安装及使用

containerd 版本必须为：1.6.4

Kubernetes 版本必须为：v1.24.2。

一、安装 Kubernetes

1.升级内核

命令方式

```
#1.下载内核版本
https://repo.openeuler.org/openEuler-22.03-LTS/everything/aarch64/Packages/kernel-rt-5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64.rpm

#2.下载后，安装即可
rpm -ivh kernel-rt-5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64.rpm --nodeps --force

#3.设置默认加载内核
grub2-set-default  "OpenEuler (5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64)"

#4.重启生效
reboot
```

脚本方式

```
#!/bin/bash

COLOR='echo -e \E[01;31m'
END='\E[0m'

#升级内核
up_kernel(){
PACK='kernel-rt-5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64.rpm'
DIR='/home/soft/kernel/'
COLOR='echo -e \E[01;31m'
END='\E[0m'
```

```
UNAME=`uname -r`

if [ $UNAME = '5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64' ];then
   $COLOR"已安装"$END
 fi
if [ $UNAME != '5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64' ];then

if [ -f $DIR$PACK  ];then
    $COLOR"检测到源码包存在！开始安装"$END
    cd $DIR &&  rpm -ivh  ./$PACK   --nodeps --force  &&  grub2-set-defau
lt 'OpenEuler (5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64)'
    $COLOR"安装完成,重启生效"$END
else
    $COLOR"源码包不存在！开始下载"$END
    wget -P $DIR  https://repo.openeuler.org/openEuler-22.03-LTS/everyth
ing/aarch64/Packages/$PACK  --no-check-certificate


if [ $? -eq 0 ];then
    $COLOR"下载成功！！"$END
    cd $DIR && yum -y install ./$PACK  && grub2-set-default 'OpenEuler
(5.10.0-136.12.0.rt62.59.oe2203sp1.aarch64)'
    $COLOR"安装完成,10 秒后重启"$END
else
    $COLOR"下载失败，请检查网络配置！！"$END
   exit

fi
fi
fi

}

up_kernel
#重启生效
```

2.配置基础环境

命令方式

```
#1.配置 hosts 文件
vim /etc/hosts
172.30.201.209 master209
```

```
#2.修改主机名
hostnamectl set-hostname master209

#3.关闭防火墙
systemctl stop firewalld && systemctl disable firewalld.service

#4.关闭 swap
swapoff -a && cp /etc/fstab /etc/fstab.bak && cat /etc/fstab.bak | grep
-v swap > /etc/fstab

#5.关闭 selinux
setenforce 0 && sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/c
onfig  &> /dev/null

#6.调整系统时区
timedatectl  set-timezone Asia/Shanghai
ln -sf /usr/share/zoneinfo/Asia/Shanghai  /etc/localtime
systemctl  restart rsyslog && systemctl  restart crond

#7.网络转发
cat > /etc/sysctl.d/k8s.conf << EOF
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
vm.swappiness = 0
EOF
modprobe br_netfilter
sysctl -p  /etc/sysctl.d/k8s.conf
```

脚本方式

```
#!/bin/bash

COLOR='echo -e \E[01;31m'
END='\E[0m'
HostIp=`hostname -i`
HostName='master'
#安装基础软件包

#基础环境
Basic_environment(){
#hosts 文件
echo  $HostIp $HostName >> /etc/hosts


#设置主机名
hostnamectl set-hostname $HostName
```

```
$COLOR "主机名: $(hostname) " $END


#关闭 cselinux
setenforce 0 && sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/c
onfig  &> /dev/null
$COLOR "关闭 cselinux" $END

# 关闭防火墙
systemctl stop firewalld && systemctl disable firewalld.service
$COLOR "关闭防火墙" $END

# 关闭 swap 分区
swapoff -a && cp /etc/fstab /etc/fstab.bak && cat /etc/fstab.bak | grep
-v swap > /etc/fstab
$COLOR "关闭 swap 分区" $END

#调整系统时区
timedatectl  set-timezone Asia/Shanghai
ln -sf /usr/share/zoneinfo/Asia/Shanghai  /etc/localtime
systemctl  restart rsyslog && systemctl  restart crond

# 修改软硬连接数
cp /etc/security/limits.conf  /etc/security/limits.conf.backup
cat > /etc/security/limits.conf <<EOF
* soft core 0
* hard core 0
* soft core 0
* hard core 0
* soft nofile 65535
* hard nofile 65535
* soft nproc 65535
* hard nproc 65535
EOF
$COLOR "修改软硬连接数" $END


#网络转发
cat > /etc/sysctl.d/k8s.conf << EOF
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
vm.swappiness = 0
EOF


modprobe br_netfilter
sysctl -p  /etc/sysctl.d/k8s.conf
```

```
$COLOR "网络转发" $END

#kubernetes 源
cat > /etc/yum.repos.d/kubernetes.repo << EOF
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-
x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https:
//mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
$COLOR "配置 kubernetes 源" $END
}
Basic_environment
```

3.部署 containerd

```
#1.下载 containerd 和工具
      #地址：https://github.com/


runc     crictl   nerdctl  containerd

#2.解压所有并安装
tar -xvf  *.tar.gz -C /
cp -rf ./bin/*   /usr/bin/
mkdir /etc/containerd
#验证
containerd --version

#配置文件
cat > /etc/crictl.yaml <<EOF
runtime-endpoint: unix:///var/run/containerd/containerd.sock
image-endpoint: unix:///var/run/containerd/containerd.sock
timeout: 0
debug: false
pull-image-on-create: false
EOF
```

```
containerd config default > /etc/containerd/config.toml
systemctl start containerd  && systemctl enable containerd


#3.关联 harbor，修改/etc/containerd/config.toml(不需要修改)

[plugins."io.containerd.grpc.v1.cri".registry]
      config_path = "/etc/containerd/certs.d"

    [plugins."io.containerd.grpc.v1.cri".registry.auths]

    [plugins."io.containerd.grpc.v1.cri".registry.headers]

    [plugins."io.containerd.grpc.v1.cri".registry.mirrors]
       [plugins."io.containerd.grpc.v1.cri".registry.mirrors."server.har
bor.com:443"]
         endpoint = ["https://server.harbor.com:443"]

    [plugins."io.containerd.grpc.v1.cri".registry.configs]
       [plugins."io.containerd.grpc.v1.cri".registry.configs."server.har
bor.com:443".tls]
          insecure_skip_verify = true
       [plugins."io.containerd.grpc.v1.cri".registry.configs."server.har
bor.com:443".auth]
          username = "admin"
          password = "123456"


#4.更改/etc/containerd/config.toml 配置文件

vim /etc/containerd/config.toml


 更改以下：
...
         SystemdCgroup = true
...
sandbox_image = "registry.cn-hangzhou.aliyuncs.com/google_containers/pa
use:3.7
```

常用命令

```
#1.登录镜像仓库
nerdctl  login  -u admin  -p 123456 172.30.201.224:80  --insecure-regist
ry

#2.tag 镜像
ctr -n k8s.io i tag docker.io/library/mysql:5.7 172.30.201.224:80/test/
```

```
mysql:5.7

#3.推送镜像
ctr -n k8s.io image push --plain-http=true -k --user admin:123456 172.30.
201.224:80/test/mysql:5.7

#4.打包镜像
ctr -n k8s.io i export  mysql.tar docker.io/library/mysql:5.7
```

4.安装 Kubernetes 组件

下载地址：

http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-aarch64/?spm=a2c6h.25603864.0.0.1b0435dcGbrybZ

cri-tools-1.24.0-0.aarch64.rpm
kubectl-1.24.2-0.aarch64.rpm
kubernetes-cni-1.2.0-0.aarch64.rpm kubeadm-1.24.2-0.aarch64.rpm
kubelet-1.24.2-0.aarch64.rpm

```
#1.rpm --force 安装
rpm -ivh cri-tools-1.24.0-0.aarch64.rpm --force --nodeps
rpm -ivh kubectl-1.24.2-0.aarch64.rpm --force --nodeps
rpm -ivh kubernetes-cni-1.2.0-0.aarch64.rpm --force --nodeps
rpm -ivh kubeadm-1.24.2-0.aarch64.rpm --force --nodeps
rpm -ivh kubelet-1.24.2-0.aarch64.rpm --force --nodeps

#2.设置 kubelet 开机自启
systemctl enable kubelet
```

5.Kubernetes 镜像拉取

镜像源配置：arm64

cat < /etc/yum.repos.d/kubernetes.repo

[kubernetes] name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-aarch64 enable=1 gpgcheck=1 repo_gpgcheck=1
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF

```
#1.镜像查询命令
kubeadm config images list

#2.镜像拉取脚本
sudo tee ./images.sh <<-'EOF'
#!/bin/bash
```

```
url='registry.cn-hangzhou.aliyuncs.com/google_containers'
images=(
kube-apiserver:v1.24.2
kube-controller-manager:v1.24.2
kube-scheduler:v1.24.2
kube-proxy:v1.24.2
pause:3.7
etcd:3.5.3-0
/coredns/coredns:v1.8.6

)
for imageName in ${images[@]} ; do
docker pull $url/$imageName
done
EOF


#3.导入镜像包至 arm 服务器(scp)
#!/bin/bash
pwd=$(ls k8s-images |xargs echo)

for  i in $pwd; do
      ctr -n k8s.io image import $i  --all-platforms
done

chmod  +x ./images.sh && ./images.sh
```

6.初始化 Kubernetes

方式 1、下载 rpm 包

yum -y install bash-completion --downloadonly --downloaddir /opt/rp

yum -y install wget lvm2 bash-completion ntpdate ntp ipset ipvsadm iptables curl vim nrt-tools git sysstat conntrack --downloadonly --downloaddir /opt/rpm

方式 2、下载 rpm 包

vim /etc/yum.conf

keepcache=0

yum install bash-completion -y find /var/cache/ -name "*.rpm" |xargs mv -t /opt/rpm

```
#1.初始化命令
kubeadm init --apiserver-advertise-address=172.30.201.220  --apiserver-
bind-port=6443 --kubernetes-version=v1.24.2 --pod-network-cidr=10.233.0.
```

```
0/16 --service-cidr=172.30.0.0/16 --image-repository=registry.cn-hangzh
ou.aliyuncs.com/google_containers  --ignore-preflight-errors=swap


#2.创建工作目录
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

#3.加入集群使用：
kubeadm join 172.30.201.209:6443 --token wdf28x.2okoq90gzwg72325 \
      --discovery-token-ca-cert-hash sha256:7efe758ad39bb0d746f9986b0bc
b508f17db12c9fbedbb3b525e9e0770966865

#4.kubectl 命令加入 tab 自动补全
echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> /etc/profile
source /etc/profile
source <(kubectl completion bash)
echo 'source <(kubectl completion bash)' >> /root/.bashrc
source  /root/.bashrc


#5.清除 master 污点
kubectl taint node master209 node-role.kubernetes.io/control-plane:NoSc
hedule-
kubectl describe nodes master209 | grep Taint
```

7.calico 部署

cni 插件： 镜像需要拉取

ghcr.io/k8snetworkplumbingwg/multus-cni:v3.8

quay.io/tigera/operator:v1.29.0

```
wget https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/man
ifests/tigera-operator.yaml
wget https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/man
ifests/custom-resources.yaml
```

执行顺序

```
kubectl  create -f tigera-operator.yaml
kubectl  create -f custom-resources.yaml
kubectl  create -f multus-daemonset.yml
```

下载地址

https://github.com/k8snetworkplumbingwg/multus-cni

```
#vim multus-daemonset.yml

---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: network-attachment-definitions.k8s.cni.cncf.io
spec:
  group: k8s.cni.cncf.io
  scope: Namespaced
  names:
    plural: network-attachment-definitions
    singular: network-attachment-definition
    kind: NetworkAttachmentDefinition
    shortNames:
    - net-attach-def
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          description: 'NetworkAttachmentDefinition is a CRD schema spec
ified by the Network Plumbing
            Working Group to express the intent for attaching pods to one
 or more logical or physical
            networks. More information available at: https://github.com/
k8snetworkplumbingwg/multi-net-spec'
          type: object
          properties:
            apiVersion:
              description: 'APIVersion defines the versioned schema of th
is represen
                tation of an object. Servers should convert recognized sc
hemas to the
                latest internal value, and may reject unrecognized values.
 More info:
                https://git.k8s.io/community/contributors/devel/sig-arch
itecture/api-conventions.md#resources'
              type: string
            kind:
              description: 'Kind is a string value representing the REST
resource this
                object represents. Servers may infer this from the endpoi
nt the client
                submits requests to. Cannot be updated. In CamelCase. Mor
e info: https://git.k8s.io/community/contributors/devel/sig-architectur
e/api-conventions.md#types-kinds'
              type: string
            metadata:
```

```
            type: object
          spec:
            description: 'NetworkAttachmentDefinition spec defines the
desired state of a network attachment'
            type: object
            properties:
              config:
                description: 'NetworkAttachmentDefinition config is a J
SON-formatted CNI configuration'
                type: string
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: multus
rules:
  - apiGroups: ["k8s.cni.cncf.io"]
    resources:
      - '*'
    verbs:
      - '*'
  - apiGroups:
      - ""
    resources:
      - pods
      - pods/status
    verbs:
      - get
      - update
  - apiGroups:
      - ""
      - events.k8s.io
    resources:
      - events
    verbs:
      - create
      - patch
      - update
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: multus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: multus
subjects:
- kind: ServiceAccount
  name: multus
```

```
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: multus
  namespace: kube-system
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: multus-cni-config
  namespace: kube-system
  labels:
    tier: node
    app: multus
data:
  # NOTE: If you'd prefer to manually apply a configuration file, you ma
y create one here.
  # In the case you'd like to customize the Multus installation, you sho
uld change the arguments to the Multus pod
  # change the "args" line below from
  # - "--multus-conf-file=auto"
  # to:
  # "--multus-conf-file=/tmp/multus-conf/70-multus.conf"
  # Additionally -- you should ensure that the name "70-multus.conf" is
the alphabetically first name in the
  # /etc/cni/net.d/ directory on each node, otherwise, it will not be us
ed by the Kubelet.
  cni-conf.json: |
    {
      "name": "multus-cni-network",
      "type": "multus",
      "capabilities": {
        "portMappings": true
      },
      "delegates": [
        {
          "cniVersion": "0.3.1",
          "name": "default-cni-network",
          "plugins": [
            {
              "type": "flannel",
              "name": "flannel.1",
                "delegate": {
                  "isDefaultGateway": true,
                  "hairpinMode": true
                }
              },
              {
```

```
              "type": "portmap",
              "capabilities": {
                "portMappings": true
              }
            }
          ]
        }
      ],
      "kubeconfig": "/etc/cni/net.d/multus.d/multus.kubeconfig"
    }
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-multus-ds
  namespace: kube-system
spec:
  selector:
    matchLabels:
      name: multus
  updateStrategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        tier: node
        app: multus
        name: multus
    spec:
      hostNetwork: true
      tolerations:
      - operator: Exists
        effect: NoSchedule
      - operator: Exists
        effect: NoExecute
      serviceAccountName: multus
      containers:
      - name: kube-multus
        image: ghcr.io/k8snetworkplumbingwg/multus-cni:v3.8
        command: ["/entrypoint.sh"]
        args:
        - "--multus-conf-file=auto"
        - "--cni-version=0.3.1"
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
            cpu: "100m"
            memory: "50Mi"
```

```
      securityContext:
        privileged: true
      volumeMounts:
      - name: cni
        mountPath: /host/etc/cni/net.d
      - name: cnibin
        mountPath: /host/opt/cni/bin
      - name: multus-cfg
        mountPath: /tmp/multus-conf
    initContainers:
      - name: install-multus-binary
        image: ghcr.io/k8snetworkplumbingwg/multus-cni:v3.8
        command:
          - "cp"
          - "/usr/src/multus-cni/bin/multus"
          - "/host/opt/cni/bin/multus"
        resources:
          requests:
            cpu: "10m"
            memory: "15Mi"
        securityContext:
          privileged: true
        volumeMounts:
          - name: cnibin
            mountPath: /host/opt/cni/bin
            mountPropagation: Bidirectional
    terminationGracePeriodSeconds: 10
    volumes:
      - name: cni
        hostPath:
          path: /etc/cni/net.d
      - name: cnibin
        hostPath:
          path: /opt/cni/bin
      - name: multus-cfg
        configMap:
          name: multus-cni-config
          items:
          - key: cni-conf.json
            path: 70-multus.conf
```

8.work 节点配置

#1.执行 1 升级内核
#2.执行配置基础环境
#3.执行部署 containerd
#4.执行安装 Kubernetes 组件

```
#5.导入 Kubernetes 镜像和 calico 镜像


#6.加入集群使用：
kubeadm join 172.30.201.209:6443 --token wdf28x.2okoq90gzwg72325 \
      --discovery-token-ca-cert-hash sha256:7efe758ad39bb0d746f9986b0bc
b508f17db12c9fbedbb3b525e9e0770966865


#7.kubectl 命令加入 tab 自动补全
echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> /etc/profile
source /etc/profile
source <(kubectl completion bash)
echo 'source <(kubectl completion bash)' >> /root/.bashrc
source  /root/.bashrc

#8.拷贝主机/etc/kubernetes/admin.conf
scp -r root@172.30.201.209:/etc/kubernetes/admin.conf  /etc/kubernetes/
```

9.部署 dashboard

grep images kubernetes-dashboard.yaml

下载相关镜像

kubernetes-dashboard.yaml

```yaml
# Copyright 2017 The Kubernetes Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implie
d.
# See the License for the specific language governing permissions and
# limitations under the License.

apiVersion: v1
kind: Namespace
metadata:
  name: kubernetes-dashboard

---

apiVersion: v1
```

```yaml
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 30081
  type: NodePort
  selector:
    k8s-app: kubernetes-dashboard

---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-certs
  namespace: kubernetes-dashboard
type: Opaque

---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-csrf
  namespace: kubernetes-dashboard
type: Opaque
data:
  csrf: ""
```

```yaml
---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-key-holder
  namespace: kubernetes-dashboard
type: Opaque

---

kind: ConfigMap
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-settings
  namespace: kubernetes-dashboard

---

kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
rules:
  # Allow Dashboard to get, update and delete Dashboard exclusive secret
s.
  - apiGroups: [""]
    resources: ["secrets"]
    resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dash
board-certs", "kubernetes-dashboard-csrf"]
    verbs: ["get", "update", "delete"]
    # Allow Dashboard to get and update 'kubernetes-dashboard-settings'
config map.
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["kubernetes-dashboard-settings"]
    verbs: ["get", "update"]
    # Allow Dashboard to get metrics.
  - apiGroups: [""]
    resources: ["services"]
    resourceNames: ["heapster", "dashboard-metrics-scraper"]
    verbs: ["proxy"]
```

```
  - apiGroups: [""]
    resources: ["services/proxy"]
    resourceNames: ["heapster", "http:heapster:", "https:heapster:", "da
shboard-metrics-scraper", "http:dashboard-metrics-scraper"]
    verbs: ["get"]

---

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
rules:
  # Allow Metrics Scraper to get metrics from the Metrics server
  - apiGroups: ["metrics.k8s.io"]
    resources: ["pods", "nodes"]
    verbs: ["get", "list", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: kubernetes-dashboard
subjects:
  - kind: ServiceAccount
    name: kubernetes-dashboard
    namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kubernetes-dashboard
subjects:
```

```yaml
  - kind: ServiceAccount
    name: kubernetes-dashboard
    namespace: kubernetes-dashboard

---

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  template:
    metadata:
      labels:
        k8s-app: kubernetes-dashboard
    spec:
      securityContext:
        seccompProfile:
          type: RuntimeDefault
      containers:
        - name: kubernetes-dashboard
          image: registry.aliyuncs.com/google_containers/dashboard:v2.7.
0
          imagePullPolicy: Always
          ports:
            - containerPort: 8443
              protocol: TCP
          args:
            - --auto-generate-certificates
            - --namespace=kubernetes-dashboard
            # Uncomment the following line to manually specify Kubernetes
 API server Host
            # If not specified, Dashboard will attempt to auto discover t
he API server and connect
            # to it. Uncomment only if the default does not work.
            # - --apiserver-host=http://my-address:port
          volumeMounts:
            - name: kubernetes-dashboard-certs
              mountPath: /certs
              # Create on-disk volume to store exec logs
            - mountPath: /tmp
              name: tmp-volume
```

```
          livenessProbe:
            httpGet:
              scheme: HTTPS
              path: /
              port: 8443
            initialDelaySeconds: 30
            timeoutSeconds: 30
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            runAsUser: 1001
            runAsGroup: 2001
      volumes:
        - name: kubernetes-dashboard-certs
          secret:
            secretName: kubernetes-dashboard-certs
        - name: tmp-volume
          emptyDir: {}
      serviceAccountName: kubernetes-dashboard
      nodeSelector:
        "kubernetes.io/os": linux
      # Comment the following tolerations if Dashboard must not be deplo
yed on master
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: dashboard-metrics-scraper
  name: dashboard-metrics-scraper
  namespace: kubernetes-dashboard
spec:
  ports:
    - port: 8000
      targetPort: 8000
  selector:
    k8s-app: dashboard-metrics-scraper

---

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
```

```
    k8s-app: dashboard-metrics-scraper
  name: dashboard-metrics-scraper
  namespace: kubernetes-dashboard
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: dashboard-metrics-scraper
  template:
    metadata:
      labels:
        k8s-app: dashboard-metrics-scraper
    spec:
      securityContext:
        seccompProfile:
          type: RuntimeDefault
      containers:
        - name: dashboard-metrics-scraper
          image: registry.aliyuncs.com/google_containers/metrics-scraper:
v1.0.8
          ports:
            - containerPort: 8000
              protocol: TCP
          livenessProbe:
            httpGet:
              scheme: HTTP
              path: /
              port: 8000
            initialDelaySeconds: 30
            timeoutSeconds: 30
          volumeMounts:
          - mountPath: /tmp
            name: tmp-volume
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            runAsUser: 1001
            runAsGroup: 2001
      serviceAccountName: kubernetes-dashboard
      nodeSelector:
        "kubernetes.io/os": linux
      # Comment the following tolerations if Dashboard must not be deplo
yed on master
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      volumes:
        - name: tmp-volume
          emptyDir: {}
```

kubernetes-dashboard-user-admin.yaml

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

启动

```
kubectl create -f kubernetes-dashboard.yaml
kubectl create -f kubernetes-dashboard-user-admin.yaml
```

token 创建

```
kubectl -n kubernetes-dashboard create token admin-user
```

访问：https://ip:30081
