## kubernetes-1：使用kubeadm搭建K8S单master节点集群

**原创；微信公众号：千里行走；**

**受限图片大小限制，有些图片不是很清晰，可以到微信公众号查看；**

目录

(1).方案选择

(2).前期准备

(3).安装kubectl，kubeadm，kubelet

(4).配置kubelet

(5).安装前置依赖 crictl

(6).安装前置依赖 socat

(7).初始化集群

(8).安装kubernetes-dashboard

(9).配置https访问k8s-dashboard

(10).给集群增加node

(11).重置集群

(12).参考文章&感谢

正文

**(1).方案选择**

使用官方推荐的kubeadm搭建。

注意：

现在官方推荐的是kubespray，但也是基于kubeadm；除此之外，还有kind，minikube，但是并不试用于部署生产级别集群。

如果没有足够的K8S储备，生产环境还是使用阿里云的K8S集群更稳妥；阿里云的容器化服务支持多可用区的K8S集群，多可用区在物理上有隔离，可以在一定程度上避免阿里云自身的物理故障。

**(2).前期准备**

1.我们需要禁用 swap。

使用 sudo cat /proc/swaps 验证 swap 配置的设备和文件。

通过 swapoff -a 关闭 swap 。

执行完上述操作，swap 便会被禁用，当然你也可以再次通过上述命令，或者 free 命令来确认是否还有 swap 存在。

2.通过 sudo cat /sys/class/dmi/id/product\_uuid 可查看机器的 product\_uuid 请确保要搭建集群的所有节点的 product\_uuid 均不相同。同时所有节点的 Mac 地址也不能相同，通过 ip a 或者 ifconfig -a 可进行查看。

3.通过 sudo netstat -ntlp |grep -E '6443|23[79,80]|1025[0,1,2]' 查看这些端口是否被占用，如果被占用，请手动释放。

4.安装docker:

我使用：官网方法安装最新

stable-version:https://docs.docker.com/install/linux/docker-ce/centos/#install-docker-ce

**(3).安装kubectl，kubeadm，kubelet**

https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.13.md#downloads-for-v1133

选择v1.13.3的server:

wget https://dl.k8s.io/v1.13.3/kubernetes-server-linux-amd64.tar.gz

下载完成后，验证文件是否正确无误，验证通过后进行解压。

[root@master tmp]# echo '024847f8a370c4edb920a4621904540bf15c3afc0c688a134090ae8503a51296c83ebc402365f5a2be5aff01a8ca89910423874489dd2412608de0e38b455fb5 kubernetes-server-linux-amd64.tar.gz' | sha512sum -c

kubernetes-server-linux-amd64.tar.gz: 确定

[root@master tmp]# tar -zxf kubernetes-server-linux-amd64.tar.gz

[root@master tmp]# ls kubernetes

addons kubernetes-src.tar.gz LICENSES server

[root@master tmp]# ls kubernetes-server/server/bin/ | grep -E 'kubeadm|kubelet|kubectl'

kubeadm

kubectl

kubelet

ln -s /app/3rd/kubernetes-server/server/bin/kubeadm /usr/bin/kubeadm

ln -s /app/3rd/kubernetes-server/server/bin/kubectl /usr/bin/kubectl

ln -s /app/3rd/kubernetes-server/server/bin/kubelet /usr/bin/kubelet

kubeadm version

kubeadm version: &version.Info{Major:"1", Minor:"13", GitVersion:"v1.13.3", GitCommit:"721bfa751924da8d1680787490c54b9179b1fed0", GitTreeState:"clean", BuildDate:"2019-02-01T20:05:53Z", GoVersion:"go1.11.5", Compiler:"gc", Platform:"linux/amd64"}

kubectl version --client

Client Version: version.Info{Major:"1", Minor:"13", GitVersion:"v1.13.3", GitCommit:"721bfa751924da8d1680787490c54b9179b1fed0", GitTreeState:"clean", BuildDate:"2019-02-01T20:08:12Z", GoVersion:"go1.11.5", Compiler:"gc", Platform:"linux/amd64"}

kubelet --version

Kubernetes v1.13.3

**(4).配置kubelet**

新建文件： /etc/systemd/system/kubelet.service

文件内容：

[Unit]

Description=kubelet: The Kubernetes Agent

Documentation=http://kubernetes.io/docs/

[Service]

ExecStart=/usr/bin/kubelet

Restart=always

StartLimitInterval=0

RestartSec=10

[Install]

WantedBy=multi-user.target

新建文件：/etc/systemd/system/kubelet.service.d/kubeadm.conf

文件内容：

[Service]

Environment="KUBELET\_KUBECONFIG\_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"

Environment="KUBELET\_CONFIG\_ARGS=--config=/var/lib/kubelet/config.yaml"

EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env

EnvironmentFile=-/etc/default/kubelet

ExecStart=

ExecStart=/usr/bin/kubelet $KUBELET\_KUBECONFIG\_ARGS $KUBELET\_CONFIG\_ARGS $KUBELET\_KUBEADM\_ARGS $KUBELET\_EXTRA\_ARGS

systemctl enable kubelet

Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.

在这里我们添加了 kubelet 的 systemd 配置，然后添加了它的 Drop-in 文件，我们增加的这个 kubeadm.conf 文件，会被 systemd 自动解析，用于复写 kubelet 的基础 systemd 配置，可以看到我们增加了一系列的配置参数。

**(5).安装前置依赖 crictl**

crictl 包含在 cri-tools 项目中，这个项目中包含两个工具：

crictl 是 kubelet CRI (Container Runtime Interface) 的 CLI 。

critest 是 kubelet CRI 的测试工具集。

release page:

https://github.com/kubernetes-sigs/cri-tools/releases

版本兼容readme：

https://github.com/kubernetes-sigs/cri-tools#current-status

我使用的k8s是v.13.3，对应的crictl版本是：

Kubernetes Version cri-tools Version cri-tools branch

1.13.X v1.13.0 v1.13.0

Install critest

VERSION="v1.13.0"

wget https://github.com/kubernetes-sigs/cri-tools/releases/download/$VERSION/critest-$VERSION-linux-amd64.tar.gz

sudo tar zxvf critest-$VERSION-linux-amd64.tar.gz -C /usr/local/bin

rm -f critest-$VERSION-linux-amd64.tar.gz

crictl:

wget https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.13.0/crictl-v1.13.0-linux-amd64.tar.gz

crictltest:

wget https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.13.0/critest-v1.13.0-linux-amd64.tar.gz

./critest -version

critest version: v1.13.0

PASS

./crictl -version

crictl version v1.13.0

ln -s /app/3rd/critools/bin/crictl /bin/crictl

ln -s /app/3rd/critools/bin/critest /bin/critest

**(6).安装前置依赖 socat**

socat 是一款很强大的命令行工具，可以建立两个双向字节流并在其中传输数据。这么说你也许不太理解，简单点说，它其中的一个功能是可以实现端口转发。

无论在 K8S 中，还是在 Docker 中，如果我们需要在外部访问服务，端口转发是个必不可少的部分。当然，你可能会问基本上没有任何地方提到说 socat 是一个依赖项啊，别急，我们来看下 K8S 的源码便知道了。

func portForward(client libdocker.Interface, podSandboxID string, port int32, stream io.ReadWriteCloser) error {

// 省略了和 socat 无关的代码

socatPath, lookupErr := exec.LookPath("socat")

if lookupErr != nil {

return fmt.Errorf("unable to do port forwarding: socat not found.")

}

args := []string{"-t", fmt.Sprintf("%d", containerPid), "-n", socatPath, "-", fmt.Sprintf("TCP4:localhost:%d", port)}

// ...

}

socat 的安装很简单 CentOS 下执行 sudo yum install -y socat ，Debian/Ubuntu 下执行 sudo apt-get install -y socat 即可完成安装。

**(7).初始化集群**

使用网络方案：flannel。

7.1.下载K8S相关镜像，如果不能翻墙，先按照下列步骤下载镜像。

出现了墙的问题：

批量下载及转换标签

kubeadm config images list |sed -e 's/^/docker pull /g' -e 's#k8s.gcr.io#docker.io/mirrorgooglecontainers#g' |sh -x 下载需要的镜像

docker images |grep mirrorgooglecontainers |awk '{print "docker tag ",$1":"$2,$1":"$2}' |sed -e 's#docker.io/mirrorgooglecontainers#k8s.gcr.io#2' |sh -x 重命名镜像

docker images |grep mirrorgooglecontainers |awk '{print "docker tag ",$1":"$2,$1":"$2}' |sed -e 's#mirrorgooglecontainers#k8s.gcr.io#2' |sh -x 重命名镜像

docker images |grep mirrorgooglecontainers |awk '{print "docker rmi ", $1":"$2}' |sh -x 删除mirrorgooglecontainers镜像

coredns没包含在docker.io/mirrorgooglecontainers中，需要手工从coredns官方镜像转换下。

docker pull coredns/coredns:1.2.6

docker tag coredns/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6

docker rmi coredns/coredns:1.2.6

docker tag k8s.gcr.io/coredns:1.2.6 k8s.gcr.io/coredns:stable-1

docker tag k8s.gcr.io/kube-controller-manager:v1.13.3 k8s.gcr.io/kube-controller-manager:stable-1

docker tag k8s.gcr.io/kube-apiserver:v1.13.3 k8s.gcr.io/kube-apiserver:stable-1

docker tag k8s.gcr.io/kube-proxy:v1.13.3 k8s.gcr.io/kube-proxy:stable-1

docker tag k8s.gcr.io/kube-scheduler:v1.13.3 k8s.gcr.io/kube-scheduler:stable-1

docker tag k8s.gcr.io/etcd:3.2.24 k8s.gcr.io/etcd:stable-1

docker tag k8s.gcr.io/pause:3.1 k8s.gcr.io/pause:stable-1

docker images:

REPOSITORY TAG IMAGE ID CREATED SIZE

docker.io/k8s.gcr.io/kube-apiserver v1.13.3 fe242e556a99 9 days ago 181 MB

docker.io/k8s.gcr.io/kube-controller-manager v1.13.3 0482f6400933 9 days ago 146 MB

docker.io/k8s.gcr.io/kube-proxy v1.13.3 98db19758ad4 9 days ago 80.3 MB

docker.io/k8s.gcr.io/kube-scheduler v1.13.3 3a6f709e97a0 9 days ago 79.6 MB

k8s.gcr.io/coredns 1.2.6 f59dcacceff4 3 months ago 40 MB

docker.io/k8s.gcr.io/etcd 3.2.24 3cab8e1b9802 4 months ago 220 MB

docker.io/k8s.gcr.io/pause 3.1 da86e6ba6ca1 13 months ago 742 kB

7.2.下载cni-plugins相关组件：

这个是需要手动的部分：cni-plugins-amd64-v0.6.0.tgz。

下载地址：

https://github.com/hepyu/kubernetes-util/blob/master/cni-plugins-amd64-v0.6.0.tgz

下载后解压到目录/opt/cni/bin。

7.3.初始化集群

kubeadm init --ignore-preflight-errors='NumCPU' --kubernetes-version v1.13.3 --pod-network-cidr=10.244.0.0/16

[root@iZ253ayhxa9Z kubelet.service.d]# kubeadm init --ignore-preflight-errors='NumCPU' --kubernetes-version v1.13.3 --pod-network-cidr=10.244.0.0/16

关键日志：

Your Kubernetes master 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

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/

You can now join any number of machines by running the following on each node

as root:

kubeadm join 192.17.7.150:6443 --token 01qdlc.r6nt1ely2k7n1efb --discovery-token-ca-cert-hash sha256:bdd31cd9bbe138503ae43f14f03e9f91feab5d36524d95908e2d89dfede5cbc5

执行：

mkdir -p $HOME/.kube

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

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

[root@iZ253ayhxa9Z kubelet.service.d]# kubectl cluster-info

Kubernetes master is running at https://192.17.7.150:6443

KubeDNS is running at https://192.17.7.150:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

[root@iZ253ayhxa9Z kubelet.service.d]# kubectl get nodes  查看k8s集群的nodes 列表。

NAME STATUS ROLES AGE VERSION

iz253ayhxa9z NotReady master 112s v1.13.3

docker ps -a 看到，此时，CNI 也尚未初始化完成，我们还需完成以下的步骤：安装flannel到k8s。

[root@iZ253ayhxa9Z kubelet.service.d]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

clusterrole.rbac.authorization.k8s.io/flannel created

clusterrolebinding.rbac.authorization.k8s.io/flannel created

serviceaccount/flannel created

configmap/kube-flannel-cfg created

daemonset.extensions/kube-flannel-ds created

稍等片刻，再次查看 Node 状态：

[root@iZ253ayhxa9Z kubelet.service.d]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

iz253ayhxa9z Ready master 16m v1.13.3

完成。

**(8).安装kubernetes-dashboard**

kubernetes-dashboard先从国内镜像拉下来&重命名：

$ docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.1

$ docker tag registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.1 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1

$ docker rmi registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.1

打上kis.gcr.io的tag，这样执行Dashboard拉取的时候就直接本地拿pull下来的直接安装。

安装

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

安装后：查看状态：

kubectl describe pod kubernetes-dashboard -n kube-system

docker ps 中可以看到dashboard容器running。

当已经执行完以上步骤后，可检查下是否安装成功:

kubectl -n kube-system get all -l k8s-app=kubernetes-dashboard

kubectl -n kube-system port-forward pod/kubernetes-dashboard-57df4db6b-7cr67 8443

nohup kubectl -n kube-system port-forward pod/kubernetes-dashboard-57df4db6b-7cr67 8443 > /data/coohua/logs/k8s-dashboard/nohup.out &

**(9).配置https访问k8s-dashboard**

配置nginx+https访问：

location /dashboard/ {

                proxy\_pass https://127.0.0.1:8443/;

                #proxy\_pass http://localhost:9090/;

                proxy\_set\_header Host $host;

                proxy\_set\_header X-Real-IP $remote\_addr;

                proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

                proxy\_set\_header Connection "";

}

# HTTPS server

#

server {

        listen       443 ssl;

        server\_name  47.92.123.228;

        access\_log /data/logs/nginx/future.ssl.access.log  access;

        ssl\_certificate      /usr/local/openresty/nginx/conf/ssl/server.crt;

        ssl\_certificate\_key  /usr/local/openresty/nginx/conf/ssl/server.key;

        ssl\_session\_cache    shared:SSL:1m;

        ssl\_session\_timeout  5m;

        ssl\_ciphers  HIGH:!aNULL:!MD5;

        ssl\_prefer\_server\_ciphers  on;

        include conf.d/\*.conf;

}

对于我们的 新版本 而言，我们 使用令牌登录 的方式。

查找 Token

master $ kubectl -n kube-system get serviceaccount -l k8s-app=kubernetes-dashboard -o yaml

apiVersion: v1

items:

- apiVersion: v1

  kind: ServiceAccount

  metadata:

    annotations:

      kubectl.kubernetes.io/last-applied-configuration: |

        {"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},"labels":{"k8s-app":"kubernetes-dashboard"},"name":"kubernetes-dashboard","namespace":"kube-system"}}

    creationTimestamp: 2019-01-15T09:06:04Z

    labels:

      k8s-app: kubernetes-dashboard

    name: kubernetes-dashboard

    namespace: kube-system

    resourceVersion: "85278"

    selfLink: /api/v1/namespaces/kube-system/serviceaccounts/kubernetes-dashboard

    uid: c95ab4b8-18a4-11e9-8dd9-02420f98d972

  secrets:

  - name: kubernetes-dashboard-token-2blsj

kind: List

metadata:

  resourceVersion: ""

  selfLink: ""

首先，我们查看刚才创建出的 serviceaccount 可以看到其中有配置 secrets 。

查看该 secret 详情获得 Token

master $ kubectl -n kube-system describe secrets kubernetes-dashboard-token-2blsj

Name:         kubernetes-dashboard-token-2blsj

Namespace:    kube-system

Labels:       <none>

Annotations:  kubernetes.io/service-account.name: kubernetes-dashboard

              kubernetes.io/service-account.uid: c95ab4b8-18a4-11e9-8dd9-02420f98d972

Type:  kubernetes.io/service-account-token

Data

====

ca.crt:     1025 bytes

namespace:  11 bytes

token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9..eltaG9xfgT2YAGi9Crn-I-Jak8MJ\_-JqhxM4tJ4Mppadk1x4lzqRM8RbxCTlPXdEZKettKbJtokvtO6hGC302oszWOHTSS1ipIsM348bdh5-8YcVaa2nt0SDSbmdnXlC3bz4p2qaURaDz4k8IekiaxD3fg63TL0mDPxGPfoNLoyDGK-vh57vMl9CqWtgo4G2WV0oSb0lgm2P3K5nUvXTbTfnztC5Up\_bNE9Fzmt5BmDCSeV-1YMjakKWgRyFZoXThqval52Df1CgMNKbltNrGVlyaqRTZRG9p5JEoYwuguZvIjZ3FVWVvXNxB\_r6vvqDWNixCaNtc4sDVXcVq1ZYQA

使用上边的token登陆页面。

发现很多页面没有权限，需要我们创建有权限的serviceAccount账户。

建立文件：

> admin-user.service-account.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

  name: admin-user

  namespace: kube-system

执行：

kubectl create -f ./admin-user.service-account.yaml

serviceaccount/admin-user created

创建 RoleBinding: 这里为了方便直接绑定了 cluster-admin 的 ClusterRole ，但是生产环境下，请按照实际情况进行授权，参考前面第 8 节相关的内容：

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: kube-system

执行：kubectl create -f ./admin-user.role-binding.yaml

clusterrolebinding.rbac.authorization.k8s.io/admin-user created

执行命令找到secrets-name:

kubectl -n kube-system get serviceaccount -o yaml | grep -i -A30 -B30 admin-user

apiVersion: v1

items:

- apiVersion: v1

  kind: ServiceAccount

  metadata:

    creationTimestamp: 2019-01-18T06:42:05Z

    name: admin-user

    namespace: kube-system

    resourceVersion: "386003"

    selfLink: /api/v1/namespaces/kube-system/serviceaccounts/admin-user

    uid: 2b553857-1aec-11e9-8dd9-02420f98d972

  secrets:

  - name: admin-user-token-ctbwd

执行命令找到token:

kubectl -n kube-system describe secrets admin-user-token-ctbwd

Name:         admin-user-token-ctbwd

Namespace:    kube-system

Labels:       <none>

Annotations:  kubernetes.io/service-account.name: admin-user

              kubernetes.io/service-account.uid: 2b553857-1aec-11e9-8dd9-02420f98d972

Type:  kubernetes.io/service-account-token

Data

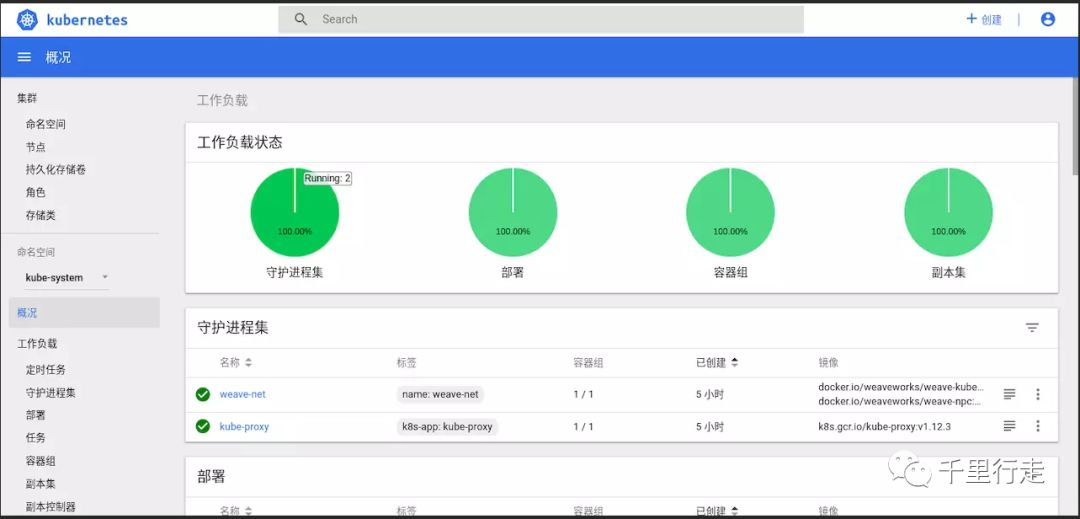
====

ca.crt:     1025 bytes

namespace:  11 bytes

token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLTduZDZqIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI2MjYyOGMwMS0yNTJhLTExZTktOTlhMS0wMjQyYWM2MzMzNzUiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.IXWpSjyt8QP\_PiVM6fO5pg9JeqrxtutTNPgWeo0GW497lBOGOtn0lKYO3jVDPMeU6VQouUzQcW12I6i-od3SFM0NlT7Syufpik\_9aR2UaldSXK43p-hpuAx7PldxUlJrZNiUUsmQVy-wtq183wxaXbscsZHlG3gnWtc-85Cf-3NSqARDY2Ot8V3SkSTisFckY5PfipQ6xo5HXFZBIG7Ankpqhe3IWnZvkGoUQoFR9LPTE8-yljAHwb0jnfIdGBiT5yP8jkHLNLdBPVGCilYOdDL8hldsA-EuphOS9-jQppeSFY\_c1vPdUtdIn-ps7WYMqBGUlXP28zo\_UWXW7YzSOA

使用上边的token令牌登录，就可以正常使用k8s-dashboard了。



**(10).给集群增加node**

前置需要把kubeadm, kubectl, kubelet都安装好，但是不要执行kubeadm init,直接执行kubeadm join。

kubeadm join 192.16.11.137:6443 --token 8mgmjx.jrp18nhypiggypst --discovery-token-ca-cert-hash sha256:9005c3e51189da23fdfa00a326f979fdb167c624579adeb6db5b6441339bf8ba

这个joiin命令需要在k8s-master节点执行如下命令或得：

kubeadm token create --print-join-command

**(11).重置集群**

如果想重新安装，执行如下命令恢复安装前状态：kubeadm reset。

再次执行如下命令即可快速重装k8s节点：

kubeadm init --ignore-preflight-errors='NumCPU' --kubernetes-version v1.13.3 --pod-network-cidr=10.244.0.0/16

**(12).参考文章&感谢：**

1.解决k8s镜像被墙的问题参考：

https://blog.csdn.net/bbwangj/article/details/85017765

2.k8s dashboard安装参考：

https://juejin.im/post/5c24d8b15188252a9412fb46

3.特别感谢：

@张晋淘(掘金小册--Kubernetes 从上手到实践作者)