

kubeadm极速部署Kubernetes 1.24版本集群

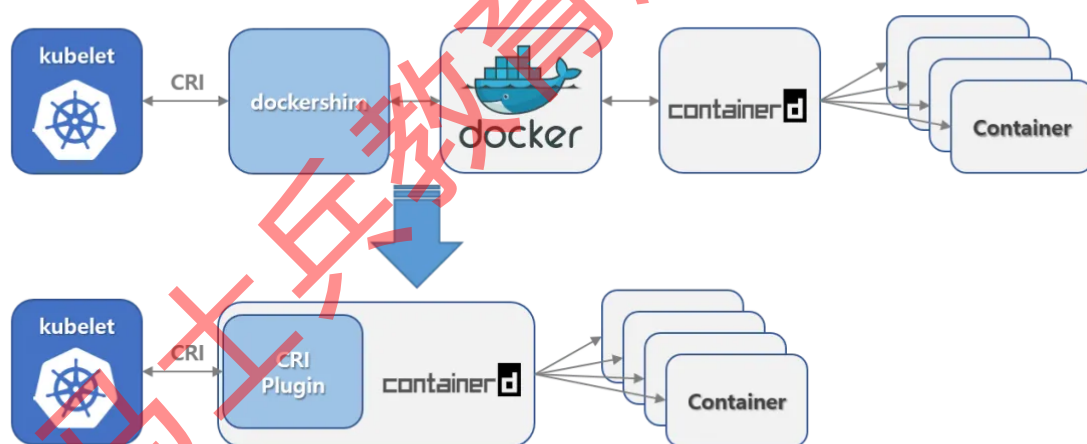
一、Kubernetes 1.24版本发布及重磅改动

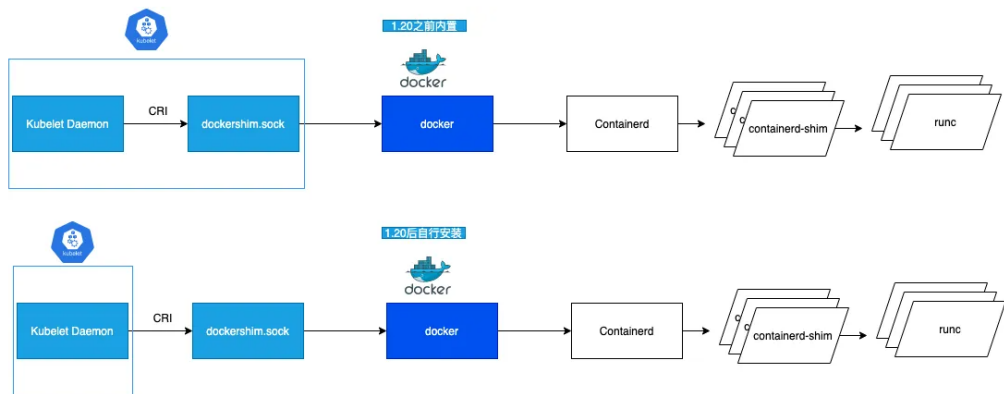
1.1 Kubernetes 1.24 发布

2022 年 5 月 3 日，Kubernetes 1.24 正式发布，在新版本中，我们看到 Kubernetes 作为容器编排的事实标准，正愈发变得成熟，有 12 项功能都更新到了稳定版本，同时引入了很多实用的功能，例如 StatefulSets 支持批量滚动更新，NetworkPolicy 新增 NetworkPolicyStatus 字段方便进行故障排查等

1.2 Kubernetes 1.24 重磅改动

Kubernetes 正式移除对 Dockershim 的支持，讨论很久的“弃用 Dockershim”也终于在这个版本画上了句号。





二、Kubernetes 1.24版本集群部署

2.1 Kubernetes 1.24版本集群部署环境准备

2.1.1 主机操作系统说明

序号	操作系统及版本	备注
1	CentOS7u9	

2.1.2 主机硬件配置说明

需求	CPU	内存	硬盘	角色	主机名
值	4C	8G	100GB	master	k8s-master01
值	4C	8G	100GB	worker(node)	k8s-worker01
值	4C	8G	100GB	worker(node)	k8s-worker02

2.1.3 主机配置

2.1.3.1 主机名配置

由于本次使用3台主机完成kubernetes集群部署，其中1台为master节点,名称为k8s-master01;其中2台为worker节点，名称分别为：k8s-worker01及k8s-worker02

```
master节点
# hostnamectl set-hostname k8s-master01
```

```
worker1节点
# hostnamectl set-hostname k8s-worker01
```

```
worker2节点
# hostnamectl set-hostname k8s-worker02
```

2.1.3.2 主机IP地址配置

```
k8s-master节点IP地址为: 192.168.10.200/24
# vim /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="ens33"
DEVICE="ens33"
ONBOOT="yes"
IPADDR="192.168.10.200"
PREFIX="24"
GATEWAY="192.168.10.2"
DNS1="119.29.29.29"
```

```
k8s-worker1节点IP地址为: 192.168.10.201/24
# vim /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="ens33"
DEVICE="ens33"
ONBOOT="yes"
IPADDR="192.168.10.201"
PREFIX="24"
```

```
GATEWAY="192.168.10.2"  
DNS1="119.29.29.29"
```

```
k8s-worker2节点IP地址为: 192.168.10.202/24  
# vim /etc/sysconfig/network-scripts/ifcfg-ens33  
TYPE="Ethernet"  
PROXY_METHOD="none"  
BROWSER_ONLY="no"  
BOOTPROTO="none"  
DEFROUTE="yes"  
IPV4_FAILURE_FATAL="no"  
IPV6INIT="yes"  
IPV6_AUTOCONF="yes"  
IPV6_DEFROUTE="yes"  
IPV6_FAILURE_FATAL="no"  
IPV6_ADDR_GEN_MODE="stable-privacy"  
NAME="ens33"  
DEVICE="ens33"  
ONBOOT="yes"  
IPADDR="192.168.10.202"  
PREFIX="24"  
GATEWAY="192.168.10.2"  
DNS1="119.29.29.29"
```

2.1.3.3 主机名与IP地址解析

所有集群主机均需要进行配置。

```
# cat /etc/hosts  
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.10.200 k8s-master01  
192.168.10.201 k8s-worker01  
192.168.10.202 k8s-worker02
```

2.1.3.4 防火墙配置

所有主机均需要操作。

```
关闭现有防火墙firewalld
# systemctl disable firewalld
# systemctl stop firewalld
# firewall-cmd --state
not running
```

2.1.3.5 SELINUX配置

所有主机均需要操作。修改SELinux配置需要重启操作系统。

```
# sed -ri 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
```

2.1.3.6 时间同步配置

所有主机均需要操作。最小化安装系统需要安装ntpd软件。

```
# crontab -l
0 */1 * * * /usr/sbin/ntpdate time1.aliyun.com
```

2.1.3.7 升级操作系统内核

所有主机均需要操作。

```
导入elrepo gpg key
# rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
```

安装elrepo YUM源仓库

```
# yum -y install https://www.elrepo.org/elrepo-release-7.0-4.el7.elrepo.noarch.rpm
```

安装kernel-ml版本，ml为长期稳定版本，lt为长期维护版本

```
# yum --enablerepo="elrepo-kernel" -y install kernel-ml.x86_64
```

设置grub2默认引导为0
grub2-set-default 0

重新生成grub2引导文件
grub2-mkconfig -o /boot/grub2/grub.cfg

更新后，需要重启，使用升级的内核生效。
reboot

重启后，需要验证内核是否为更新对应的版本
uname -r

2.1.3.8 配置内核转发及网桥过滤

所有主机均需要操作。

添加网桥过滤及内核转发配置文件
cat /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
vm.swappiness = 0

加载br_netfilter模块
modprobe br_netfilter

查看是否加载
lsmod | grep br_netfilter
br_netfilter 22256 0
bridge 151336 1 br_netfilter

加载网桥过滤及内核转发配置文件
sysctl -p /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
vm.swappiness = 0

2.1.3.9 安装ipset及ipvsadm

所有主机均需要操作。

```
安装ipset及ipvsadm
# yum -y install ipset ipvsadm
```

配置ipvsadm模块加载方式

添加需要加载的模块

```
# cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack
EOF
```

授权、运行、检查是否加载

```
# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip_vs -e nf_conntrack
```

2.1.3.10 关闭SWAP分区

修改完成后需要重启操作系统，如不重启，可临时关闭，命令为swapoff -a

永远关闭swap分区，需要重启操作系统

```
# cat /etc/fstab
```

```
.....
```

```
# /dev/mapper/centos-swap swap swap defaults 0 0
```

在上一行中行首添加#

2.2 Docker准备

2.2.1 Docker安装YUM源准备

使用阿里云开源软件镜像站。

```
# wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -O
/etc/yum.repos.d/docker-ce.repo
```

2.2.2 Docker安装

```
# yum -y install docker-ce
```

2.2.3 启动Docker服务

```
# systemctl enable --now docker
```

2.2.4 修改cgroup方式

/etc/docker/daemon.json 默认没有此文件，需要单独创建

在/etc/docker/daemon.json添加如下内容

```
# cat /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

```
# systemctl restart docker
```

2.2.5 cri-dockerd安装

2.2.5.1 golang环境准备

下载链接地址: <https://golang.google.cn/dl/>

获取golang安装包

```
# wget https://golang.google.cn/dl/go1.16.10.linux-amd64.tar.gz
```

解压golang至指定目录

```
# tar -xzf go1.16.10.linux-amd64.tar.gz -C /usr/local
```


添加环境变量

```
# cat /etc/profile  
  
.....  
export GOROOT=/usr/local/go  
export GOPATH=$HOME/go  
export PATH=$PATH:$GOROOT/bin:$GOPATH/bin
```

加载/etc/profile文件

```
# source /etc/profile
```

验证golang是否安装完成

```
# go version
```

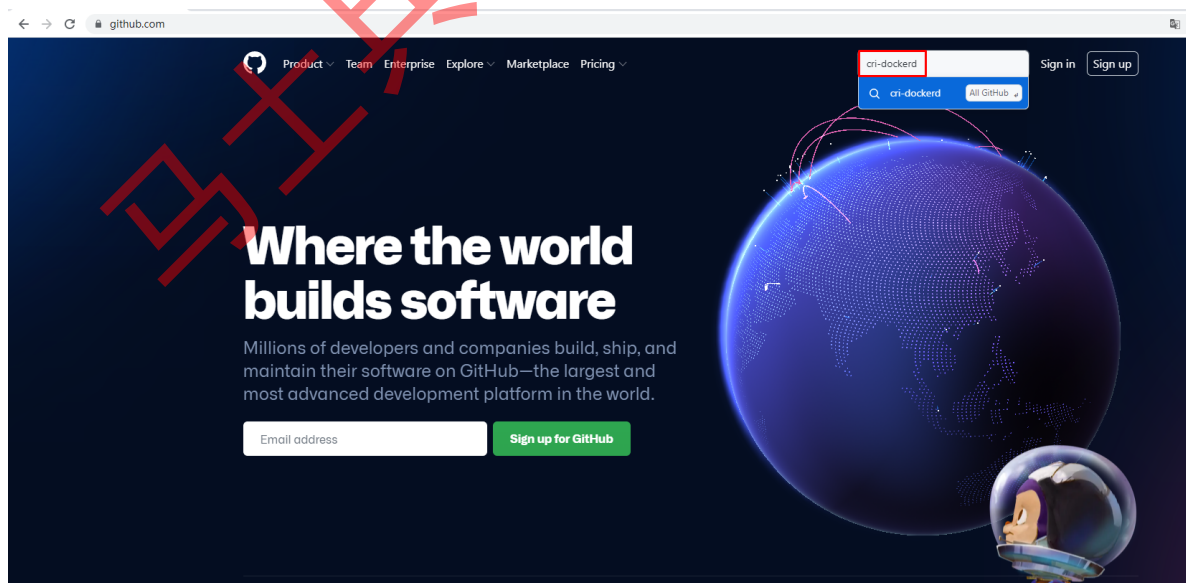
输出

```
go version go1.16.10 linux/amd64
```

创建gopath目录

```
# mkdir -p ~/go/bin ~/go/src ~/go/pkg
```

2.2.5.2 构建并安装cri-dockerd



← → ↻ github.com/search?q=cri-dockerd

Product Team Enterprise Explore Marketplace Pricing cri-dockerd

Repositories 4

Code ?

Commits 498

Issues 900

Discussions 2

Packages 0

Marketplace 0

Topics 0

Wikis 8

Users 0

Languages

Go 2

Shell 1

Advanced search Cheat sheet

4 repository results

Mirantis/cri-dockerd

☆ 129 Go Apache-2.0 license Updated 15 days ago

dims/cri-dockerd

☆ 15 Go Updated on 25 Jun 2020

klts-io/setup-cri-dockerd

It is very easy to switch from Docker Shim to CRI Dockerd and back

☆ 6 Shell Apache-2.0 license Updated on 7 Mar

diannaowa/cri-dockerd-demo

基于K8S 1.19

Updated on 17 Jul 2021

← → ↻ github.com/Mirantis/cri-dockerd

Product Team Enterprise Explore Marketplace Pricing Search Sign in Sign up

Mirantis / cri-dockerd Public

<> Code Issues 8 Pull requests 4 Actions Projects Wiki Security Insights

master 2 branches 1 tag

Go to file Code

About

evol262 Merge pull request #34 from Mirantis/ldflag-rename

✓ 0607a1b on 5 Feb 871 commits

.github/workflows

Allow overwrite

4 months ago

packaging

Rename LDFlags

3 months ago

src

Parameterize more Makefile variables, add a --buildinfo flag

4 months ago

.gitignore

Fix terrible interface name

5 months ago

LICENSE

Add license file

5 months ago

Makefile

Rename LDFlags

3 months ago

README.md

Update the name of the systemd unit files

4 months ago

VERSION

Packaging updates for release

5 months ago

README.md

cri-dockerd

This adapter provides a shim for [Docker Engine](#) that lets you control Docker via the Kubernetes [Container Runtime interface](#).

Motivation

Releases 1

v0.2.0 Latest

on 4 Jan

Packages

No packages published

Contributors 156

+ 145 contributors

github.com/Mirantis/cri-dockerd

Product Team Enterprise Explore Marketplace Pricing

Search Sign in Sign up

Mirantis / cri-dockerd Public

Code Issues 8 Pull requests 4 Actions Projects Wiki Security Insights

master 2 branches 1 tag

Go to file Code

evol262 Merge pull request #34 from Mirantis/ldflag-rename

.github/workflows	Allow overwrite
packaging	Rename LDFlags
src	Parameterize more Makefile variables
.gitignore	Fix terrible interface name
LICENSE	Add license file
Makefile	Rename LDFlags
README.md	Update the name of the systemd unit files 4 months ago
VERSION	Packaging updates for release 5 months ago

Clone

HTTPS GitHub CLI

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

Use Git or checkout with SVN using the web URL

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

Apache-2.0 license

138 stars

14 watching

32 forks

Releases 1

v0.2.0 Latest on 4 Jan

Packages

No packages published

Contributors 156

cri-dockerd

This adapter provides a shim for Docker Engine that lets you control Docker via the Kubernetes Container Runtime Interface.

克隆cri-dockerd源码

```
# git clone https://github.com/Mirantis/cri-dockerd.git
```

查看克隆下来的目录

```
# ls
cri-dockerd
```

查看目录中内容

```
# ls cri-dockerd/
LICENSE  Makefile  packaging  README.md  src  VERSION
```

```
# cd cri-dockerd
```

创建bin目录并构建cri-dockerd二进制文件

```
# mkdir bin
# cd src && go get && go build -o ../bin/cri-dockerd
```

创建/usr/local/bin,默认存在时,可不用创建

```
# mkdir -p /usr/local/bin
```

安装cri-dockerd

```
# install -o root -g root -m 0755 bin/cri-dockerd /usr/local/bin/cri-dockerd
```

复制服务管理文件至/etc/systemd/system目录中

```
# cp -a packaging/systemd/* /etc/systemd/system
```

指定cri-dockerd运行位置

```
#sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,'  
/etc/systemd/system/cri-docker.service
```

启动服务

```
# systemctl daemon-reload
```

```
# systemctl enable cri-docker.service
```

```
# systemctl enable --now cri-docker.socket
```

2.3 kubernetes 1.24.0 集群部署

2.3.1 集群软件及版本说明

	kubeadm	kubelet	kubectl
版本	1.24.0	1.24.0	1.24.0
安装位置	集群所有主机	集群所有主机	集群所有主机
作用	初始化集群、管理集群等	用于接收api-server指令，对pod生命周期进行管理	集群应用命令行管理工具

2.3.2 kubernetes YUM源准备

2.3.2.1 谷歌YUM源

```
[kubernetes]  
name=kubernetes  
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64  
enabled=1  
gpgcheck=0  
repo_gpgcheck=0  
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg  
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

2.3.2.2 阿里云YUM源

```
[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
```

2.3.3 集群软件安装

所有节点均可安装

安装

```
# yum -y install kubeadm kubelet kubect1
```

2.3.4 配置kubelet

为了实现docker使用的cgroupdriver与kubelet使用的cgroup的一致性，建议修改如下文件内容。

```
# vim /etc/sysconfig/kubelet
KUBELET_EXTRA_ARGS="--cgroup-driver=systemd"
```

设置kubelet为开机自启动即可，由于没有生成配置文件，集群初始化后自动启动

```
# systemctl enable kubelet
```

2.3.5 集群镜像准备

可使用VPN实现下载。

```
# kubeadm config images list --kubernetes-version=v1.24.0
k8s.gcr.io/kube-apiserver:v1.24.0
k8s.gcr.io/kube-controller-manager:v1.24.0
k8s.gcr.io/kube-scheduler:v1.24.0
k8s.gcr.io/kube-proxy:v1.24.0
k8s.gcr.io/pause:3.7
k8s.gcr.io/etcd:3.5.3-0
k8s.gcr.io/coredns/coredns:v1.8.6
```

```
# cat image_download.sh
#!/bin/bash
images_list='
k8s.gcr.io/kube-apiserver:v1.24.0
k8s.gcr.io/kube-controller-manager:v1.24.0
k8s.gcr.io/kube-scheduler:v1.24.0
k8s.gcr.io/kube-proxy:v1.24.0
k8s.gcr.io/pause:3.7
k8s.gcr.io/etcd:3.5.3-0
k8s.gcr.io/coredns/coredns:v1.8.6'

for i in $images_list
do
    docker pull $i
done

docker save -o k8s-1-24-0.tar $images_list
```

2.3.6 集群初始化

```
[root@k8s-master01 ~]# kubeadm init --kubernetes-version=v1.24.0 --pod-network-cidr=10.224.0.0/16 --apiserver-advertise-address=192.168.10.200 --cri-socket unix:///var/run/cri-dockerd.sock
```

如果不添加--cri-socket选项，则会报错，内容如下：

Found multiple CRI endpoints on the host. Please define which one you wish to use by setting the 'criSocket' field in the kubeadm configuration file:
 unix:///var/run/containerd/containerd.sock, unix:///var/run/cri-dockerd.sock
 To see the stack trace of this error execute with --v=5 or higher

初始化过程输出

```
[init] Using Kubernetes version: v1.24.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [k8s-master01 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.10.200]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
```

```
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [k8s-master01
localhost] and IPs [192.168.10.200 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8s-master01 localhost]
and IPs [192.168.10.200 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in
"/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as
static pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 13.006785 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in
the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with
the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node k8s-master01 as control-plane by adding
the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-
from-external-load-balancers]
[mark-control-plane] Marking the node k8s-master01 as control-plane by adding
the taints [node-role.kubernetes.io/master:NoSchedule node-
role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 8x4o2u.hs1o8xzwwlrncr8s
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC
roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get
nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post
CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller
automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all
node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public"
namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a
rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy
```

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 192.168.10.200:6443 --token 8x4o2u.hs1o8xzw1rncr8s \
--discovery-token-ca-cert-hash
sha256:7323a8b0658fc33d89e627f078f6eb16ac94394f9a91b3335dd3ce73a3f313a0
```

2.3.7 集群应用客户端管理集群文件准备

```
[root@k8s-master01 ~]# mkdir -p $HOME/.kube
[root@k8s-master01 ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master01 ~]# chown $(id -u):$(id -g) $HOME/.kube/config
[root@k8s-master01 ~]# ls /root/.kube/
config
```

```
[root@k8s-master01 ~]# export KUBECONFIG=/etc/kubernetes/admin.conf
```

2.3.8 集群网络准备

使用calico部署集群网络

安装参考网址: <https://projectcalico.docs.tigera.io/about/about-calico>

2.3.8.1 calico安装

About

Calico

Networking

Kubernetes

Networking

Network Policy

Kubernetes Services

Kubernetes Ingress

Kubernetes Egress

eBPF

Install Calico

用于安装

Networking

ABOUT / CALICO

Edit this page

4 MINUTE READ



What is Calico?

Calico is an open source networking and network security solution for containers, virtual machines, and native host-based workloads. Calico supports a broad range of platforms including Kubernetes, OpenShift, Mirantis Kubernetes Engine (MKE), OpenStack, and bare metal services.

Whether you opt to use Calico's eBPF data plane or Linux' s standard networking pipeline, Calico delivers blazing fast performance with true cloud-native scalability. Calico provides developers and cluster operators with a consistent experience and set of capabilities whether running in public cloud or on-prem, on a single node, or across a multi-thousand node cluster.

eBPF

ABOUT / CALICO

4 MINUTE READ

Install Calico

Kubernetes

OpenStack

Non-cluster hosts

Networking


Security

Operations

Reference

Release notes



Project CalicoCommunitySupportGitHubGet CertifiedVersion 3.21Search

About

Install Calico

Kubernetes

OpenStack

Non-cluster hosts

Networking

Security

Operations

Reference

Release notes

INSTALL CALICO / KUBERNETESEdit this page

Kubernetes

Get Calico up and running in your Kubernetes cluster.

Quickstart

Managed public cloud

Self-managed public cloud

Self-managed on-premises


OpenShift


Rancher Kubernetes Engine

Flannel

Calico for Windows

K3s

TIGERAProductsSolutionsLearnCompany

Project CalicoCommunitySupportGitHubGet CertifiedVersion 3.21

About

Install Calico

Kubernetes

Quickstart

Managed public cloud

Self-managed public cloud

Self-managed on-premises

OpenShift

Rancher Kubernetes Engine

Install Calico

1. Install the Tigera Calico operator and custom resource definitions.

```
kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml
```
2. Install Calico by creating the necessary custom resource. For more information on configuration options available in this manifest, see [the installation reference](#).

```
kubectl create -f https://docs.projectcalico.org/manifests/custom-resources.yaml
```

Note: Before creating this manifest, read its contents and make sure its settings are correct for your environment. For example, you may need to change the default IP pool CIDR to match your pod network CIDR.
3. Confirm that all of the pods are running with the following command.

```
watch kubectl get pods -n calico-system
```

Wait until each pod has the **STATUS** of **Running**.

下载operator资源清单文件

```
[root@k8s-master01 ~]# wget https://docs.projectcalico.org/manifests/tigera-operator.yaml
```

应用资源清单文件，创建operator

```
[root@k8s-master01 ~]# kubectl apply -f tigera-operator.yaml
```

通过自定义资源方式安装

```
[root@k8s-master01 ~]# wget https://docs.projectcalico.org/manifests/custom-resources.yaml
```

修改文件第13行，修改为使用kubeadm init ----pod-network-cidr对应的IP地址段

```
[root@k8s-master01 ~]# vim custom-resources.yaml
```

```
.....
11     ipPools:
12     - blockSize: 26
13       cidr: 10.224.0.0/16
14       encapsulation: VXLANCrossSubnet
.....
```

应用资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f custom-resources.yaml
```

监视calico-system命名空间中pod运行情况

```
[root@k8s-master01 ~]# watch kubectl get pods -n calico-system
```

Wait until each pod has the STATUS of Running.

删除 master 上的 taint

```
[root@k8s-master01 ~]# kubectl taint nodes --all node-role.kubernetes.io/master-
```

已经全部运行

```
[root@k8s-master01 ~]# kubectl get pods -n calico-system
```

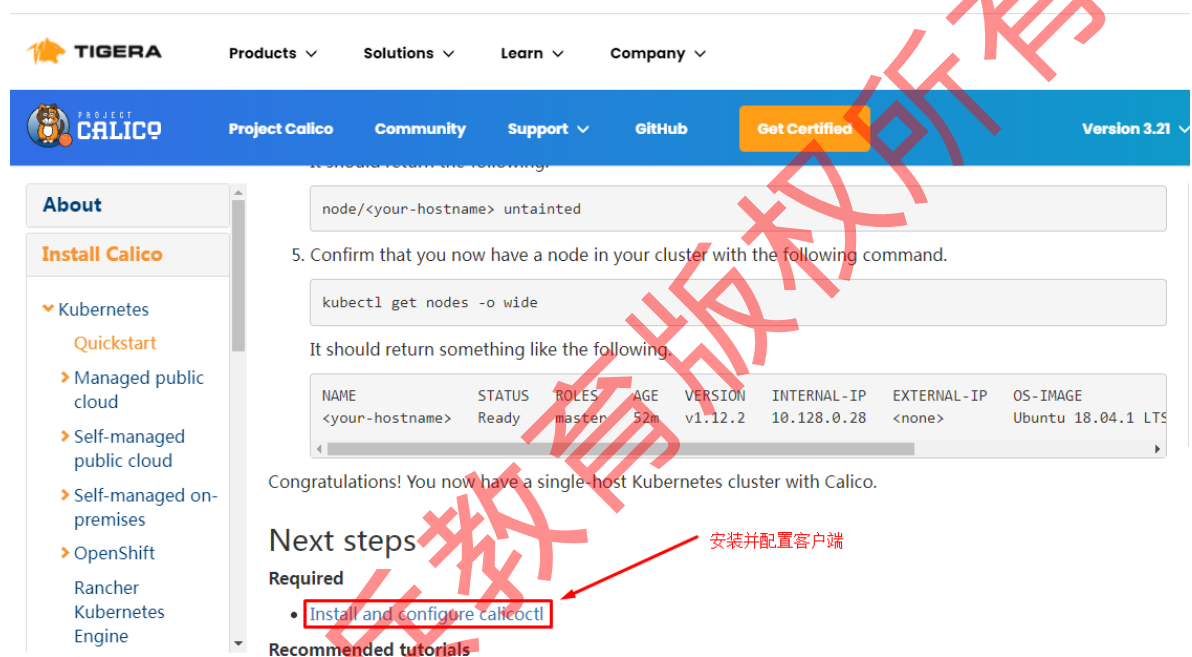
NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-666bb9949-dzp68	1/1	Running	0	11m
calico-node-jhcf4	1/1	Running	4	11m
calico-typha-68b96d8d9c-7qfq7	1/1	Running	2	11m

查看kube-system命名空间中coredns状态，处于Running状态表明联网成功。

```
[root@k8s-master01 ~]# kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-6d4b75cb6d-js5p1	1/1	Running	0	12h
coredns-6d4b75cb6d-zm8pc	1/1	Running	0	12h
etcd-k8s-master01	1/1	Running	0	12h
kube-apiserver-k8s-master01	1/1	Running	0	12h
kube-controller-manager-k8s-master01	1/1	Running	0	12h
kube-proxy-7nhr7	1/1	Running	0	12h
kube-proxy-fv4kr	1/1	Running	0	12h
kube-proxy-vv5vg	1/1	Running	0	12h
kube-scheduler-k8s-master01	1/1	Running	0	12h

2.3.8.2 calico客户端安装



The screenshot shows the Tigera Project Calico website. The sidebar on the left has a link to 'Install Calico' under the 'Kubernetes' section. The main content area shows the following steps:

- node/<your-hostname> untainted
5. Confirm that you now have a node in your cluster with the following command.
kubectl get nodes -o wide
- It should return something like the following.

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE
<your-hostname>	Ready	master	52m	v1.12.2	10.128.0.28	<none>	Ubuntu 18.04.1 LTS

Congratulations! You now have a single-host Kubernetes cluster with Calico.

Next steps

- Required**
 - Install and configure calicoctl**

Recommended tutorials

安装并配置客户端

About

Install Calico

Networking

Security

Operations

➤ Upgrade

➤ calicoctl

Install calicoctl

➤ Configure calicoctl

➤ Deploy image options

Migrate datastores

Install calicoctl as a binary on a single host

Linux

Mac OSX

Windows

Linux PPC64le

Linux arm64

1. Log into the host, open a terminal prompt, and navigate to the location where you want to install the binary.

Tip: Consider navigating to a location that's in your `PATH`. For example, `/usr/local/bin/`.

2. Use the following command to download the `calicoctl` binary.

```
$ curl -L https://github.com/projectcalico/calico/releases/download/v3.21.4/calicoctl-linux-amd64
```

3. Set the file to be executable.

```
$ chmod +x ./calicoctl
```

Note: If the location of `calicoctl` is not already in your `PATH`, move the file to one that is or add its location to your `PATH`. This will allow you to invoke it without having to prepend its location.

下载二进制文件

```
# curl -L  
https://github.com/projectcalico/calico/releases/download/v3.21.4/calicoctl-  
linux-amd64 -o calicoctl
```

安装calicoctl

```
# mv calicoctl /usr/bin/
```

为calicoctl添加可执行权限

```
# chmod +x /usr/bin/calicoctl
```

查看添加权限后文件

```
# ls /usr/bin/calicoctl  
/usr/bin/calicoctl
```

查看calicoctl版本

```
# calicoctl version
```

Client Version: v3.21.4

Git commit: 220d04c94

Cluster Version: v3.21.4

Cluster Type: typha,k8s,operator,bgp,kubeadm

通过`~/ .kube/config`连接kubernetes集群，查看已运行节点

```
# DATASTORE_TYPE=kubernetes KUBECONFIG=~/ .kube/config calicoctl get nodes  
NAME  
k8s-master01
```

2.3.9 集群工作节点添加

因容器镜像下载较慢，可能会导致报错，主要错误为没有准备好cni（集群网络插件），如有网络，请耐心等待即可。

```
[root@k8s-worker01 ~]# kubeadm join 192.168.10.200:6443 --token  
8x4o2u.hs1o8xzwwlrncr8s \                               --discovery-token-ca-  
cert-hash  
sha256:7323a8b0658fc33d89e627f078f6eb16ac94394f9a91b3335dd3ce73a3f313a0 --cri-  
socket unix:///var/run/cni-dockerd.sock
```

```
[root@k8s-worker02 ~]# kubeadm join 192.168.10.200:6443 --token  
8x4o2u.hs1o8xzwwlrncr8s \                               --discovery-token-ca-  
cert-hash  
sha256:7323a8b0658fc33d89e627f078f6eb16ac94394f9a91b3335dd3ce73a3f313a0 --cri-  
socket unix:///var/run/cni-dockerd.sock
```

在master节点上操作，查看网络节点是否添加

```
# DATASTORE_TYPE=kubernetes KUBECONFIG=~/ .kube/config calicoctl get nodes  
NAME  
k8s-master01  
k8s-worker01  
k8s-worker02
```

2.3.10 验证集群可用性

查看所有的节点

```
[root@k8s-master01 ~]# kubectl get nodes  
NAME             STATUS    ROLES    AGE   VERSION  
k8s-master01     Ready    control-plane   12h   v1.24.0  
k8s-worker01     Ready    <none>         12h   v1.24.0  
k8s-worker02     Ready    <none>         12h   v1.24.0
```

查看集群健康情况

```
[root@k8s-master01 ~]# kubectl get cs
```

Warning: v1 ComponentStatus is deprecated in v1.19+

NAME	STATUS	MESSAGE	ERROR
controller-manager	Healthy	ok	
scheduler	Healthy	ok	
etcd-0	Healthy	{ "health": "true", "reason": "" }	

查看kubernetes集群pod运行情况

```
[root@k8s-master01 ~]# kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-6d4b75cb6d-js5p1	1/1	Running	0	12h
coredns-6d4b75cb6d-zm8pc	1/1	Running	0	12h
etcd-k8s-master01	1/1	Running	0	12h
kube-apiserver-k8s-master01	1/1	Running	0	12h
kube-controller-manager-k8s-master01	1/1	Running	0	12h
kube-proxy-7nhr7	1/1	Running	0	12h
kube-proxy-fv4kr	1/1	Running	0	12h
kube-proxy-vv5vg	1/1	Running	0	12h
kube-scheduler-k8s-master01	1/1	Running	0	12h

再次查看calico-system命名空间中pod运行情况。

```
[root@k8s-master01 ~]# kubectl get pods -n calico-system
```

NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-5b544d9b48-xgfnk	1/1	Running	0	12h
calico-node-7c1f4	1/1	Running	0	12h
calico-node-cjwns	1/1	Running	0	12h
calico-node-hhr4n	1/1	Running	0	12h
calico-typha-6cb6976b97-51npk	1/1	Running	0	12h
calico-typha-6cb6976b97-9w9s8	1/1	Running	0	12h