8s@node1 docker]$
[k8s@node1 docker]$ journalctl -f -u kubelet
-- Logs begin at Tue 2022-03-22 01:45:18 CST. --
Mar 22 09:34:02 node1.k8s.com systemd[1]: kubelet.service: main process exited, code=exited
 status=1/FAILURE
Mar 22 09:34:02 node1.k8s.com systemd[1]: Unit kubelet.service entered failed state.
Mar 22 09:34:02 node1.k8s.com systemd[1]: kubelet.service failed.
Mar 22 09:34:13 node1.k8s.com systemd[1]: kubelet.service holdoff time over, scheduling rest
rt.
Mar 22 09:34:13 node1.k8s.com systemd[1]: Stopped kubelet: The Kubernetes Node Agent.
Mar 22 09:34:13 node1.k8s.com systemd[1]: Started kubelet: The Kubernetes Node Agent.
Mar 22 09:34:13 node1.k8s.com kubelet[12740]: E0322 09:34:13.108138   12740 server.go:205]
"Failed to load kubelet config file" err="failed to load Kubelet config file /var/lib/kubelet/conf
g.yaml, error failed to read kubelet config file \"/var/lib/kubelet/config.yaml\", error: open /va
/lib/kubelet/config.yaml: no such file or directory" path="/var/lib/kubelet/config.yaml"
Mar 22 09:34:13 node1.k8s.com systemd[1]: kubelet.service: main process exited, code=exited
```

```
 status=1/FAILURE
Mar 22 09:34:13 node1.k8s.com systemd[1]: Unit kubelet.service entered failed state.
Mar 22 09:34:13 node1.k8s.com systemd[1]: kubelet.service failed.
.....
```

此时，kubelet 每隔几秒就会重启，因为它陷入了一个等待 kubeadm 指令的死循环。

部署集群

通过下面这两条命令就能完成一个 Kubernetes 集群的部署：

```
# 创建一个 Master 节点
kubeadm init

# 将一个 Node 节点加入到当前集群中
kubeadm join <Master节点的IP和端口> ...
```

部署 Master 节点

在 node1 节点执行下面的命令：

```
sudo kubeadm init \
--image-repository registry.aliyuncs.com/google_containers \
--pod-network-cidr 10.244.0.0/16 \
--kubernetes-version stable-1.23
```

执行效果：

```
[k8s@node1 docker]$ sudo kubeadm init \
> --image-repository registry.aliyuncs.com/google_containers \
> --pod-network-cidr 10.244.0.0/16 \
> --kubernetes-version stable-1.23
[sudo] password for k8s:
[init] Using Kubernetes version: v1.23.5
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connect
on
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kuberne
es.default.svc kubernetes.default.svc.cluster.local node1.k8s.com] and IPs [10.96.0.1 192.168.1
3.21]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost node1.k8s.com] and IPs [1
2.168.153.21 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
```

```
[certs] etcd/peer serving cert is signed for DNS names [localhost node1.k8s.com] and IPs [192
168.153.21 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-fl
gs.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from
irectory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 6.503693 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-
ystem" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.23" in namespace kube-system with the co
figuration for the kubelets in the cluster
NOTE: The "kubelet-config-1.23" naming of the kubelet ConfigMap is deprecated. Once the
nversionedKubeletConfigMap feature gate graduates to Beta the default name will become ju
t "kubelet-config". Kubeadm upgrade will handle this transition transparently.
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node node1.k8s.com as control-plane by adding the labels:
node-role.kubernetes.io/master(deprecated) node-role.kubernetes.io/control-plane node.kub
rnetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node node1.k8s.com as control-plane by adding the taints
node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: c0wcm5.0yu9szfktsxvurza
[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 ord
r for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically ap
rove CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certifi
ates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet clie
t certificate and key
[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 192.168.153.21:6443 --token c0wcm5.0yu9szfktsxvurza \
        --discovery-token-ca-cert-hash sha256:f3cf8d3ccd735c39b8e941343f113cd4eb045a702
8db818a3fe918dd164e379
[k8s@node1 docker]$
```

按照上面的提示，执行下面的命令：

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

查看当前 Master 节点的状态：

```
[k8s@node1 docker]$ kubectl get nodes
NAME           STATUS     ROLES                  AGE     VERSION
node1.k8s.com   NotReady   control-plane,master   5m23s   v1.23.5
```

检查当前节点各个系统 Pod 的状态：

```
[k8s@node1 docker]$ kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
coredns-6d8c4cb4d-6t47l               0/1     Pending   0          5m28s
coredns-6d8c4cb4d-969m7               0/1     Pending   0          5m28s
etcd-node1.k8s.com                    1/1     Running   0          5m44s
kube-apiserver-node1.k8s.com          1/1     Running   0          5m42s
kube-controller-manager-node1.k8s.com   1/1     Running   0          5m42s
kube-proxy-x6x2g                      1/1     Running   0          5m29s
kube-scheduler-node1.k8s.com          1/1     Running   0          5m44s
```

Join 其他节点

复制刚才初始化后生成的 join 命令，用 root 用户在另外两个节点执行：

```
kubeadm join 192.168.153.21:6443 --token c0wcm5.0yu9szfktsxvurza \
        --discovery-token-ca-cert-hash sha256:f3cf8d3ccd735c39b8e941343f113cd4eb045a702
8db818a3fe918dd164e379
```

执行效果如下：

```
[root@node3 ~]# kubeadm join 192.168.153.21:6443 --token c0wcm5.0yu9szfktsxvurza \
```

```
>        --discovery-token-ca-cert-hash sha256:f3cf8d3ccd735c39b8e941343f113cd4eb045a70
98db818a3fe918dd164e379
[preflight] Running pre-flight checks
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable
kubelet.service'
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-
onfig -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-fl
gs.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.

[root@node3 ~]#
```

现在再查看当前各个节点的状态：

```
[k8s@node1 docker]$ kubectl get nodes
NAME           STATUS     ROLES                  AGE     VERSION
node1.k8s.com   NotReady   control-plane,master   8m23s   v1.23.5
node2.k8s.com   NotReady   <none>                 100s    v1.23.5
node3.k8s.com   NotReady   <none>                 90s     v1.23.5
```

新增了两个节点，但各个节点的状态还是 NotReady，因为现在还没有部署网络插件。

上面的 join 命令最好保存下来，方便下次使用。如果忘记，可以用下面的命令获取：

```
kubeadm token create --print-join-command
```

部署容器网络插件

Kubernetes 支持容器网络插件，使用的是一个名叫 CNI 的通用接口，它也是当前容器网络的事实标，市面上的所有容器网络开源项目都可以通过 CNI 接入 Kubernetes，比如 Flannel、Calico、Canal Romana 等等。

在 Master 节点执行下面的命令：

```
kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation
kube-flannel.yml
```

执行效果：

```
[k8s@node1 docker]$ kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/
aster/Documentation/kube-flannel.yml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
```

```
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[k8s@node1 docker]$
```

稍等会儿再查看，三个节点的状态已经是 Ready 了：

```
[k8s@node1 docker]$ kubectl get nodes
NAME           STATUS   ROLES                  AGE     VERSION
node1.k8s.com   Ready    control-plane,master   10m     v1.23.5
node2.k8s.com   Ready    <none>                 3m48s   v1.23.5
node3.k8s.com   Ready    <none>                 3m38s   v1.23.5
```

部署 Dashboard 可视化插件

Dashboard 是基于 web 的 Kubernetes 用户界面。 可以通过 Dashboard 将容器应用部署到 Kubernetes 集群中，也可以对容器应用排错，还能管理集群资源。

在 Master 节执行下面的命令：

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.1/aio/deploy/recommended.yaml
```

执行效果：

```
[k8s@node1 docker]$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.1/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
[k8s@node1 docker]$
```

部署完之后，可以用下面的命令查看 Dashboard 对应的 Pod 的状态：

```
[k8s@node1 docker]$ kubectl get pods -n kubernetes-dashboard
NAME                                         READY   STATUS    RESTARTS   AGE
dashboard-metrics-scraper-799d786dbf-zz4pj   1/1     Running   0          42s
kubernetes-dashboard-fb8648fd9-zdxnt         1/1     Running   0          42s
```

本地访问 Dashboard

为了避免造成安全隐患，1.7 版本之后的 Dashboard 项目部署完成后，默认只能通过 Proxy 的方式本地访问。用下面的命令启动代理：

```
[k8s@node1 docker]$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

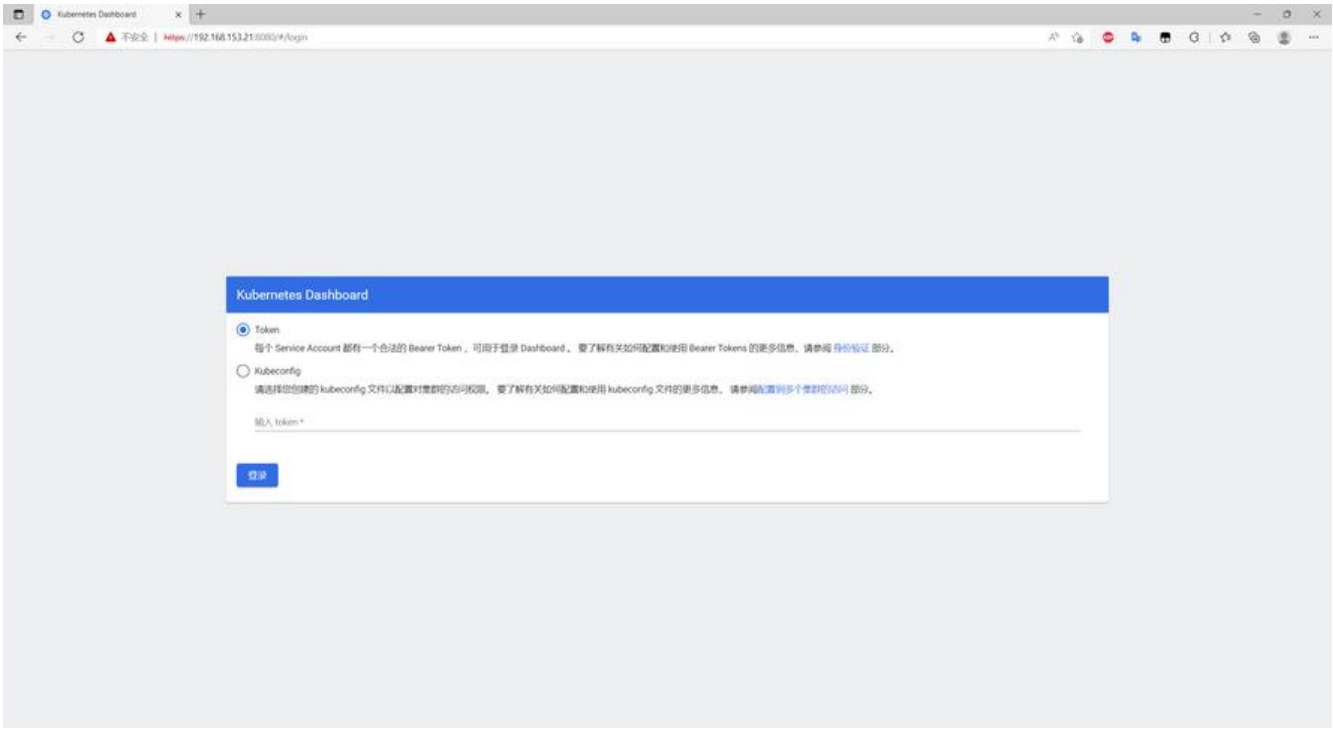
然后就可以在本地的浏览器输入 localhost:8001 地址，访问 Dashboard 了。

异地访问 Dashboard

如果想要在集群之外的机器上访问 Dashboard，可以采用端口转发命令：

```
kubectl port-forward --address 0.0.0.0 -n kubernetes-dashboard service/kubernetes-dashboad 8080:443
```

然后在浏览器中输入 https://192.168.153.21:8080/ ：



在登录界面可以看到 Kubernetes Dashborad 支持两种认证方式：

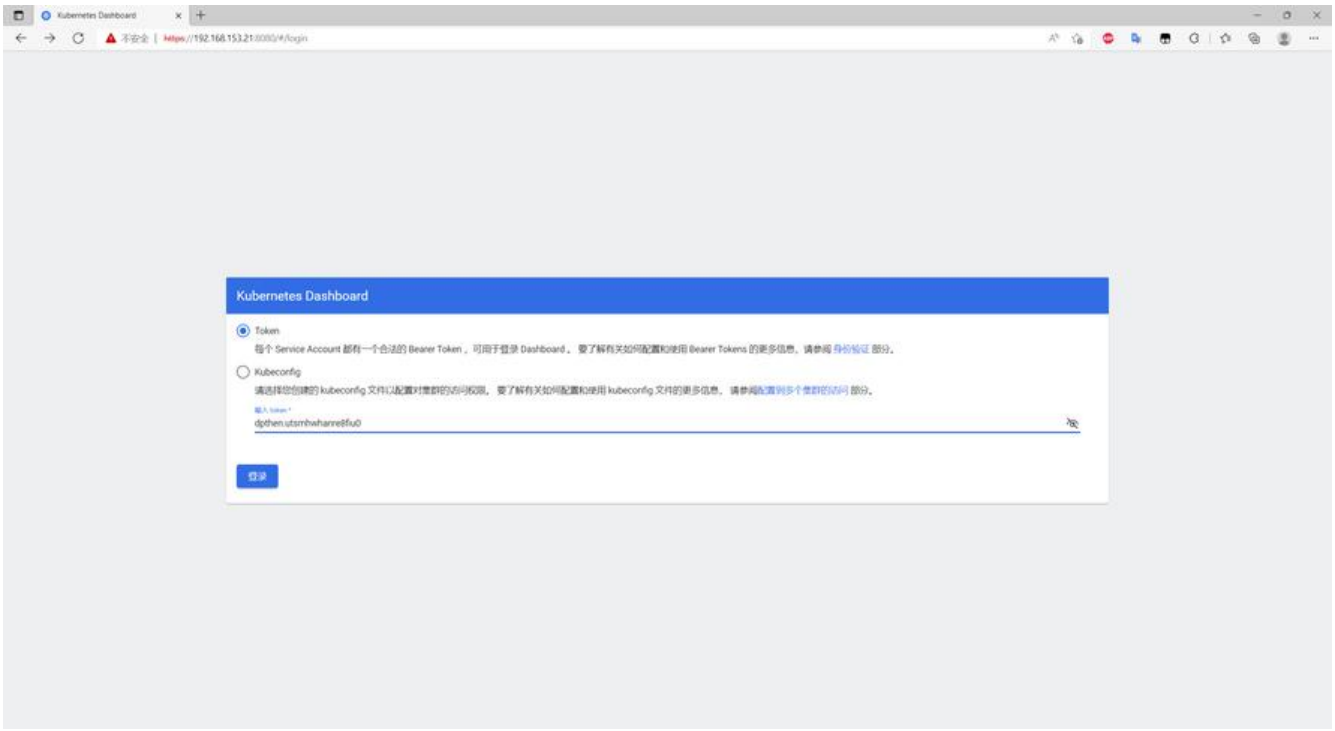
- **TOKEN**： 每个 Service Account 都有一个合法的 Bearer Token ，可用于登录 Dashboard 。 要解有关如何配置和使用 Bearer Tokens 的更多信息，请参阅 身份验证 部分。
- **Kubeconfig**：请选择您创建的 kubeconfig 文件以配置对集群的访问权限。 要了解有关如何配置使用 kubeconfig 文件的更多信息， 请参阅 配置到多个集群的访问 部分。

刚才在初始化的时候， kubeadm init 创建了一个有效期为 24 小时的 TOKEN，使用下面的命令查看 TOKEN：

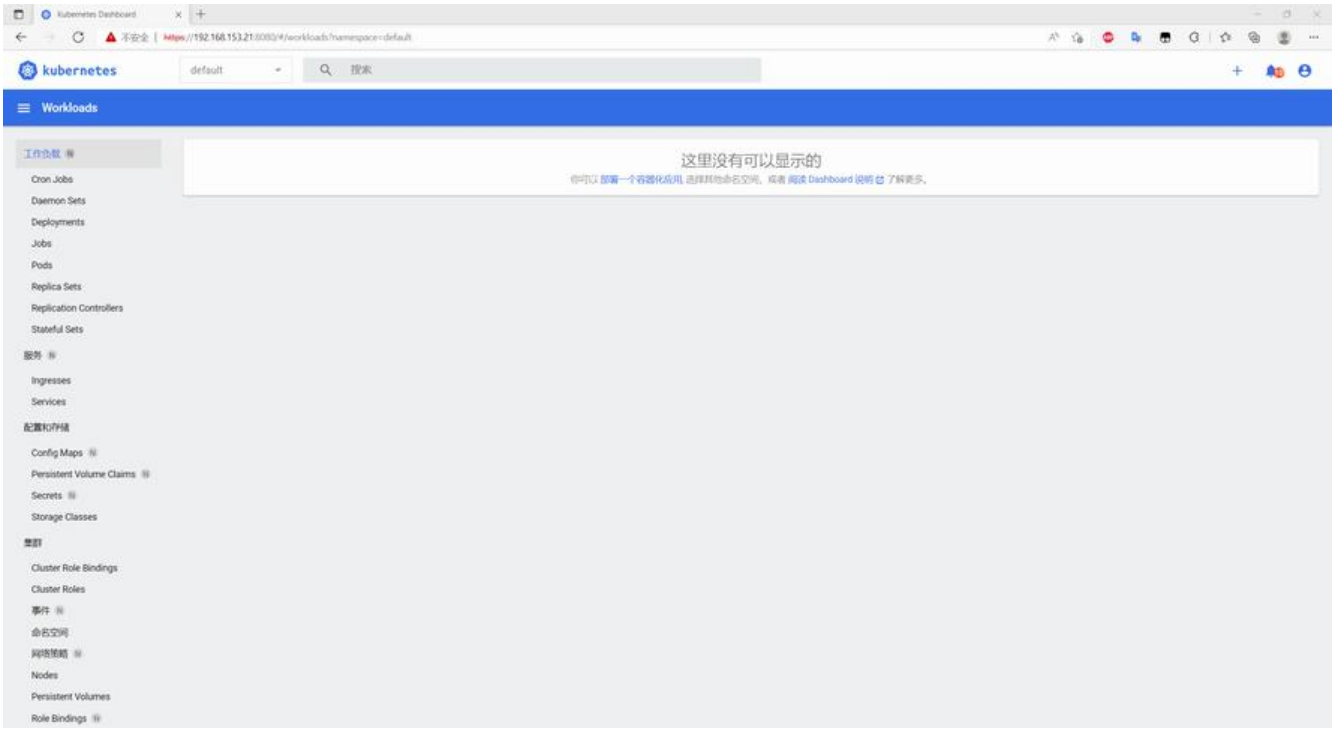
```
[k8s@node1 docker]$ kubeadm token list
TOKEN                     TTL         EXPIRES                     USAGES                   DESCRIPTION                                                EXTRA GROUPS
dpthen.utsmhwhanre8fiu0   23h         2022-03-23T01:44:17Z        authentication,signing   The default bootstrap token generated by 'kubeadm init'.   system:bootstrappers:kubeadm:default-no
```

e-token
[k8s@K8s01 ~]$

使用这个 TOKEN 可以登录 Dashboard：



登录后的效果：



但是，这个 TOKEN 的权限十分有限。

相关资料

使用 kubeadm 引导集群

Kubernetes部署 — Cloud Atlas 0.1 文档

Kubernetes 镜像

kubernetes/dashboard: General-purpose web UI for Kubernetes clusters

11 | 从0到1：搭建一个完整的Kubernetes集群