1. Kubernetes安装部署

[1. Kubernetes安装](https://note.youdao.com/ynoteshare1/iframe.html" \l "6458-1575354536287)

[1.2 基础环境准备](https://note.youdao.com/ynoteshare1/iframe.html#5516-1575354387310)

[1.2.1 硬件准备](https://note.youdao.com/ynoteshare1/iframe.html#5742-1575355937645)

[1.2.2 基础环境配置](https://note.youdao.com/ynoteshare1/iframe.html#4026-1575356226581)

[1.3 创建 single control-plane的集群](https://note.youdao.com/ynoteshare1/iframe.html#5251-1575362462944)

[1.3.1 初始化集群](https://note.youdao.com/ynoteshare1/iframe.html#8038-1575362587225)

[1.3.2 配置Kubernetes网络](https://note.youdao.com/ynoteshare1/iframe.html#9730-1575364105338)

[1.3.3 配置kubectl自动补全](https://note.youdao.com/ynoteshare1/iframe.html#3234-1575364203797)

[1.4 添加work节点到kubernetes集群](https://note.youdao.com/ynoteshare1/iframe.html#3970-1575364261077)

[1.4.1 检测Kubernetes集群是否正常](https://note.youdao.com/ynoteshare1/iframe.html#4256-1576245932336)

[1.4.2 添加Work节点](https://note.youdao.com/ynoteshare1/iframe.html#4534-1575364381928)

**1. Kubernetes安装**

**1.1 常见部署方式**

源码部署

从github上拉取kubernetes组件的源码，自行编译，生成证书，配置启动参数，安装。

优点：能快速熟悉kubernetes架构和组件构成和组件与组件之前的耦合关系。

缺点：部署起来繁琐、复杂，容易出错。

rke部署

RKE (Rancher Kubernetes Engine)是RancherLabs提供的一个工具，可以在裸机、虚拟机、公私有云上快速安装Kubernetes集群。整个集群的部署只需要一个命令、一个配置文件，解决了如何轻松部署Kubernetes的问题。

优点：使用起来方便快捷，一条命令，一个配置文件，并且能自动给kubernetes组件做HA。

缺点：全有组件都做成了docker image，部署后想定制化修改参数，需要重新改yaml文件。

kubeadm部署

kubeadm是kubernetes官方推荐的自动化部署工具，他将kubernetes的组件以pod的形式部署在master和node节点上，并自动完成证书认证等操作。因为kubeadm默认要从google的镜像仓库下载镜像，但目前国内无法访问google镜像仓库。需要提前将镜像下好，然后导入。

优点：kubernetes社区推荐部署工具，能紧根社区版本，部署起来方便快捷。

缺点：镜像在google镜像仓库，国内使用非常麻烦，没有自动的HA需要手动配置。

**1.2 基础环境准备**

**1.2.1 硬件准备**

机器配置：2核CPU，4G内存，40G系统盘

系统：Ubuntu 16.04.6 LTS

机器数量：3台 （master01 node01 node02）

**1.2.2 基础环境配置**

### 修改配置静态hostname

hostnamectl set-hostname master01

### hosts表：

# Kubernetes

172.31.53.87 master01

172.31.53.88 node01

172.31.53.86 node02

### 关闭firewalld

ufw disable

### 关闭SELinux

ubuntu默认关闭SELinux

### 关闭swap

临时关闭：swapoff -a

永久关闭：注释掉/etc/fstab下的swap一行

### 开启参数自动补全

vim /etc/bash.bashrc

取消bash-completion注释

# enable bash completion in interactive shells

if ! shopt -oq posix; then

if [ -f /usr/share/bash-completion/bash\_completion ]; then

. /usr/share/bash-completion/bash\_completion

elif [ -f /etc/bash\_completion ]; then

. /etc/bash\_completion

fi

fi

source /etc/bash.bashrc

<https://docs.docker.com/engine/install/ubuntu/>

### 更新apt包索引

sudo apt-get update

### 安装软件包以允许apt通过HTTPS使用存储库

sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ gnupg-agent \ software-properties-common

### 添加Docker的官方GPG密钥

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

### 搜索指纹的后8个字符，验证您现在是否拥有带有指纹的密钥

sudo apt-key fingerprint 0EBFCD88

### 安装add-apt-repository工具（可不用）

apt-get -y install software-properties-common

### 添加稳定的存储库

add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

### 更新apt包索引

apt-get update

### 安装最新版本

sudo apt-get install docker-ce docker-ce-cli containerd.io

### 查看Docker版本（可不执行，除非想安装特定版本）

apt-cache madison docker-ce

### 安装docker-ce=5:18.09.9~3-0~ubuntu-bionic（可不执行，除非想安装特定版本）

apt-get -y install docker-ce=5:18.09.9~3-0~ubuntu-xenial docker-ce-cli=5:18.09.9~3-0~ubuntu-xenial containerd.io

docker info（出现版本说明安装成功）

### 解决问题：WARNING: No swap limit support(操作系统下docker不支持内存限制的警告)

在基于RPM的系统上不会发生此警告，该系统默认情况下启用这些功能。

vim /etc/default/grub 添加或编辑GRUB\_CMDLINE\_LINUX行以添加这两个键值对"cgroup\_enable=memory swapaccount=1"，

最终效果：

GRUB\_CMDLINE\_LINUX="cgroup\_enable=memory swapaccount=1 net.ifnames=0 vga=792 console=tty0 console=ttyS0,115200n8 noibrs"

### 执行命令更新grub并重启机器

update-grub && reboot

### 重启docker

sudo systemctl restart docker

iptables -L

### docker 在 1.13 版本之后，将系统iptables 中 FORWARD 链的默认策略设置为 DROP，并为连接到 docker0 网桥的容器添加了ACCEPT规则，

临时解决办法：iptables -P FORWARD ACCEPT

永久解决办法：

systemctl docker status

vim /lib/systemd/system/docker.service

在[Service]下添加：ExecStartPost=/sbin/iptables -P FORWARD ACCEPT

systemctl daemon-reload && systemctl restart docker.service

k8s推荐systemd，docker推荐。。。

### 设置daemon.json

cat > /etc/docker/daemon.json <<EOF

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

systemctl daemon-reload && systemctl restart docker.service

<https://developer.aliyun.com/mirror/kubernetes?spm=a2c6h.13651102.0.0.3e221b11XhvEN2>

### 配置apt库，安装kubeadm、kubelet、kubectl

sudo apt-get update && sudo apt-get install -y apt-transport-https

curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-key add -

#curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main EOF

#cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

### 开启iptables bridge

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

**1.3 创建 single control-plane的集群**

**1.3.1 初始化集群**

### 命令详解

kubeadm config upload from-file：由配置文件上传到集群中生成ConfigMap。

kubeadm config upload from-flags：由配置参数生成ConfigMap。

kubeadm config view：查看当前集群中的配置值。

kubeadm config print init-defaults：输出kubeadm init默认参数文件的内容。

kubeadm config print join-defaults：输出kubeadm join默认参数文件的内容。

kubeadm config migrate：在新旧版本之间进行配置转换。

kubeadm config images list：列出所需的镜像列表。

kubeadm config images pull：拉取镜像到本地。

# 配置kubeadm自动补全

kubeadm completion -h

source <(kubeadm completion bash)

echo "source <(kubeadm completion bash)" >> ~/.bashrc ; su

### 初始化control-plane节点

# 生成配置文件

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

# vim init-defaults.yaml修改：clusterName:kubernetes advertiseAddress: 172.24.51.176 imageRepository: registry.aliyuncs.com/google\_containers

localAPIEndpoint:

advertiseAddress: 172.31.53.87

podSubnet: 192.168.0.0/16

clusterName: demo1

# 执行初始化操作

kubeadm init --config=init-defaults.yaml

或者

kubeadm init \

--apiserver-advertise-address=172.31.53.124 \

--pod-network-cidr=192.168.0.0/16

执行完初始化保存最后输出的结果到文件：管理用户配置、部署网络、添加节点相关信息

注意：如果在初始化集群的时候出现报错，请执行 kubeadm reset 命令执行重置，解决提示的报错后在执行初始化操作。

# 配置用户使用Kubernetes集群

## root用户

echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> /root/.bashrc

source /root/.bashrc

或者

export KUBECONFIG=/etc/kubernetes/admin.conf

## 非root用户

mkdir -p $HOME/.kube

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

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

**1.3.2 配置Kubernetes网络**

kubectl apply -f <https://docs.projectcalico.org/v3.8/manifests/calico.yaml>

kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml

或者

kubectl apply -f https://docs.projectcalico.org/v3.8/manifests/canal.yaml

root@master01:~/yaml/pods# kubectl describe nodes master01

The connection to the server 172.24.64.61:6443 was refused - did you specify the right host or port?

重新applycalico.yaml即可

**1.3.3 配置kubectl自动补全**

## 查看completion帮助

kubectl completion -h

## 若要将kubectl自动补全添加到当前shell

source <(kubectl completion bash)

## 将kubectl自动补全添加到配置文件中，可以在以后的shell中自动加载它

echo "source <(kubectl completion bash)" >> ~/.bashrc

**1.4 添加work节点到kubernetes集群**

**1.4.1 检测Kubernetes集群是否正常 ( Master01节点执行)**

# 检查组件状态是否正常

kubectl get componentstatuses

# 查看集群系统信息

kubectl cluster-info

# 查看核心组件是否运行正常（Running）

kubectl -n kube-system get pod

**1.4.2 添加Work节点**

# 创建token （Master01节点执行）

kubeadm token create

#永久token

kubeadm token create --ttl 0

# 查看token （Master01节点执行）

kubeadm token list

# 获取discovery-token-ca-cert-hash值（Master01节点执行）

openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | \ openssl dgst -sha256 -hex | sed 's/^.\* //'

# 添加work节点到kubernetes集群（work node节点执行）

kubeadm join <api-server-ip:port> --token <toke> \

--discovery-token-ca-cert-hash sha256:<discovery-token-ca-cert-hash>