**最新的稳定版kubernetes1.24.4安装部署**

1. 使用最新稳定版背景 和contianerd背景和参考官方说明
2. 节点规划
3. 初始化sysctl ,repo,modules
4. 方案一：使用 cri-docker docker
5. 方案二：使用 contianerd
6. Node节点安装 keepalived haproxy
7. 安装 kubeadm、kubelet 和 kubectl
8. 导入官网镜像
9. Kubeadm 初始化安装
10. 加入control节点 和worker节点
11. 添加集群网络组件calico
12. 验证测试群
13. 安装图形管理界面（可完成大部分命令行管理操作）
14. 截图docker模式 和 contianerd模式区别
15. 常见问题
16. **使用最新稳定版背景 和contianerd背景和参考官方说明**

最新稳定版背景：

1. 最新版稳定1.24.4支持更多新特性，修复新漏洞问题。
2. 从1.20开始，就宣布弃用 Dockershim。
3. 新版1.24开始正式弃用Dockershim。
4. 新版支持更多容器，如podman、docker、contianerd

contianerd背景：

1. 新版1.24 开始，K8S已取消docker作为其默认CRI(运行时), 默认支持 contianerd，无需Dockershim，而docker需要第三方cri-docker支持。
2. containerd 可以很好管理容器，包括容器镜像的传输和存储、容器的执行和管理、存储和网络等。
3. containerd 是从 docker 中分离出来的一个项目，命令和docker类似。
4. 旧的的k8s 的kubelet 需要先要通过 dockershim 去调用 docker，最后再通过 docker 去调用 containerd。
5. 使用 containerd 作为 K8S 容器运行时的话，由于 containerd 内置了 CRI （Container Runtime Interface：容器运行时接口）插件，kubelet 可以直接调用 containerd。

参考官方说明：

**CRI**:https://kubernetes.io/zh-cn/docs/setup/production-environment/container-runtimes/

自 1.24 版起，Dockershim 已从 Kubernetes 项目中移除。

Kubernetes 1.24支持CRI:

containerd

RI-O

cri-docker（Docker Engine）

Mirantis Container Runtime

1. **节点规划**

服务器列表 最小需要配置 主要服务 运行服务 备注

Control01 cpu2核/内存2G/硬盘50G k8s-master etcd

Control02 cpu2核/内存2G/硬盘50G k8s-master etcd

Control03 cpu2核/内存2G/硬盘51G k8s-master etcd

Node04 cpu4核/内存4G/硬盘50G k8s-node 。

Node05 cpu4核/内存4G/硬盘50G k8s-node

Node06 cpu4核/内存4G/硬盘50G k8s-node

/etc/hosts配置

192.168.80.87 node01 master01 etcd01

192.168.80.88 node02 master02 etcd02

192.168.80.89 node03 master03 etcd03

192.168.80.90 node04 slave01

192.168.80.91 node05 slave02

192.168.80.92 node06 slave03

192.168.80.93 node07 base01 harbor gitea jenkins efk prometheus

192.168.80.94 mastervip nodevip

1. **初始化sysctl ,repo,modules**

Sysctl.conf 和 repo 配置(Ansible copy到每台k8s节点

)如下：

文本

描述已自动生成

[yunwei@ansible yum.repos.d]$ cat kubernetes.repo

[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>

加载模块：

sudo modprobe overlay

sudo modprobe br\_netfilter

1. **方案一：使用 cri-docker docker**

下载CRI：<https://github.com/Mirantis/cri-dockerd>

cri-docker（Docker Engine）配置：

1. wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.2.2/cri-dockerd-0.2.2.amd64.tgz && tar -zxvf cri-dockerd-0.2.2.amd64.tgz
2. 在cri-docker.service修改 --pod-infra-container-image=docker.io/gotok8s/pause:3.7 --network-plugin=cni
3. cd cri-dockerd && cp -a cri-dockerd /usr/bin/ && ll /usr/bin/cri-dockerd
4. wget <https://github.com/Mirantis/cri-dockerd/blob/master/packaging/systemd/cri-docker.service> 并systemctl启动

docker安装：

yum -y install docker-ce 并执行启动

sudo systemctl restart docker

1. **方案二：使用 contianerd**

安装方法：yum -y install contianerd

配置CRI：

注释/etc/containerd/config.toml 中disabled\_plugins = ["cri"]，启用cri。

添加 [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]

SystemdCgroup = true

启动 sudo systemctl restart contianerd

1. **Node节点安装 keepalived haproxy**

安装: yum clean all && yum -y keepalived haproxy

配置如下图：   
文本

描述已自动生成 文本

描述已自动生成

启动：systemctl restart keepalived ; systemctl restart haproxy

1. **安装 kubeadm、kubelet 和 kubectl**

安装：yum install -y kubectl-1.24.4-0 kubeadm-1.24.4-0 kubelet-1.24.4-0 --disableexcludes=Kubernetes

echo -e 'KUBELET\_EXTRA\_ARGS=\"--cgroup-driver=systemd\"' >> /etc/sysconfig/kubelet"

启动：sudo systemctl enable --now kubelet （k8s没安装完，有错误日志提示）

1. **导入官方镜像**

Github下载地址：

https://dl.k8s.io/v1.24.4/kubernetes-server-linux-amd64.tar.gz

解压后导入镜像：

docker image load -i /tmp/kube-apiserver.tar

docker image load -i /tmp/kube-controller-manager.tar

docker image load -i /tmp/ kube-scheduler.tar

docker image load -i /tmp/kube-proxy.tar

tag镜像：

docker tag k8s.gcr.io/kube-apiserver-amd64:v1.24.4 k8s.gcr.io/kube-apiserver:v1.24.4

docker tag k8s.gcr.io/kube-controller-manager-amd64:v1.24.4 k8s.gcr.io/kube-controller-manager:v1.24.4

docker tag k8s.gcr.io/kube-scheduler-amd64:v1.24.4 k8s.gcr.io/kube-scheduler:v1.24.4

docker tag k8s.gcr.io/kube-proxy-amd64:v1.24.4 k8s.gcr.io/kube-proxy:v1.24.4

1. **Kubeadm 初始化安装**

两种安装方式（用一种即可以）：

1、命令直接初始化

kubeadm init --kubernetes-version v1.24.4 --pod-network-cidr “10.64.0.0/12” --service-cidr "10.128.0.0/12" --control-plane-endpoint "LOAD\_BALANCER\_DNS:LOAD\_BALANCER\_PORT" --upload-certs

2、生成默认配置文件：

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

**加podsubnet 和** controlPlaneEndpoint: mastervip:6443

为了实现docker使用的cgroupdriver与kubeproxy使用的mode的一致性，建议修改kubeadm init的默认配置文件 添加如下文件内容。

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: ipvs

文本

描述已自动生成

镜像下载：

ansible k8s -uroot -m shell -a "docker pull k8s.gcr.io/pause:3.7 ;docker pull k8s.gcr.io/etcd:3.5.3-0; docker pull k8s.gcr.io/coredns/coredns:v1.8.6"

初始化安装：

kubeadm init --config "/root/init-defaults.yaml" --upload-certs --v=5

文本

描述已自动生成

1. **加入control节点 和 worker节点**

Control01 和 control02节点执行：

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

--discovery-token-ca-cert-hash sha256:508078d51b7bcd2dbf3dd4620547a02cb83a6f40b05d8be9db585630f5964ff1 \

--control-plane --certificate-key 3c8de57bcd226f7b1c9e2ced0931c7392b7769e6bd85b3ea9119977e4f1f1384

文本

描述已自动生成

三台worker节点执行：

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

--discovery-token-ca-cert-hash sha256:508078d51b7bcd2dbf3dd4620547a02cb83a6f40b05d8be9db585630f5964ff1

文本

描述已自动生成

1. **添加集群网络组件calico**

没有安装网络组件时：

文本

中度可信度描述已自动生成

安装方法：

curl https://docs.projectcalico.org/manifests/calico.yaml -O

sed -ie "s/# - name: CALICO\_IPV4POOL\_CIDR/- name: CALICO\_IPV4POOL\_CIDR/g" calico.yaml

sed -ie 's?# value: "192.168.0.0/16"? value: "192.168.0.0/16"?g' calico.yaml

POD\_CIDR="10.64.0.0/12" && sed -i -e "s?192.168.0.0/16?$POD\_CIDR?g" calico.yaml

在Deployment 修改replicas数量，1到3个数量。

安装以后：

文本

中度可信度描述已自动生成

1. **验证测试群**

创建deployment，测试访问正常，集群OK

kubectl create deployment mynginx --image=nginx --replicas=2

kubectl expose deployment mynginx --port=80 --target-port=80

kubectl get svc

curl 10.142.137.79

文本

描述已自动生成

1. **安装图形界面（**可完成大部分命令行管理操作**）**

参考官网安装参考官网：

https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard

安装步骤：

wget https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.1/aio/deploy/recommended.yaml

Kubectl apply -f recommended.yaml

图形界面dashboard:

1、Creating a Service Account 和 Creating a ClusterRoleBinding（如下内容保存为yaml并执行）

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

2、Getting a Bearer Token

kubectl -n kubernetes-dashboard create token admin-user 命令输出的token保存用于登录

3、Remove the admin ServiceAccount and ClusterRoleBinding（如取消执行，这步一般无需操作执行）

kubectl -n kubernetes-dashboard delete serviceaccount admin-user

kubectl -n kubernetes-dashboard delete clusterrolebinding admin-user

4、开放对外端口 （官网用的是 kubectl proxy，我习惯采用nodport）

kubectl edit -n kubernetes-dashboard svc/kubernetes-dashboard

删除ClusterIP参数 ，修改type=NodePort,添加nodePort 8443

**登录 https://192.168.80.94:8443**

图形用户界面, 文本, 应用程序, 电子邮件

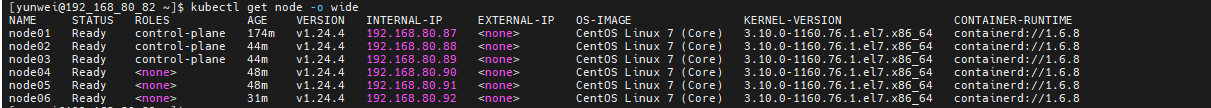
描述已自动生成

图表, 气泡图

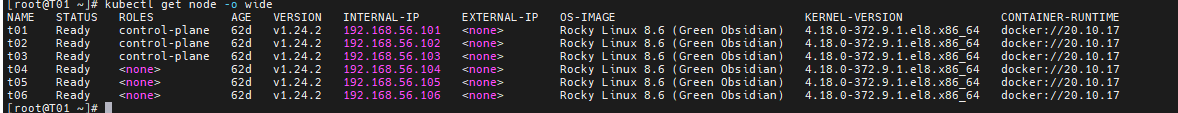
描述已自动生成

1. **截图docker模式 和 contianerd模式区别**

对kubectl命令来说没有任何区别，都是调用CRI



Docker模式如下图



1. **常见问题**

错误：\* spec.ports[0].nodePort: Invalid value: 80: provided port is not in the valid range. The range of valid ports is 30000-32767

解决： vim /etc/kubernetes/manifests/kube-apiserver.yaml 添加

- --service-node-port-range=1-65535