## **1.什么是 Kubernetes**

* Kubernetes 看作是用来是一个部署镜像的平台
* 可以用来操作多台机器调度部署镜像
* 在 Kubernetes 中，可以使用集群来组织服务器的。集群中会存在一个 Master 节点，该节点是 Kubernetes 集群的控制节点，负责调度集群中其他服务器的资源。其他节点被称为 Node

## **2. 基础安装**

* Master & Node 节点都需要安装

### **2.1 安装些必备组件**

* vim 是 Linux 下的一个文件编辑器
* wget 可以用作文件下载使用
* ntpdate 则是可以用来同步时区

yum install vim wget ntpdate -y

### **2.2 关闭防火墙**

* kubernetes 会创建防火墙规则,先关闭firewalld

systemctl stop firewalld & systemctl disable firewalld

### **2.3 关闭 Swap 分区**

* Swap 是 Linux 的交换分区，在系统资源不足时，Swap 分区会启用,这个我们不需要
* 应该让新创建的服务自动调度到集群的其他 Node 节点中去，而不是使用 Swap 分区

#临时关闭

swapoff -a

### **2.4 关闭 Selinux**

* 关闭Selinux是为了支持容器可以访问宿主机文件系统

# 暂时关闭 selinux

setenforce 0

# 永久关闭

vi /etc/sysconfig/selinux

# 修改以下参数，设置为disable

SELINUX=disabled

### **2.5 统一我们的系统时间和时区**

* 使用 ntpdate 来统一我们的系统时间和时区,服务器时间与阿里云服务器对齐

# 统一时区，为上海时区

ln -snf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

bash -c "echo 'Asia/Shanghai' > /etc/timezone"

# 统一使用阿里服务器进行时间更新

ntpdate ntp1.aliyun.com

### **2.6 安装 Docker**

* 在 kubernetes 中的组件，服务都可以 Docker 镜像方式部署的,所以需要安装Docker
* device-mapper-persistent-data: 存储驱动，Linux上的许多高级卷管理技术
* lvm: 逻辑卷管理器，用于创建逻辑磁盘分区使用

yum install -y yum-utils device-mapper-persistent-data lvm2

sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

yum install docker-ce -y

systemctl start docker

systemctl enable docker

sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["https://fwvjnv59.mirror.aliyuncs.com"]

}

EOF

sudo systemctl daemon-reload

sudo systemctl restart docker.service

### **2.7 安装 Kubernetes 组件**

#### **2.7.1 切换阿里云源**

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

[kubernetes]

name=Kubernetes

baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=0

repo\_gpgcheck=0

gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg

http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

#### **2.7.2 安装 Kubernetes 组件**

* kubelet 是 Kubernetes 中的核心组件。它会运行在集群的所有节点上，并负责创建启动服务容器
* kubectl 则是Kubernetes的命令行工具。可以用来管理，删除，创建资源
* kubeadm 则是用来初始化集群，子节点加入的工具

yum install -y kubelet kubeadm kubectl

# 启动kubelet

systemctl enable kubelet && systemctl start kubelet

### **2.8 设置bridge-nf-call-iptables**

* 配置内核参数，将桥接的IPV4浏览传递到iptables链
* 开启了bridge-nf-call-iptables

echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables

## **3. Master**

* Master 节点是集群内的调度和主要节点

### **3.1 修改主机名称为 master**

hostnamectl **set**-hostname master

### **3.2 配置hosts**

ip addr

vim /etc/hosts

172.31.178.169 master master

### **3.3 配置 Kubernetes 初始化文件**

* init-defaults 输出一份默认初始化配置文件

kubeadm config print init-defaults > init-kubeadm.conf

vim init-kubeadm.conf

* 更换 Kubernetes 镜像仓库为阿里云镜像仓库，加速组件拉取
* 替换 ip 为自己主机 ip
* 配置 pod 网络为 flannel 网段
* 为了让集群之间可以互相通信，需要配置子网络,这些在后面的flannel网络中需要用到
  + 10.96.0.0/12 是Kubernetes内部的网络pods需要的网络
  + 10.244.0.0/16 是Kubernetes内部services需要的网络

- imageRepository: k8s.gcr.io 更换k8s镜像仓库+ imageRepository: registry.cn-hangzhou.aliyuncs.com/google\_containers- localAPIEndpointc，advertiseAddress为master ip ,port默认不修改

localAPIEndpoint:+ advertiseAddress: 172.31.178.169 # 此处为master的IP

bindPort: 6443

# 配置子网络

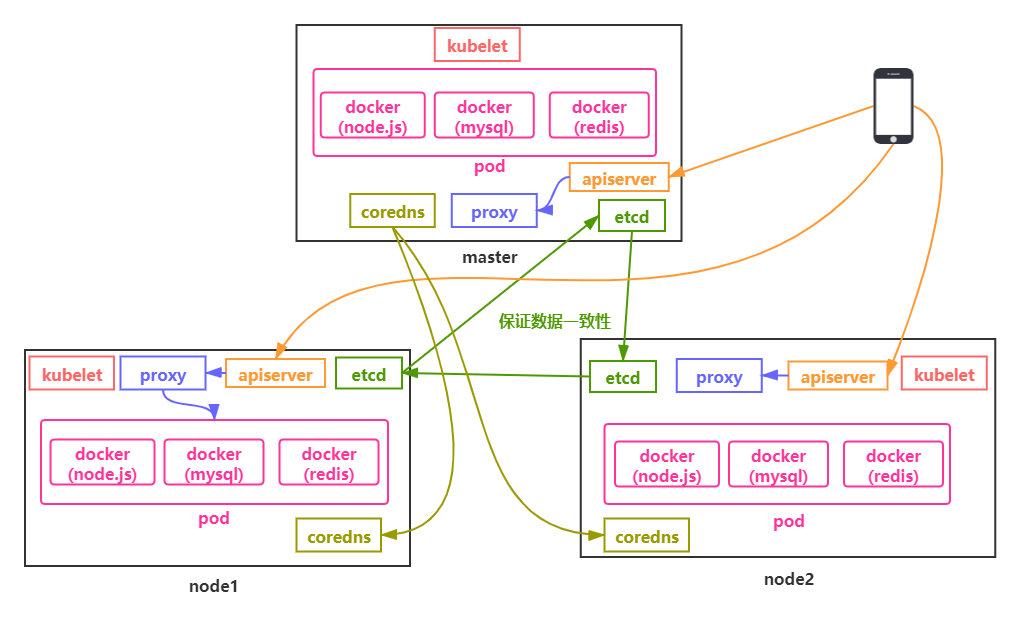
networking:

dnsDomain: cluster.local

serviceSubnet: 10.96.0.0/12+ podSubnet: 10.244.0.0/16 # 添加这个

### **3.3 拉取其它组件**

* kubeadm 可以用来拉取我们的默认组件镜像
* kube-apiserver 提供接口服务，可以让外网访问集群
* kube-controller-manager 内部的控制指令工具
* kube-scheduler 内部的任务调度器
* kube-proxy 反向代理和负载均衡，流量转发
* pause 进程管理工具
* etcd 保持 集群内部的数据一致性
* coredns 集群内网通信



// 查看缺少的组件

kubeadm config images list --config init-kubeadm.conf// 拉取缺少的组件

kubeadm config images pull --config init-kubeadm.conf

### **3.4 初始化 Kubernetes**

kubeadm init --config init-kubeadm.conf

* kubeadm join 可以快速将 Node 节点加入到 Master 集群内
* Master 节点需要执行的初始化命令
* 将默认的 Kubernetes 认证文件拷贝进 .kube 文件夹内，才能默认使用该配置文件

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 172.31.178.169:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:8aac19f4dbe68f1e15ba3d80e141acdc912e353f9757ad69187e8fb9780bc975

### **3.5 安装 Flannel**

* flannel 主要的作用是通过创建一个虚拟网络，让不同节点下的服务有着全局唯一的IP地址，且服务之前可以互相访问和连接。
* 集群内网网络通信协议通信模式采用了Flannel协议

#wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

wget https://img.zhufengpeixun.com/kube-flannel.yml

docker pull quay.io/coreos/flannel:v0.13.0-rc2

kubectl apply -f kube-flannel.yml

net-conf.json: |

{

"Network": "10.244.0.0/16",

"Backend": {

"Type": "vxlan"

}

}

### **3.6 查看启动情况**

kubectl **get** nodes

NAME STATUS ROLES AGE VERSION

master Ready control-plane,master 9m34s v1.20.4

### **3.7 Node节点配置**

* Node 节点的地位则是负责运行服务容器，负责接收调度的。
* 先执行基础安装

hostnamectl **set**-hostname node1

### **3.8 拷贝 Master 节点配置文件**

* 将 master 节点的配置文件拷贝 k8s 到 node1 节点

scp $HOME/.kube/config root@172.31.178.170:~/

* 在node1节点归档配置文件

mkdir -p $HOME/.kube

sudo mv $HOME/config $HOME/.kube/config

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

### **3.9 加入 Master 节点**

* 让 Node 节点加入到 master 集群内

kubeadm join 172.16.81.164:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:b4a059eeffa2e52f2eea7a5d592be10c994c7715c17bda57bbc3757d4f13903d

* 如果刚才的命令丢了，可以在 master 机器上使用 kubeadm token create 重新生成一条命令

kubeadm token create --print-join-command

### **3.10. 安装 Flannel**

scp ~/kube-flannel.yml root@172.31.178.170:~/

kubectl apply -f kube-flannel.yml

### **4. 查看状态**

kubectl **get** nodes

NAME STATUS ROLES AGE VERSION

master Ready control-plane,master 24m v1.20.4

node1 Ready <none> 101s v1.20.4

## **5.直接布署nginx**

kubectl create deployment nginx --image=nginx

[root@master ~]# kubectl expose deployment nginx --port=80 --type=NodePort

service/nginx exposed

kubectl get pod,svc

NAME READY STATUS RESTARTS AGE

pod/nginx-6799fc88d8-bt5n6 1/1 Running 0 5m32s

curl 127.0.0.1:32636

//快速扩容为3个副本

[root@master ~]# kubectl scale deployment nginx --replicas=3

deployment.apps/nginx scaled

## **6.通过yaml布署mysql**

### **6.1 配置文件**

apiVersion: v1kind: ReplicationController metadata:

name: mysql spec:

replicas: 1 #Pod副本的期待数量

selector:

app: mysql #符合目标的Pod拥有此标签

template: #根据此模板创建Pod的副本（实例）

metadata:

labels:

app: mysql #Pod副本拥有的标签，对应RC的Selector

spec:

containers: #Pod内容器的定义部分

- name: mysql #容器的名称

image: mysql #容器对应的Docker image

ports:

- containerPort: 3306 #容器应用监听的端口号

env: #注入容器内的环境变量

- name: MYSQL\_ROOT\_PASSWORD

value: "123456"

### **6.2 创建POD**

kubectl create -f mysql-rc.yaml

replicationcontroller/mysql created

kubectl **get** pods

NAME READY STATUS RESTARTS AGE

mysql 1/1 Running 0 5m56s

### **6.3 查看状态**

kubectl describe pod mysql