

K8S 1.16版本 (<https://www.kubernetes.org.cn/tags/kubernetes-1-16>) 1.15 (<https://www.kubernetes.org.cn/tags/kubernetes-1-15>) 1.14 (<https://www.kubernetes.org.cn/tags/kubernetes1-14>)
CI/CD (<https://www.kubernetes.org.cn/tags/cicd>) 网络 (<https://www.kubernetes.org.cn/tags/网络>) 存储 (<https://www.kubernetes.org.cn/tags/存储>) 安全 (<https://www.kubernetes.org.cn/tags/安全>)
监控 (<https://www.kubernetes.org.cn/tags/监控>) 输入关键字 (<http://docs.kubernetes.org.cn/683.html>) 关注本站 
登录 (<https://www.kubernetes.org.cn/wp-login.php?cms=k8s>) | 加入我们 (<https://www.kubernetes.org.cn/%e8%81%94%e7%b3%bb%e6%88%91%e4%bb%ac>)

[开发实践](https://www.kubernetes.org.cn/practice) (<https://www.kubernetes.org.cn/practice>) [行业动态](https://www.kubernetes.org.cn/news) (<https://www.kubernetes.org.cn/news>) [入门教程](https://www.kubernetes.org.cn/course) (<https://www.kubernetes.org.cn/course>)
[安装教程](https://www.kubernetes.org.cn/course/install) (<https://www.kubernetes.org.cn/course/install>) [文档下载](https://www.kubernetes.org.cn/%E6%96%87%E6%A1%A3%E4%B8%8B%E8%BD%BD) (<https://www.kubernetes.org.cn/%E6%96%87%E6%A1%A3%E4%B8%8B%E8%BD%BD>)
[QQ/微信群](https://www.kubernetes.org.cn/kubernetes) (<https://www.kubernetes.org.cn/kubernetes>) [Kubernetes交流群](https://www.kubernetes.org.cn/video) (<https://www.kubernetes.org.cn/video>) [视频](https://www.kubernetes.org.cn/meetup) (<https://www.kubernetes.org.cn/video>) [活动](https://www.kubernetes.org.cn/meetup) (<https://www.kubernetes.org.cn/meetup>)
[中文文档](http://docs.kubernetes.org.cn) (<http://docs.kubernetes.org.cn>) 

Kubernetes线下实战培训 3天突击CKA

官方认证CKA讲师、实操环境实战、现场答疑互动

上海（11月15-17日） / 深圳（11月22-24日） / 武汉 / 成都 / 北京 /

点击查看

<https://www.kubernetes.org.cn/peixun>

您目前处于：[社区首页](https://www.kubernetes.org.cn) (<https://www.kubernetes.org.cn>) > [Kubernetes安装说明](https://www.kubernetes.org.cn/course/install) (<https://www.kubernetes.org.cn/course/install>) >

Kubernetes 1.8.x 全手动安装教程

Kubernetes 1.8.x 全手动安装教程

(<https://www.kubernetes.org.cn/3096.html>)

2017-11-03 09:13 Kyle.Bai (<https://www.kubernetes.org.cn/author/kyle-bai>) 分类：[Kubernetes安装说明](https://www.kubernetes.org.cn/course/install) (<https://www.kubernetes.org.cn/course/install>) 阅读(86313) 作者：Kyle.Bai / [GitHub](https://github.com/kairan) (<https://github.com/kairan>) 评论(46)



Kubernetes 提供了许多云端平台与操作系统的安装方式，本章将以全手动安装方式来部署，主要是学习与了解 Kubernetes 创建流程。若想要了解更多平台的部署可以参考 [Picking the Right Solution](http://docs.kubernetes.org.cn/282.html) (<http://docs.kubernetes.org.cn/282.html>)来选择自己最喜欢的方式。

本次安装版本为：

- Kubernetes v1.8.2
- Etcd v3.2.9
- Calico v2.6.2
- Docker v17.10.0-ce

预先准备信息

本教程将以下列节点数与规格来进行部署 Kubernetes 集群，操作系统可采用Ubuntu 16.x与CentOS 7.x：

IP Address	Role	CPU	Memory
172.16.35.12	master1	1	2G
172.16.35.10	node1	1	2G

<https://www.kubernetes.org.cn/3096.html>

热门推荐

- 

Docker Hub上镜像发现挖矿蠕虫病毒，已导致2000台主机感染

2019-10-19

<https://www.kubernetes.org.cn/5951>
- 

K8s 工程师必懂的 10 种 Ingress 控制器

2019-10-18

<https://www.kubernetes.org.cn/5948>
- 

Kubernetes应用之道：让 Kubernetes落地的“三板斧”

2019-09-24

<https://www.kubernetes.org.cn/5872>
- 

Kubernetes v1.16 重磅发布！

2019-09-19

<https://www.kubernetes.org.cn/5838>
- 

容器该不该放到VM中？

2019-08-24

<https://www.kubernetes.org.cn/5765>
- 

美团点评Kubernetes集群管理实践

2019-08-22

<https://www.kubernetes.org.cn/5747>

社区标签

- BoCloud博云
(<https://www.kubernetes.org.cn/tags/bocloud%e5%8d%9a%e>)
[calico](https://www.kubernetes.org.cn/tags/calico) (<https://www.kubernetes.org.cn/tags/calico>)
- CI/CD (<https://www.kubernetes.org.cn/tags/cicd>)
- CNCF
(<https://www.kubernetes.org.cn/tags/cncf>)
[CoreOS](https://www.kubernetes.org.cn/tags/coreos) (<https://www.kubernetes.org.cn/tags/coreos>)
- DevOps
(<https://www.kubernetes.org.cn/tags/devop>)


```
$ export CFSSL_URL="https://pkg.cfssl.org/R1.2"
$ wget "${CFSSL_URL}/cfssl_linux-amd64" -O /usr/local/bin/cfssl
$ wget "${CFSSL_URL}/cfssljson_linux-amd64" -O /usr/local/bin/cfssljson
$ chmod +x /usr/local/bin/cfssl /usr/local/bin/cfssljson
```

Etdc

在开始安装 Kubernetes 之前，需要先将一些必要系统创建完成，其中 Etdc 就是 Kubernetes 最重要的一环，Kubernetes 会将大部分信息储存于 Etdc 上，来提供给其他节点索取，以确保整个集群运作与沟通正常。

创建集群 CA 与 Certificates

在这部分，将会需要产生 client 与 server 的各组件 certificates，并且替 Kubernetes admin user 产生 client 证书。

建立/etc/etcd/ssl文件夹，然后进入目录完成以下操作。

```
$ mkdir -p /etc/etcd/ssl && cd /etc/etcd/ssl
$ export PKI_URL="https://kaiaren.github.io/files/manual-v1.8/pki"
```

下载ca-config.json与etcd-ca-csr.json文件，并产生 CA 密钥：

```
$ wget "${PKI_URL}/ca-config.json" "${PKI_URL}/etcd-ca-csr.json"
$ cfssl gencert -initca etcd-ca-csr.json | cfssljson -bare etcd-ca
$ ls etcd-ca*.pem
etcd-ca-key.pem  etcd-ca.pem
```

下载etcd-csr.json文件，并产生 kube-apiserver certificate 证书：

```
$ wget "${PKI_URL}/etcd-csr.json"
$ cfssl gencert \
  -ca=etcd-ca.pem \
  -ca-key=etcd-ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  etcd-csr.json | cfssljson -bare etcd

$ ls etcd*.pem
etcd-ca-key.pem  etcd-ca.pem  etcd-key.pem  etcd.pe
```

若节点 IP 不同，需要修改etcd-csr.json的hosts。

完成后删除不必要文件：

```
$ rm -rf *.json
```

确认/etc/etcd/ssl有以下文件：

```
$ ls /etc/etcd/ssl
etcd-ca.csr  etcd-ca-key.pem  etcd-ca.pem  etcd.csr  etcd-key.pem  etcd.pem
```

Etdc 安装与设定

首先在master1节点下载 Etdc，并解压缩放到 /opt 底下与安装：

service mesh

(https://www.kubernetes.org.cn/tags/service-mesh)

Spring Cloud (https://www.kubemetes.org.cn/tags/spring-cloud)

Traefik (https://www.kubernetes.org.cn/tags/traefik)

云原生 (https://www.kubernetes.org.cn/tags/%e4%ba%91%e5%8e%9f%e7%94)

企业案例

(https://www.kubernetes.org.cn/t

存储

(https://www.kubernetes.org.cn/tags/%e5

安全

(https://www.kubernetes.org.cn/tags/%e5%ae%'

容器

(https://www.kubernetes.org.cn/tags/%e5%ae%b9%e5%99%'

容器云 (https://www.kubernetes.org.cn/tags/%e5%ae%b9%e5%99%a8%e4%b'

微服务

(https://www.kubernetes.org.cn/tag

微软Azure (https://www.kubernetes.org.cn/tags/azure)

日志 (https://www.kubernetes.org.cn/tags/%e6%97%a5%e5%bf%97)

灵雀云

(https://www.kubernetes.org.c

监控

(https://www.kubernetes.org.cn/tags/%e7%9f

网络

(https://www.kubernetes.org.cn/tags

最新评论

零度幻冰 4天前说：
将外部服务(公网)映射到内部，是否要求K8S集群中的所有节点都能访问公网？不设置任何亲和度的情况下
(https://www.kubernetes.org.cn/4317.html#comment-1306)

哎的宣言 5天前说：
部署落地+业务迁移 玩转k8s进阶与企业级实践技能 网盘地址：
https://pan.baidu.com/s/1uIEhyQTyKdEZoF3PbSE13g&shfl=shareset
提取 (https://www.kubernetes.org.cn/doc-11#comment-1305)

kubeadm 5天前说：
证书轮换功能自动去apiserver更新证书，自动更新证书是否会
影响上面的业务呢？
(https://www.kubernetes.org.cn/5777.html#comment-1304)

泥鳅 6天前说：
我也遇到这个问题，，完全按照版主的文档操作的，，删除
证书，重新生成还是同样的错误，，
证书 重新生成还是同样的错误，，

```
$ export ETCD_URL="https://github.com/coreos/etcd/releases/download"
$ cd && wget -qO- --show-progress "${ETCD_URL}/v3.2.9/etcd-v3.2.9-linux-amd64.tar.gz" | tar
-zx
$ mv etcd-v3.2.9-linux-amd64/etcd* /usr/local/bin/ && rm -rf etcd-v3.2.9-linux-amd64
```

完成后新建 Etcd Group 与 User，并建立 Etcd 配置文件目录：

```
$ groupadd etcd && useradd -c "Etcd user" -g etcd -s /sbin/nologin -r etcd
```

下载etcd相关文件，我们将来管理 Etcd：

```
$ export ETCD_CONF_URL="https://kaiaren.github.io/files/manual-v1.8/master"
$ wget "${ETCD_CONF_URL}/etcd.conf" -O /etc/etcd/etcd.conf
$ wget "${ETCD_CONF_URL}/etcd.service" -O /lib/systemd/system/etcd.service
```

若与该教程 IP 不同的话，请用自己 IP 取代172.16.35.12。

建立 var 存放信息，然后启动 Etcd 服务：

```
$ mkdir -p /var/lib/etcd && chown etcd:etcd -R /var/lib/etcd /etc/etcd
$ systemctl enable etcd.service && systemctl start etcd.service
```

通过简单指令验证：

```
$ export CA="/etc/etcd/ssl"
$ ETCDCTL_API=3 etcdctl \
  --cacert=${CA}/etcd-ca.pem \
  --cert=${CA}/etcd.pem \
  --key=${CA}/etcd-key.pem \
  --endpoints="https://172.16.35.12:2379" \
  endpoint health
# output
https://172.16.35.12:2379 is healthy: successfully committed proposal: took = 641.36µs
```

Kubernetes Master

Master (<http://docs.kubernetes.org.cn/306.html>) 是 Kubernetes 的大总管，主要创建apiserver、Controller manager与Scheduler来组件管理所有 Node。本步骤将下载 Kubernetes 并安装至 master1上，然后产生相关 TLS Cert 与 CA 密钥，提供给集群组件认证使用。

下载 Kubernetes 组件

首先通过网络取得所有需要的执行文件：

```
# Download Kubernetes
$ export KUBE_URL="https://storage.googleapis.com/kubernetes-release/release/v1.8.2/bin/linu
x/amd64"
$ wget "${KUBE_URL}/kubelet" -O /usr/local/bin/kubelet
$ wget "${KUBE_URL}/kubect1" -O /usr/local/bin/kubect1
$ chmod +x /usr/local/bin/kubelet /usr/local/bin/kubect1

# Download CNI
$ mkdir -p /opt/cni/bin && cd /opt/cni/bin
$ export CNI_URL="https://github.com/containernetworking/plugins/releases/download"
$ wget -qO- --show-progress "${CNI_URL}/v0.6.0/cni-plugins-amd64-v0.6.0.tgz" | tar -zx
```

创建集群 CA 与 Certificates

在这部分，将会需要生成 client 与 server 的各组件 certificates，并且替 Kubernetes admin user 生成 client 证书。

创建pki文件夹，然后进入目录完成以下操作。

```
$ mkdir -p /etc/kubernetes/pki && cd /etc/kubernetes/pki
$ export PKI_URL="https://kairen.github.io/files/manual-v1.8/pki"
$ export KUBE_APISERVER="https://172.16.35.12:6443"
```

下载ca-config.json与ca-csr.json文件，并生成 CA 密钥：

```
$ wget "${PKI_URL}/ca-config.json" "${PKI_URL}/ca-csr.json"
$ cfssl gencert -initca ca-csr.json | cfssljson -bare ca
$ ls ca*.pem
ca-key.pem  ca.pem
```

API server certificate

下载apiserver-csr.json文件，并生成 kube-apiserver certificate 证书：

```
$ wget "${PKI_URL}/apiserver-csr.json"
$ cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=10.96.0.1,172.16.35.12,127.0.0.1,kubernetes.default \
  -profile=kubernetes \
  apiserver-csr.json | cfssljson -bare apiserver

$ ls apiserver*.pem
apiserver-key.pem  apiserver.pem
```

若节点 IP 不同，需要修改apiserver-csr.json的hosts。

Front proxy certificate

下载front-proxy-ca-csr.json文件，并生成 Front proxy CA 密钥，Front proxy 主要是用在 API aggregator 上：

```
$ wget "${PKI_URL}/front-proxy-ca-csr.json"
$ cfssl gencert \
  -initca front-proxy-ca-csr.json | cfssljson -bare front-proxy-ca

$ ls front-proxy-ca*.pem
front-proxy-ca-key.pem  front-proxy-ca.pem
```

下载front-proxy-client-csr.json文件，并生成 front-proxy-client 证书：

```
$ wget "${PKI_URL}/front-proxy-client-csr.json"
$ cfssl gencert \
  -ca=front-proxy-ca.pem \
  -ca-key=front-proxy-ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  front-proxy-client-csr.json | cfssljson -bare front-proxy-client

$ ls front-proxy-client*.pem
front-proxy-client-key.pem  front-proxy-client.pem
```

Bootstrap Token (<http://docs.kubernetes.org.cn/713.html>)

由于通过手动创建 CA 方式太过繁杂，只适合少量机器，因为每次签证时都需要绑定 Node IP，随机器增加会带来很多困扰，因此这边使用 TLS Bootstrapping 方式进行授权，由 apiserver 自动给符合条件的 Node 发送证书来授权加入集群。

主要做法是 kubelet 启动时，向 kube-apiserver 传送 TLS Bootstrapping 请求，而 kube-apiserver 验证 kubelet 请求的 token 是否与设定的一样，若一样就自动产生 kubelet 证书与密钥。具体作法可以参考 TLS bootstrapping (<http://docs.kubernetes.org.cn/698.html>)。

首先建立一个变量来产生BOOTSTRAP_TOKEN，并建立 bootstrap.conf 的 kubeconfig 文件：

```
$ export BOOTSTRAP_TOKEN=$(head -c 16 /dev/urandom | od -An -t x | tr -d ' ')
$ cat <<EOF > /etc/kubernetes/token.csv
${BOOTSTRAP_TOKEN},kubelet-bootstrap,10001,"system:kubelet-bootstrap"
EOF

# bootstrap set-cluster
$ kubectl config set-cluster kubernetes \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=../bootstrap.conf

# bootstrap set-credentials
$ kubectl config set-credentials kubelet-bootstrap \
  --token=${BOOTSTRAP_TOKEN} \
  --kubeconfig=../bootstrap.conf

# bootstrap set-context
$ kubectl config set-context default \
  --cluster=kubernetes \
  --user=kubelet-bootstrap \
  --kubeconfig=../bootstrap.conf

# bootstrap set default context
$ kubectl config use-context default --kubeconfig=../bootstrap.conf
```

若想要用 CA 方式来认证，可以参考 Kubelet certificate (<https://gist.github.com/kairen/60ad8545b79e8e7aa9bdc8a2893df7a0>)。

Admin certificate

下载admin-csr.json文件，并生成 admin certificate 证书：

```
$ wget "${PKI_URL}/admin-csr.json"
$ cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  admin-csr.json | cfssljson -bare admin

$ ls admin*.pem
admin-key.pem  admin.pem
```

接着通过以下指令生成名称为 admin.conf 的 kubeconfig 文件：

```
# admin set-cluster
$ kubectl config set-cluster kubernetes \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=./admin.conf

# admin set-credentials
$ kubectl config set-credentials kubernetes-admin \
  --client-certificate=admin.pem \
  --client-key=admin-key.pem \
  --embed-certs=true \
  --kubeconfig=./admin.conf

# admin set-context
$ kubectl config set-context kubernetes-admin@kubernetes \
  --cluster=kubernetes \
  --user=kubernetes-admin \
  --kubeconfig=./admin.conf

# admin set default context
$ kubectl config use-context kubernetes-admin@kubernetes \
  --kubeconfig=./admin.conf
```

Controller manager certificate

下载manager-csr.json文件，并生成 kube-controller-manager certificate 证书：

```
$ wget "${PKI_URL}/manager-csr.json"
$ cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  manager-csr.json | cfssljson -bare controller-manager

$ ls controller-manager*.pem
```

若节点 IP 不同，需要修改manager-csr.json的hosts。

接着通过以下指令生成名称为controller-manager.conf的 kubeconfig 文件：

```
# controller-manager set-cluster
$ kubectl config set-cluster kubernetes \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=./controller-manager.conf

# controller-manager set-credentials
$ kubectl config set-credentials system:kube-controller-manager \
  --client-certificate=controller-manager.pem \
  --client-key=controller-manager-key.pem \
  --embed-certs=true \
  --kubeconfig=./controller-manager.conf

# controller-manager set-context
$ kubectl config set-context system:kube-controller-manager@kubernetes \
  --cluster=kubernetes \
  --user=system:kube-controller-manager \
  --kubeconfig=./controller-manager.conf

# controller-manager set default context
$ kubectl config use-context system:kube-controller-manager@kubernetes \
  --kubeconfig=./controller-manager.conf
```

Scheduler certificate

下载scheduler-csr.json文件，并生成 kube-scheduler certificate 证书：

```
$ wget "${PKI_URL}/scheduler-csr.json"
$ cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  scheduler-csr.json | cfssljson -bare scheduler

$ ls scheduler*.pem
scheduler-key.pem  scheduler.pem
```

若节点 IP 不同，需要修改scheduler-csr.json的hosts。

接着通过以下指令生成名称为 scheduler.conf 的 kubeconfig 文件：

```
# scheduler set-cluster
$ kubectl config set-cluster kubernetes \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=./scheduler.conf

# scheduler set-credentials
$ kubectl config set-credentials system:kube-scheduler \
  --client-certificate=scheduler.pem \
  --client-key=scheduler-key.pem \
  --embed-certs=true \
  --kubeconfig=./scheduler.conf

# scheduler set-context
$ kubectl config set-context system:kube-scheduler@kubernetes \
  --cluster=kubernetes \
  --user=system:kube-scheduler \
  --kubeconfig=./scheduler.conf

# scheduler set default context
$ kubectl config use-context system:kube-scheduler@kubernetes \
  --kubeconfig=./scheduler.conf
```

Kubelet master certificate

下载kubelet-csr.json文件，并生成 master node certificate 证书：

```
$ wget "${PKI_URL}/kubelet-csr.json"
$ sed -i 's/${NODE}/master1/g' kubelet-csr.json
$ cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=master1,172.16.35.12,172.16.35.12 \
  -profile=kubernetes \
  kubelet-csr.json | cfssljson -bare kubelet

$ ls kubelet*.pem
kubelet-key.pem  kubelet.pem
```

这边\$NODE需要随节点名称不同而改变。

接着通过以下指令生成名称为 kubelet.conf 的 kubeconfig 文件：


```
# kubelet set-cluster
$ kubect1 config set-cluster kubernetes \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=./kubelet.conf

# kubelet set-credentials
$ kubect1 config set-credentials system:node:master1 \
  --client-certificate=kubelet.pem \
  --client-key=kubelet-key.pem \
  --embed-certs=true \
  --kubeconfig=./kubelet.conf

# kubelet set-context
$ kubect1 config set-context system:node:master1@kubernetes \
  --cluster=kubernetes \
  --user=system:node:master1 \
  --kubeconfig=./kubelet.conf

# kubelet set default context
$ kubect1 config use-context system:node:master1@kubernetes \
  --kubeconfig=./kubelet.conf
```

Service account key

Service account 不是通过 CA 进行认证，因此不要通过 CA 来做 Service account key 的检查，这边建立一组 Private 与 Public 密钥提供给 Service account key 使用：

```
$ openssl genrsa -out sa.key 2048
$ openssl rsa -in sa.key -pubout -out sa.pub
$ ls sa.*
sa.key  sa.pub
```

完成后删除不必要文件：

```
$ rm -rf *.json *.csr
```

确认/etc/kubernetes与/etc/kubernetes/pki有以下文件：

```
$ ls /etc/kubernetes/
admin.conf  bootstrap.conf  controller-manager.conf  kubelet.conf  pki  scheduler.conf  token.csv

$ ls /etc/kubernetes/pki
admin-key.pem  apiserver-key.pem  ca-key.pem  controller-manager-key.pem  front-proxy-ca-key
.pem  front-proxy-client-key.pem  kubelet-key.pem  sa.key  scheduler-key.pem
admin.pem  apiserver.pem  ca.pem  controller-manager.pem  front-proxy-ca.pem
front-proxy-client.pem  kubelet.pem  sa.pub  scheduler.pem
```

安装 Kubernetes 核心组件

首先下载 Kubernetes 核心组件 YAML 文件，这边我们不透过 Binary 方案来创建 Master 核心组件，而是利用 Kubernetes Static Pod 来创建，因此需下载所有核心组件的Static Pod文件到/etc/kubernetes/manifests目录：

```
$ export CORE_URL="https://kaiaren.github.io/files/manual-v1.8/master"
$ mkdir -p /etc/kubernetes/manifests && cd /etc/kubernetes/manifests
$ for FILE in apiserver manager scheduler; do
  wget "${CORE_URL}/${FILE}.yaml.conf" -O ${FILE}.yaml
done
```

若IP与教程设定不同的话，请记得修改apiserver.yml、manager.yml、scheduler.yml。
apiserver 中的 NodeRestriction 请参考 Using Node Authorization (<http://docs.kubernetes.org.cn/156.html>)。

生成一个用来加密 Etcd 的 Key:

```
$ head -c 32 /dev/urandom | base64
SUpbL4juUYyvxj3/gonV5xVEx8j769/99TSAf8YT/sQ=
```

在/etc/kubernetes/目录下，创建encryption.yml的加密 YAML 文件:

```
$ cat <<EOF > /etc/kubernetes/encryption.yml
kind: EncryptionConfig
apiVersion: v1
resources:
- resources:
  - secrets
providers:
- aescbc:
  keys:
  - name: key1
    secret: SUpbL4juUYyvxj3/gonV5xVEx8j769/99TSAf8YT/sQ=
- identity: {}
EOF
```

Etcd 数据加密可参考这篇 [Encrypting data at rest \(https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/\)](https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/)。

在/etc/kubernetes/目录下，创建audit-policy.yml的进阶审核策略 YAML 文件:

```
$ cat <<EOF > /etc/kubernetes/audit-policy.yml
apiVersion: audit.k8s.io/v1beta1
kind: Policy
rules:
- level: Metadata
EOF
```

Audit Policy 请参考这篇 [Auditing \(https://kubernetes.io/docs/tasks/debug-application-cluster/audit/\)](https://kubernetes.io/docs/tasks/debug-application-cluster/audit/)。

下载kubelet.service相关文件来管理 kubelet:

```
$ export KUBELET_URL="https://kaiaren.github.io/files/manual-v1.8/master"
$ mkdir -p /etc/systemd/system/kubelet.service.d
$ wget "${KUBELET_URL}/kubelet.service" -O /lib/systemd/system/kubelet.service
$ wget "${KUBELET_URL}/10-kubelet.conf" -O /etc/systemd/system/kubelet.service.d/10-kubelet.conf
```

最后创建 var 存放信息，然后启动 kubelet 服务:

```
$ mkdir -p /var/lib/kubelet /var/log/kubernetes
$ systemctl enable kubelet.service && systemctl start kubelet.service
```

完成后会需要一段时间来下载镜像文件与启动组件，可以利用该指令来查看:

```
$ watch netstat -ntlp
tcp        0      0 127.0.0.1:10248      0.0.0.0:*            LISTEN      23012/kubele
t
tcp        0      0 127.0.0.1:10251      0.0.0.0:*            LISTEN      22305/kube-s
chedule
tcp        0      0 127.0.0.1:10252      0.0.0.0:*            LISTEN      22529/kube-c
ontroll
tcp6       0      0 :::6443              :::*                  LISTEN      22956/kube-a
piserive
```

若看到以上信息表示服务正常启动，若发生问题可以用docker cli来查看。

完成后，复制 admin kubeconfig 文件，并通过简单指令验证：

```
$ cp /etc/kubernetes/admin.conf ~/.kube/config
$ kubectl get cs
NAME                STATUS    MESSAGE             ERROR
etcd-0              Healthy   {"health": "true"}
scheduler           Healthy   ok
controller-manager  Healthy   ok

$ kubectl get node
NAME        STATUS    ROLES    AGE      VERSION
master1    NotReady  master   4m       v1.8.2

$ kubectl -n kube-system get po
NAME                                READY    STATUS    RESTARTS   AGE
kube-apiserver-master1             1/1      Running   0          4m
kube-controller-manager-master1    1/1      Running   0          4m
kube-scheduler-master1             1/1      Running   0          4m
```

确认服务能够执行 logs 等指令：

```
$ kubectl -n kube-system logs -f kube-scheduler-master1
Error from server (Forbidden): Forbidden (user=kube-apiserver, verb=get, resource=nodes, sub
resource=proxy) ( pods/log kube-apiserver-master1)
```

这边会发现出现 403 Forbidden 问题，这是因为 kube-apiserver user 并没有 nodes 的资源权限，属于正常。

由于上述权限问题，我们必需创建一个 apiserver-to-kubelet-rbac.yml 来定义权限，以供我们执行 logs、exec 等指令：

```
$ cd /etc/kubernetes/
$ export URL="https://kaiaren.github.io/files/manual-v1.8/master"
$ wget "${URL}/apiserver-to-kubelet-rbac.yml.conf" -O apiserver-to-kubelet-rbac.yml
$ kubectl apply -f apiserver-to-kubelet-rbac.yml

# 测试 logs
$ kubectl -n kube-system logs -f kube-scheduler-master1
...
I1031 03:22:42.527697      1 leaderelection.go:184] successfully acquired lease kube-system
/kube-scheduler
```

Kubernetes Node

Node 是主要执行容器实例的节点，可视为工作节点。在这步骤我们会下载 Kubernetes binary 文件，并创建 node 的 certificate 来提供给节点注册认证用。Kubernetes 使用 Node Authorizer 来提供 Authorization mode (<http://docs.kubernetes.org.cn/156.html>)，这种授权模式会替 Kubelet 生成 API request。

在开始前，我们先在master1将需要的 ca 与 cert 复制到 Node 节点上：

```
$ for NODE in node1 node2; do
  ssh ${NODE} "mkdir -p /etc/kubernetes/pki/"
  ssh ${NODE} "mkdir -p /etc/etcd/ssl"
  # Etcd ca and cert
  for FILE in etcd-ca.pem etcd.pem etcd-key.pem; do
    scp /etc/etcd/ssl/${FILE} ${NODE}:/etc/etcd/ssl/${FILE}
  done
  # Kubernetes ca and cert
  for FILE in pki/ca.pem pki/ca-key.pem bootstrap.conf; do
    scp /etc/kubernetes/${FILE} ${NODE}:/etc/kubernetes/${FILE}
  done
done
```

下载 Kubernetes 组件

首先通过网络取得所有需要的执行文件：

```
# Download Kubernetes
$ export KUBE_URL="https://storage.googleapis.com/kubernetes-release/release/v1.8.2/bin/linux/amd64"
$ wget "${KUBE_URL}/kubelet" -O /usr/local/bin/kubelet
$ chmod +x /usr/local/bin/kubelet

# Download CNI
$ mkdir -p /opt/cni/bin && cd /opt/cni/bin
$ export CNI_URL="https://github.com/containernetworking/plugins/releases/download"
$ wget -qO- --show-progress "${CNI_URL}/v0.6.0/cni-plugins-amd64-v0.6.0.tgz" | tar -zx
```

设定 Kubernetes node

接着下载 Kubernetes 相关文件，包含 drop-in file、systemd service 档案等：

```
$ export KUBELET_URL="https://kaiaren.github.io/files/manual-v1.8/node"
$ mkdir -p /etc/systemd/system/kubelet.service.d
$ wget "${KUBELET_URL}/kubelet.service" -O /lib/systemd/system/kubelet.service
$ wget "${KUBELET_URL}/10-kubelet.conf" -O /etc/systemd/system/kubelet.service.d/10-kubelet.conf
```

接着在所有node创建 var 存放信息，然后启动 kubelet 服务：

```
$ mkdir -p /var/lib/kubelet /var/log/kubernetes /etc/kubernetes/manifests
$ systemctl enable kubelet.service && systemctl start kubelet.service
```

P.S. 重复一样动作来完成其他节点。

授权 Kubernetes Node

当所有节点都完成后，在master节点，因为我们采用 TLS Bootstrapping，所需要创建一个 ClusterRoleBinding：

```
$ kubectl create clusterrolebinding kubelet-bootstrap \
  --clusterrole=system:node-bootstrapper \
  --user=kubelet-bootstrap
```

在master通过简单指令验证，会看到节点处于pending：

```
$ kubectl get csr
NAME                                     AGE      REQUESTOR           CONDITI
ON
node-csr-YWf97ZrLCTlr2hmXsNLfjVLwLfZRsu52FRKOYjpcBE  2s       kubelet-bootstrap   Pending
node-csr-eq4q6ffOwT4yqYQNU6sT7mphPOQdFN6yulMVZeu6pkE  2s       kubelet-bootstrap   Pending
```

通过 kubectl 来允许节点加入集群：

```
$ kubectl get csr | awk '/Pending/ {print $1}' | xargs kubectl certificate approve
certificatesigningrequest "node-csr-YWf97ZrLCTlr2hmXsNLfjVLwLfZRsu52FRKOYjpcBE" approved
certificatesigningrequest "node-csr-eq4q6ffOwT4yqYQNU6sT7mphPOQdFN6yulMVZeu6pkE" approved

$ kubectl get csr
NAME                                     AGE      REQUESTOR           CONDITI
ON
node-csr-YWf97ZrLCTlr2hmXsNLfjVLwLfZRsu52FRKOYjpcBE  30s      kubelet-bootstrap   Approve
d, Issued
node-csr-eq4q6ffOwT4yqYQNU6sT7mphPOQdFN6yulMVZeu6pkE  30s      kubelet-bootstrap   Approve
d, Issued

$ kubectl get no
NAME      STATUS    ROLES    AGE      VERSION
master1   NotReady  master   15m      v1.8.2
node1     NotReady  <none>   8m       v1.8.2
node2     NotReady  <none>   6s       v1.8.2
```

Kubernetes Core Addons 部署

当完成上面所有步骤后，接着我们需要安装一些插件，而这些有部分是非常重要跟好用的，如Kube-dns与Kube-proxy等。

Kube-proxy addon

Kube-proxy (<https://github.com/kubernetes/kubernetes/tree/master/cluster/addons/kube-proxy>) 是实现 Service 的关键组件，kube-proxy 会在每台节点上执行，然后监听 API Server 的 Service 与 Endpoint 资源对象的改变，然后来依据变化执行 iptables 来实现网络的转发。这边我们会需要建议一个 DaemonSet 来执行，并且创建一些需要的 certificate。Kubernetes 1.8 kube-proxy 开启 ipvs (<https://www.kubernetes.org.cn/3025.html>)

首先在master1下载kube-proxy-csr.json文件，并产生 kube-proxy certificate 证书：

```
$ export PKI_URL="https://kaiaren.github.io/files/manual-v1.8/pki"
$ cd /etc/kubernetes/pki
$ wget "${PKI_URL}/kube-proxy-csr.json" "${PKI_URL}/ca-config.json"
$ cfssl gencert \
    -ca=ca.pem \
    -ca-key=ca-key.pem \
    -config=ca-config.json \
    -profile=kubernetes \
    kube-proxy-csr.json | cfssljson -bare kube-proxy

$ ls kube-proxy*.pem
kube-proxy-key.pem  kube-proxy.pem
```

接着透过以下指令生成名称为 kube-proxy.conf 的 kubeconfig 文件：

Kubernetes 1.16 版本

今年9月18日，Kubernetes 迎来了2019年的第三个新版本1.16。根据 Release Note 介绍，Kubernetes v1.16 由31个增强功能组成：8个进入稳定，8个进入 Beta，15个进入 Alpha。
(<https://www.kubernetes.org.cn/tags/kubernetes-1-16>)



关注「K8S中文社区」微信公众号
回复“文档”
获取K8S文档下载链接
回复“加群”
加入K8S微信技术交流群

最新文章

- 

Portworx: 多云成为使用容器技术的主要驱动力
2019-11-02 评论()
(<https://www.kubernetes.org.cn/6000>).
- 

istio-断路器示例
2019-10-31 评论()
(<https://www.kubernetes.org.cn/5999>).
- 

下载达 10 万次的 IDEA 插件，K8s 一键部署了解一下？
2019-10-30 评论()
(<https://www.kubernetes.org.cn/5998>).
- 

Dubbo 在 K8s 下的思考
2019-10-29 评论()
(<https://www.kubernetes.org.cn/5996>).

Kubernetes 版本资讯

- Kubernetes v1.16 正式版已发布
(<https://www.kubernetes.org.cn/5838.html>)
- Kubernetes v1.15正式版已发布
(<https://www.kubernetes.org.cn/tags/kubernetes-1-15>)
- Kubernetes v1.14 正式版已发布
(<https://www.kubernetes.org.cn/5204.html>)
- Kubernetes v1.13 正式版已发布
(<https://www.kubernetes.org.cn/4896.html>)
- Kubernetes v1.13 beta.1 发布
(<https://github.com/kubernetes/kubernetes/releases/tag/v1.13.0-beta.1>)

```
# kube-proxy set-cluster
$ kubectl config set-cluster kubernetes \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server="https://172.16.35.12:6443" \
  --kubeconfig=./kube-proxy.conf

# kube-proxy set-credentials
$ kubectl config set-credentials system:kube-proxy \
  --client-key=kube-proxy-key.pem \
  --client-certificate=kube-proxy.pem \
  --embed-certs=true \
  --kubeconfig=./kube-proxy.conf

# kube-proxy set-context
$ kubectl config set-context system:kube-proxy@kubernetes \
  --cluster=kubernetes \
  --user=system:kube-proxy \
  --kubeconfig=./kube-proxy.conf

# kube-proxy set default context
$ kubectl config use-context system:kube-proxy@kubernetes \
  --kubeconfig=./kube-proxy.conf
```

完成后删除不必要文件:

```
$ rm -rf *.json
```

确认/etc/kubernetes有以下文件:

```
$ ls /etc/kubernetes/
admin.conf          bootstrap.conf      encryption.yml     kube-proxy.conf    pki
token.csv
audit-policy.yml    controller-manager.conf  kubelet.conf      manifests          scheduler.conf
```

在master1将kube-proxy相关文件复制到 Node 节点上:

```
$ for NODE in node1 node2; do
  for FILE in pki/kube-proxy.pem pki/kube-proxy-key.pem kube-proxy.conf; do
    scp /etc/kubernetes/${FILE} ${NODE}:/etc/kubernetes/${FILE}
  done
done
```

完成后, 在master1通过 kubectl 来创建 kube-proxy daemon:

```
$ export ADDON_URL="https://kaiaren.github.io/files/manual-v1.8/addon"
$ mkdir -p /etc/kubernetes/addons && cd /etc/kubernetes/addons
$ wget "${ADDON_URL}/kube-proxy.yml.conf" -O kube-proxy.yml
$ kubectl apply -f kube-proxy.yml
$ kubectl -n kube-system get po -l k8s-app=kube-proxy
NAME                READY   STATUS    RESTARTS   AGE
kube-proxy-bpp7q    1/1     Running   0           47s
kube-proxy-cztvh    1/1     Running   0           47s
kube-proxy-q7mm4    1/1     Running   0           47s
```

Kube-dns addon

Kube DNS (<http://docs.kubernetes.org.cn/733.html>) 是 Kubernetes 集群内部 Pod 之间互相沟通的重要 Addon, 它允许 Pod 可以通过 Domain Name 方式来连接 Service, 其主要由 Kube DNS 与 Sky DNS 组合而成, 通过 Kube DNS 监听 Service 与 Endpoint 变化, 来提供给 Sky DNS 信息, 已更新解析地址。

安装只需要在master1通过 kubectl 来创建 kube-dns deployment 即可:

```
$ export ADDON_URL="https://kairen.github.io/files/manual-v1.8/addon"
$ wget "${ADDON_URL}/kube-dns.yml.conf" -O kube-dns.yml
$ kubectl apply -f kube-dns.yml
$ kubectl -n kube-system get po -l k8s-app=kube-dns
```

NAME	READY	STATUS	RESTARTS	AGE
kube-dns-6cb549f55f-h4zr5	0/3	Pending	0	40s

Calico Network 安装与设定

Calico 是一款纯 Layer 3 的数据中心网络方案(不需要 Overlay 网络)，Calico 好处是他已与各种云原生平台有良好的整合，而 Calico 在每一个节点利用 Linux Kernel 实现高效的 vRouter 来负责数据的转发，而当数据中心复杂度增加时，可以用 BGP route reflector 来达成。

首先在master1通过 kubectl 建立 Calico policy controller:

```
$ export CALICO_CONF_URL="https://kairen.github.io/files/manual-v1.8/network"
$ wget "${CALICO_CONF_URL}/calico-controller.yml.conf" -O calico-controller.yml
$ kubectl apply -f calico-controller.yml
$ kubectl -n kube-system get po -l k8s-app=calico-policy
```

NAME	READY	STATUS	RESTARTS	AGE
calico-policy-controller-5ff8b4549d-tctmm	0/1	Pending	0	5s

在master1下载 Calico CLI 工具:

```
$ wget https://github.com/projectcalico/calicoctl/releases/download/v1.6.1/calicoctl
$ chmod +x calicoctl && mv calicoctl /usr/local/bin/
```

然后在所有节点下载 Calico，并执行以下步骤:

```
$ export CALICO_URL="https://github.com/projectcalico/cni-plugin/releases/download/v1.11.0"
$ wget -N -P /opt/cni/bin ${CALICO_URL}/calico
$ wget -N -P /opt/cni/bin ${CALICO_URL}/calico-ipam
$ chmod +x /opt/cni/bin/calico /opt/cni/bin/calico-ipam
```

接着在所有节点下载 CNI plugins配置文件，以及 calico-node.service:

```
$ mkdir -p /etc/cni/net.d
$ export CALICO_CONF_URL="https://kairen.github.io/files/manual-v1.8/network"
$ wget "${CALICO_CONF_URL}/10-calico.conf" -O /etc/cni/net.d/10-calico.conf
$ wget "${CALICO_CONF_URL}/calico-node.service" -O /lib/systemd/system/calico-node.service
```

若部署的机器是使用虚拟机，如 Virtualbox 等的话，请修改calico-node.service文件，并在 IP_AUTODETECTION_METHOD(包含 IP6)部分指定绑定的网卡，以避免默认绑定到 NAT 网络上。

之后在所有节点启动 Calico-node:

```
$ systemctl enable calico-node.service && systemctl start calico-node.service
```

在master1查看 Calico nodes:

```
$ cat <<EOF > ~/calico-rc
export ETCD_ENDPOINTS="https://172.16.35.12:2379"
export ETCD_CA_CERT_FILE="/etc/etcd/ssl/etcd-ca.pem"
export ETCD_CERT_FILE="/etc/etcd/ssl/etcd.pem"
export ETCD_KEY_FILE="/etc/etcd/ssl/etcd-key.pem"
EOF

$ . ~/calico-rc
$ calicoctl get node -o wide
NAME          ASN          IPV4          IPV6
master1      (64512)      172.16.35.12/24
node1        (64512)      172.16.35.10/24
node2        (64512)      172.16.35.11/24
```

查看 pending 的 pod 是否已执行：

```
$ kubectl -n kube-system get po
NAME                                     READY    STATUS    RESTARTS   AGE
calico-policy-controller-5ff8b4549d-tctmm 1/1      Running   0           4m
kube-apiserver-master1                   1/1      Running   0           20m
kube-controller-manager-master1          1/1      Running   0           20m
kube-dns-6cb549f55f-h4zr5                3/3      Running   0           5m
kube-proxy-fnrkb                          1/1      Running   0           6m
kube-proxy-l72bq                          1/1      Running   0           6m
kube-proxy-m6rfw                          1/1      Running   0           6m
kube-scheduler-master1                   1/1      Running   0           20m
```

最后若想省事，可以直接用 Standard Hosted (<https://docs.projectcalico.org/v2.6/getting-started/kubernetes/installation/hosted/hosted>) 方式安装。

Kubernetes Extra Addons 部署

本节说明如何部署一些官方常用的 Addons，如 Dashboard、Heapster 等。

Dashboard addon

Dashboard 是 Kubernetes 社区官方开发的仪表板，有了仪表板后管理者就能够透过 Web-based 方式来管理 Kubernetes 集群，除了提升管理方便，也让资源可视化，让人更直觉看见系统信息的呈现结果。

首先我们要建立kubernetes-dashboard-certs，来提供给 Dashboard TLS 使用：

```
$ mkdir -p /etc/kubernetes/addons/certs && cd /etc/kubernetes/addons
$ openssl genrsa -des3 -passout pass:x -out certs/dashboard.pass.key 2048
$ openssl rsa -passin pass:x -in certs/dashboard.pass.key -out certs/dashboard.key
$ openssl req -new -key certs/dashboard.key -out certs/dashboard.csr -subj '/CN=kube-dashboa
rd'
$ openssl x509 -req -sha256 -days 365 -in certs/dashboard.csr -signkey certs/dashboard.key -
out certs/dashboard.crt
$ rm certs/dashboard.pass.key
$ kubectl create secret generic kubernetes-dashboard-certs\
  --from-file=certs -n kube-system
```

接着在master1通过 kubectl 来建立 kubernetes dashboard 即可：

```
$ export ADDON_URL="https://kaiaren.github.io/files/manual-v1.8/addon"
$ wget ${ADDON_URL}/kube-dashboard.yml.conf -O kube-dashboard.yml
$ kubectl apply -f kube-dashboard.yml
$ kubectl -n kube-system get po,svc -l k8s-app=kubernetes-dashboard
NAME                                     READY    STATUS    RESTARTS   AGE
po/kubernetes-dashboard-747c4f7cf-md5m8 1/1      Running   0           56s

NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
svc/kubernetes-dashboard ClusterIP    10.98.120.209 <none>         443/TCP     56s
```


P.S. 这边会额外创建一个名称为anonymous-open-door Cluster Role Binding。这仅作为方便测试时使用，在一般情况下不要开启，不然就会直接被存取所有 API。

完成后，就可以透过浏览器访问 Dashboard，<https://172.16.35.12:6443/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>

Heapster addon

Heapster (<https://www.kubernetes.org.cn/932.html>) 是 Kubernetes 社区维护的容器集群监控分析工具。

Heapster 会从 Kubernetes apiserver 获得所有 Node 信息，然后再通过这些 Node 来获得 kubelet 上的数据，最后再将所有收集到数据送到 Heapster 的后台储存 InfluxDB。最后利用 Grafana 来抓取 InfluxDB 的数据源来进行可视化。

在master1通过 kubectl 来创建 kubernetes monitor 即可：

```
$ export ADDON_URL="https://kairen.github.io/files/manual-v1.8/addon"
$ wget ${ADDON_URL}/kube-monitor.yml.conf -O kube-monitor.yml
$ kubectl apply -f kube-monitor.yml
$ kubectl -n kube-system get po,svc
```

NAME	READY	STATUS	RESTARTS	AGE
...				
po/heapster-74fb5c8cdc-62xzc	4/4	Running	0	7m
po/influxdb-grafana-55bd7df44-nw4nc	2/2	Running	0	7m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
...					
svc/heapster	ClusterIP	10.100.242.225	<none>	80/TCP	7m
svc/monitoring-grafana	ClusterIP	10.101.106.180	<none>	80/TCP	7m
svc/monitoring-influxdb	ClusterIP	10.109.245.142	<none>	8083/TCP,8086/TCP	7m
...					

完成后，就可以透过浏览器存取 Grafana Dashboard，<https://172.16.35.12:6443/api/v1/proxy/namespaces/kube-system/services/monitoring-grafana>

简单部署 Nginx 服务

Kubernetes 可以选择使用指令直接创建应用程序与服务，或者撰写 YAML 与 JSON 档案来描述部署应用程序的配置，以下将创建一个简单的 Nginx 服务：

```
$ kubectl run nginx --image=nginx --port=80
$ kubectl expose deploy nginx --port=80 --type=LoadBalancer --external-ip=172.16.35.12
$ kubectl get svc,po
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	1h
svc/nginx	LoadBalancer	10.97.121.243	172.16.35.12	80:30344/TCP	22s

NAME	READY	STATUS	RESTARTS	AGE
po/nginx-7cbc4b4d9c-77961	1/1	Running	0	28s

这边type可以选择 NodePort 与 LoadBalancer，在本地裸机部署，两者差异在于NodePort只映射 Host port 到 Container port，而LoadBalancer则继承NodePort额外多出映射 Host target port 到 Container port。

确认没问题后即可在浏览器存取 <http://172.16.35.12>

扩展服务数量

```
$ kubectl scale deploy nginx --replicas=2
```



```
$ kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
nginx-158599303-0h9lr	1/1	Running	0	25s	10.244.100.5	node2
nginx-158599303-k7cbt	1/1	Running	0	1m	10.244.24.3	node1



关注微信公众号，加入社区

[illegible]

上一篇: [kubernetes1.8.1安装指南](https://kubernetes.io/zh-cn/docs/concepts/overview/what-is-kubernetes/)
(<https://www.kubernetes.org.cn/3063.html>)

下一篇: [好用的K8S 1.6.4 集群搭建初学表结构](https://kubernetes.io/zh-cn/docs/concepts/overview/what-is-kubernetes/)
(<https://www.kubernetes.org.cn/3265.html>)

标签: `Kubernetes1.8` (<https://www.kubernetes.org.cn/tags/kubernetes1-8>)

相关推荐

- 好用的K8S1.8.4 集群二进制安装指南 (<https://www.kubernetes.org.cn/3265.html>)
- Kubernetes 1.8 kube-scheduler的源码分析 (<https://www.kubernetes.org.cn/3235.html>)
- kubernetes 1.8.1安装指南，离线安装，内网安装 (<https://www.kubernetes.org.cn/3063.html>)
- Kubernetes 1.8 kube-proxy 开启 ipvs (<https://www.kubernetes.org.cn/3025.html>)
- 使用kubeadm安装Kubernetes 1.8版本 (<https://www.kubernetes.org.cn/2906.html>)
- 越来越稳！Kubernetes 1.8.0 版本发布 (<https://www.kubernetes.org.cn/2819.html>)
- Kubernetes 1.8.0 RC.1 版本发布 (<https://www.kubernetes.org.cn/2789.html>)
- Kubernetes 1.8.0-beta.1 版本发布 (<https://www.kubernetes.org.cn/2674.html>)

评论 46

社区交流

1

8

3

5

2

提交评论

昵称

邮箱

网址

昵称 (必填)









邮箱 (必填)

网址

你好，我这里连接不上。是需要翻墙吗 #36
_黑夜 2年前 (2017-11-06)

貌似网络不是很好 #35
其实很简单 2年前 (2017-11-07)

-  我按照以上步骤一步一步，一直到使用watch netstat -ntlp命令查看服务启动情况，但是10251、10252、6443端口的服务都没启动。。。我也是懵逼了
其实很简单 2年前 (2017-11-07) #34
-  你好,你这问题如何解决的,
不二心 2年前 (2017-11-08)
-  我也是。。。
欠你一个拥抱 2年前 (2017-11-08)
-  我的也是和你一样情况，没6443和10251,10252端口，咋解决的？
疯人愿 2年前 (2018-02-06)
-  你有搞到这三个镜像吗？
逝、我想念你 2年前 (2018-03-27)
-  久等一会就好了
舍不得舍 2年前 (2017-11-08) #33
-  这篇文档有问题，很多初始环境的设置没有写进去，作者还是修改一下吧。
离心咒 2年前 (2017-11-20)
-  启动kubelet的时候报错，说没有cni network以及no client provided，具体log：
Nov 09 21:01:13 master1 kubelet[5570]: W1109 21:01:13.977813 5570 cni.go:196] Unable to update cni config: No networks found in /etc/cni/net.d
Nov 09 21:01:13 master1 kubelet[5570]: I1109 21:01:13.983192 5570 feature_gate.go:156] feature gates: map[]
Nov 09 21:01:13 master1 kubelet[5570]: W1109 21:01:13.983297 5570 server.go:289] --cloud-provider=auto-detect is deprecated. The desired cloud provider should be set explicitly
Nov 09 21:01:13 master1 kubelet[5570]: W1109 21:01:13.985029 5570 server.go:381] invalid kubeconfig: invalid configuration: no server found for cluster kubernetes
Nov 09 21:01:13 master1 kubelet[5570]: error: failed to run Kubelet: no client provided, cannot use webhook authorization
Nov 09 21:01:13 master1 systemd[1]: kubelet.service: main process exited, code=exited, status=1/FAILURE #32
-  请问启动kubelet的时候报错，说no cni network 以及no client provided怎么办，你最后搭起来了吗？我搭了好几次了每次都是各种问题想死了
无双猪 2年前 (2017-11-09) #31
-  请问，我把三个文件下载到本地后，需要怎样操作，把google的镜像更改到本地 kube-apiserver-amd64.v1.8.2.tar kube-controller-manager-amd64.v1.8.2.tar kube-scheduler-amd64.v1.8.2.tar
asdsal 2年前 (2017-11-13) #30
-  请问：你这三个文件是从哪下载的？
_黑夜 2年前 (2017-11-15)
-  需要搭建本地仓库，然后把制作好的镜像push到本地仓库，更改Yaml文件，就可以了
asdsal 2年前 (2017-11-13) #29
-  我也是端口无法启动，这个需要翻墙吗
三宝玉如意 2年前 (2017-11-14) #28
-  没装成功。。。我只想知道不重装系统，怎么便捷的恢复到安装之前的状态[大哭]
似水柔情 2年前 (2017-11-17) #27
-  不建议大家按这篇文去做。一步一步按着做，都是提示6443没有响应。而且那个master的IP不同时，总是出错。
此文只能参考而以。做了两次，一点一点的查，都是报一样的错。
曲终人散成幻 2年前 (2017-11-18) #26

	目前我根据此文档安装k8s已经到了Kubernetes Extra Addons 部署，有兴趣一起交流填坑的朋友家下我的QQ群吧558468868 普希雪兰特 2年前 (2017-11-21)	#25
	看得云里雾里一脸懵逼，这种教程好毁人 好网名 2年前 (2017-11-23)	#24
	我按照以上步骤一步一步，一直到使用watch netstat -ntlp命令查看服务启动情况，但是10251、10252、6443端口的服务都没启动。。。我也是懵逼了 时光流离 2年前 (2017-12-01)	#23
	我也卡在这步骤，怎么破? 时光流离 2年前 (2017-12-01)	#22
	dd 小步调 2年前 (2017-12-07)	#21
	本文亲测可用 小步调 2年前 (2017-12-07)	#20
	我也成功了 梦回中 2年前 (2018-01-16)	
	哈喽，能否指导一下~~~ aTaylorswift 2年前 (2018-03-09)	
	我尝试到使用watch netstat -ntlp命令查看服务启动情况，但是10251、10252、6443端口的服务都没启动，麻烦作者把初始化的参数标一下下，那个位置需要改ip的！ 浮_沉_ 2年前 (2017-12-19)	#19
	计算节点起 kubelet报错,按教程走到这里 授权 Kubernetes Node Dec 23 05:15:04 slave1 systemd[1]: kubelet.service holdoff time over, scheduling restart. Dec 23 05:15:04 slave1 systemd[1]: Started kubelet: The Kubernetes Node Agent. Dec 23 05:15:04 slave1 systemd[1]: Starting kubelet: The Kubernetes Node Agent... Dec 23 05:15:04 slave1 kubelet[1861]: I1223 05:15:04.620173 1861 feature_gate.go:156] feature gates: map[] Dec 23 05:15:04 slave1 kubelet[1861]: I1223 05:15:04.620227 1861 controller.go:114] kubelet config controller: starting controller Dec 23 05:15:04 slave1 kubelet[1861]: I1223 05:15:04.620231 1861 controller.go:118] kubelet config controller: validating combination of defaults and flags Dec 23 05:15:04 slave1 kubelet[1861]: I1223 05:15:04.625415 1861 client.go:75] Connecting to docker on unix:///var/run/docker.sock Dec 23 05:15:04 slave1 kubelet[1861]: I1223 05:15:04.625483 1861 client.go:95] Start docker client	#18
	支持下。 Locke 2年前 (2017-12-26)	#17
	按照此教程，完成k8s集群部署，教程没毛病，有问题的是国内网络，下不到谷歌的镜像，你的容器起不来，端口怎么会有呢? 梦回中 2年前 (2018-01-16)	#16
	翻墙 网络会不会好些 qq174540409 2年前 (2018-03-16)	
	能详细说说么？我也碰到这个问题，DNS死活起不来，一直i/o timeout 原上草 2年前 (2018-02-11)	#15
	ffff 扯不断 2年前 (2018-02-22)	#14
	kubelet 启动加上 --hostname-override=master1	#13



岱 2年前 (2018-02-27)



能不能解释一下，master上按照操作均能kubelet启动成功了，配置node节点的时候，启动kubelet服务报错，kubelet: error: failed to run Kubelet: cannot create certificate signing request: Unauthorized是什么原因啊。为什么是：Unauthorized，还有我也没有看到教程里面有提到创建/etc/cni/cni.d/cni.conf 这个是网络插件么，master启动时要去找，但是没有，结果master仍然能够正常启动。有知道的朋友解答一下么？

滔滔 2年前 (2018-03-03)



我也遇到了同样的问题，请问你的解决了吗

aTaylorswift 2年前 (2018-03-14)



你好，我也遇到同样的问题，请问你搞定了么？

冒泡哥 2年前 (2018-03-08)



[root@node162 manifests]# kubectl get po //查看pods的时候出现如下错误：
The connection to the server 192.168.161.162:6443 was refused – did you specify the right host or port?
[root@node162 manifests]# tailf /var/log/messages //查看系统日志的时候不停的滚动如下报错信息，这是怎么回事？大牛来来来，帮小弟一把！
Mar 17 14:10:40 node162 kubelet: W0317 14:10:40.861680 5056 helpers.go:847] eviction manager: no observation found for eviction signal allocatableNodeFs.available
Mar 17 14:10:50 node162 kubelet: I0317 14:10:50.863147 5056 kubelet_node_status.go:280] Setting node annotation to enable volume controller attach/detach
Mar 17 14:10:50 node162 kubelet: W0317 14:10:50.898163 5056 helpers.go:847] eviction manager: no observation found for eviction signal allocatableNodeFs.available
Mar 17 14:11:00 node162 kubelet: I0317 14:11:00.901128 5056 kubelet_node_status.go:280] Setting node annotation to enable volume controller attach/detach
Mar 17 14:11:00 node162 kubelet: W0317 14:11:00.948701 5056 helpers.go:847] eviction manager: no



Controller manager certificate说要修改manager-csr.json里面的hosts文件，打开文件，根本就没有hosts这个关键字？

逝、我想冬謔 2年前 (2018-03-27)



谁有镜像文件啊？给发给我一下下，google上不了啊

逝、我想冬謔 2年前 (2018-03-27)



我的10251,10252,10248都启动了，但是6443怎么也启动不了。docker翻墙正常，但日志一直在报这个。dial tcp 192.168.16.170:6443: getsockopt: connection refused。请问要怎么处理，从etcd开始重新做过几次了

willis 2年前 (2018-03-30)



镜像都下载好了，docker也启动这些镜像了，但是为啥6443还是提示 connection refused

willis 2年前 (2018-03-30)



请问这个错误是神吗？

大漠雄鹰 2年前 (2018-04-04)



asdf

大漠雄鹰 2年前 (2018-04-04)



不靠谱不靠谱。

同君醉 1年前 (2018-06-05)



node节点无法启动kubelet:
8月 07 14:10:57 node1 kubelet[6425]: error: failed to run Kubelet: cannot create certificate signing request: Unauthorized
原因应该是Bootstrap Token 这一步不对
~~~  
\$ export BOOTSTRAP\_TOKEN=\$(head -c 16 /dev/urandom | od -An -t x | tr -d ' ')  
\$ cat < /etc/kubernetes/token.csv  
\${BOOTSTRAP\_TOKEN},kubelet-bootstrap,10001,"system:kubelet-bootstrap"

EOF

~~~

看一下cat /etc/kubernetes/token.csv是否正确，我的是这个问题\${BOOTSTRAP_TOKEN}不正确

神马王族 1 条评论 / 2018-09-26



同没有kube-apiserver,kube-scheduler,kube-controlle,kube-proxy服务

#1

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 127.0.0.1:10248 0.0.0.0:* LISTEN 17538/kubelet

tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 838/sshd

tcp6 0 0 :::4194 :::* LISTEN 17538/kubelet

tcp6 0 0 :::10250 :::* LISTEN 17538/kubelet

tcp6 0 0 :::2379 :::* LISTEN 16343/etcd

tcp6 0 0 :::2380 :::* LISTEN 16343/etcd

tcp6 0 0 :::10255 :::* LISTEN 17538/kubelet

tcp6 0 0 :::22 :::* LISTEN 838/sshd

kinamu 1年前 (2018-09-26)