Kubeadm 超详细安装 Kubernetes 1.13

原创： 宝哥 云技术 今天

前言

Kubernetes目前已经是非常火爆，跟当年巅峰时期的OpenStack有的一比。国内很多同学学习的时候，因为“科学上网”的问题，导致第一步安装就被卡住了。不用担心，本篇文章将告诉大家一个不用“科学上网”就能很顺利安装Kubernetes的方法，也不用费劲弄什么私有仓库，让你轻松搭建Kubernetes，上手有信心。

另外本篇文章采用最新的Kubernetes 1.13版本。

Kubernetes 版本查阅地址：
https://github.com/kubernetes/kubernetes/releases

一、实践环境准备

1. 服务器虚拟机准备

IP地址	节点角色	CPU	Memory	Hostname
192.168.1.11	master and etcd	>=2c	>=2G	master
192.168.1.12	worker	>=2c	>=2G	node

本实验我这里用的VM是vmware workstation创建的，每个给了4C 4G 100GB配置，大家根据自己的资源情况，按照上面给的建议最低值创建即可。

注意：hostname不能有大写字母，比如Master这样。

2. 软件版本

系统类型	Kubernetes版本	docker版本	kubeadm版本	kubectl版本	kubelet版本
CentOS7.5.1804	v1.13	18.06.1-ce	v1.13	v1.13	v1.13

注意：这里采用的软件版本，请大家严格与我保持一致！ 开源软件，版本非常敏感和重要！

3. 环境初始化操作

3.1 配置hostname

```
$ hostnamectl set-hostname master
$ hostnamectl set-hostname node
```

每台机器上设置对应好hostname，注意，不能有大写字母！

3.2 配置/etc/hosts

注意，hosts文件非常重要，请在每个节点上执行：

```
cat <<EOF > /etc/hosts 127.0.0.1
localhost localhost.localdomain
localhost4 localhost4.localdomain4 ::1
localhost localhost.localdomain
localhost6 localhost6.localdomain6 192.168.1.11
master 192.168.1.12
node EOF
```

3.3 关闭防火墙、selinux、swap

```
# 停防火墙
$ systemctl stop firewalld
$ systemctl disable firewalld

# 关闭Selinux
$ setenforce  0
$ sed -i "s/^SELINUX=enforcing/SELINUX=disabled/g" /etc/sysconfig/selinux
$ sed -i "s/^SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
$ sed -i "s/^SELINUX=permissive/SELINUX=disabled/g" /etc/sysconfig/selinux
$ sed -i "s/^SELINUX=permissive/SELINUX=disabled/g" /etc/selinux/config

# 关闭Swap
$ swapoff –a
$ sed -i 's/.*swap.*/#&/' /etc/fstab

# 加载br_netfilter
$ modprobe br_netfilter
```

3.4 配置内核参数

```
# 配置sysctl内核参数
$ cat > /etc/sysctl.d/k8s.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

# 生效文件
$ sysctl -p /etc/sysctl.d/k8s.conf

# 修改Linux 资源配置文件，调高ulimit最大打开数和systemctl管理的服务文件最大打开数
$ echo "* soft nofile 655360" >> /etc/security/limits.conf
$ echo "* hard nofile 655360" >> /etc/security/limits.conf
$ echo "* soft nproc 655360"  >> /etc/security/limits.conf
$ echo "* hard nproc 655360"  >> /etc/security/limits.conf
$ echo "* soft  memlock  unlimited"  >> /etc/security/limits.conf
```

$ echo "* hard memlock unlimited" >> /etc/security/limits.conf
$ echo "DefaultLimitNOFILE=1024000" >> /etc/systemd/system.conf $ echo "DefaultLimitNPROC=1024000" >> /etc/systemd/system.conf

4. 配置CentOS YUM源

配置国内tencent yum源地址、epel源地址、Kubernetes源地址
$ yum install -y wget $ rm -rf /etc/yum.repos.d/*
$ wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.cloud.tencent.com/repo/centos7_base.repo

$ wget -O /etc/yum.repos.d/epel.repo http://mirrors.cloud.tencent.com/repo/epel-7.repo
$ yum clean all && yum makecache

#配置国内Kubernetes源地址
$ cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF

5. 安装依赖软件包

有些依赖包我们要把它安装上，方便到时候使用
$ yum install -y conntrack ipvsadm ipset jq sysstat curl iptables libseccomp bash-completion yum-utils device-mapper-persistent-data lvm2 net-tools conntrack-tools vim libtool-ltdl

6. 时间同步配置

Kubernetes是分布式的，各个节点系统时间需要同步对应上。
$ yum install chrony –y
$ systemctl enable chronyd.service && systemctl start chronyd.service && systemctl status chronyd.service
$ chronyc sources

```
[root@Master ~]# chronyc sources
210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? 120.25.115.19                 2   6     1     1  +28797s[+28797s] +/- 5697us
^? 85.199.214.100                1   6     1     0  +28797s[+28797s] +/-  116ms
^? 85.199.214.101                1   6     1     0  +28797s[+28797s] +/-  105ms
^? 120.25.115.20                 2   6     1     1  +28797s[+28797s] +/- 5251us
```

运行date命令看下系统时间，过一会儿时间就会同步。

7. 配置节点间ssh互信

配置ssh互信，那么节点之间就能无密访问，方便日后执行自动化部署

$ ssh-keygen   # 每台机器执行这个命令， 一路回车即可
$ ssh-copy-id node   # 到master上拷贝公钥到其他节点，这里需要输入 yes和密码

8. 初始化环境配置检查
- 重启，做完以上所有操作，最好reboot重启一遍
- ping 每个节点hostname 看是否能ping通
- ssh 对方hostname看互信是否无密码访问成功
- 执行date命令查看每个节点时间是否正确
- 执行 ulimit -Hn 看下最大文件打开数是否是655360
- cat /etc/sysconfig/selinux |grep disabled 查看下每个节点selinux是否都是disabled状态

二、docker安装

Kubernetes 是容器调度编排PaaS平台，那么docker是必不可少要安装的。最新Kubernetes 1.13 支持最新的docker版本是18.06.1，那么我们就安装最新的 docker-ce 18.06.1

具体Kubernetes changelog 文档地址：
https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.12.md#v1123

1. remove旧版本docker
$ yum remove -y docker docker-ce docker-common docker-selinux docker-engine

2. 设置docker yum源
$ yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

3. 列出docker版本
$ yum list docker-ce --showduplicates | sort -r

4. 安装docker 指定18.06.1
$ yum install -y docker-ce-18.06.1.ce-3.el7

5. 配置镜像加速器和docker数据存放路径
$ tee /etc/docker/daemon.json <<-'EOF'
{
"registry-mirrors": ["https://q2hy3fzi.mirror.aliyuncs.com"],
"graph": "/tol/docker-data"
}
EOF

6. 启动docker
$ systemctl daemon-reload && systemctl restart docker && systemctl enable docker && systemctl status docker
# 查看docker 版本

$ docker --version

三、安装kubeadm、kubelet、kubectl

这一步是所有节点都得安装（包括node节点）

1. 工具说明

• kubeadm: 部署集群用的命令
• kubelet: 在集群中每台机器上都要运行的组件，负责管理pod、容器的生命周期
• kubectl: 集群管理工具

2. yum 安装

# 安装工具
$ yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

# 启动kubelet
$ systemctl enable kubelet && systemctl start kubelet

注意：kubelet 服务会暂时启动不了，先不用管它。

四、镜像下载准备

1. 初始化获取要下载的镜像列表

使用kubeadm来搭建Kubernetes，那么就需要下载得到Kubernetes运行的对应基础镜像，比如：kube-proxy、kube-apiserver、kube-controller-manager等等 。那么有什么方法可以得知要下载哪些镜像呢？从kubeadm v1.11+版本开始，增加了一个kubeadm config print-default 命令，可以让我们方便的将kubeadm的默认配置输出到文件中，这个文件里就包含了搭建K8S对应版本需要的基础配置环境。另外，我们也可以执行 kubeadm config images list 命令查看依赖需要安装的镜像列表。

```
[root@master ~]# kubeadm config images list
I1214 14:00:20.376044   20978 version.go:94] could not fetch a Kubernetes version from t
s.com/kubernetes-release/release/stable-1.txt: net/http: request canceled while waiting
I1214 14:00:20.376114   20978 version.go:95] falling back to the local client version: v
k8s.gcr.io/kube-apiserver:v1.13.0
k8s.gcr.io/kube-controller-manager:v1.13.0
k8s.gcr.io/kube-scheduler:v1.13.0
k8s.gcr.io/kube-proxy:v1.13.0
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.2.24
k8s.gcr.io/coredns:1.2.6
```

注意：这个列表显示的tag名字和镜像版本号，从Kubernetes v1.12+开始，镜像名后面不带amd64, arm, arm64, ppc64le 这样的标识了。

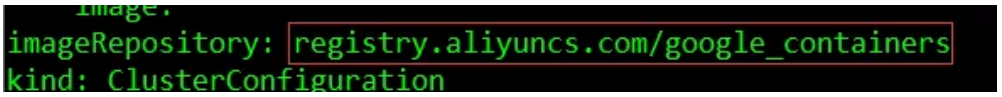
1.1 生成默认kubeadm.conf文件

# 执行这个命令就生成了一个kubeadm.conf文件
$ kubeadm config print init-defaults > kubeadm.conf

1.2 绕过墙下载镜像方法（注意认真看，后期版本安装也可以套用这方法）

注意这个配置文件默认会从google的镜像仓库地址k8s.gcr.io下载镜像，如果你没有科学上网，那么就会下载不来。因此，我们通过下面的方法把地址改成国内的，比如用阿里的：

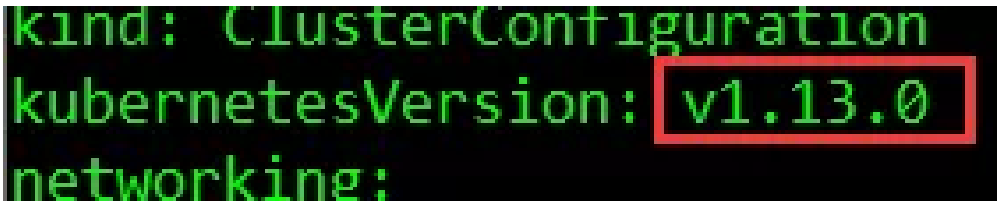
$ sed -i "s/imageRepository: .*/imageRepository: registry.aliyuncs.com/google_containers/g" kubeadm.conf



1.3 指定kubeadm安装的Kubernetes版本

我们这次要安装的Kubernetes版本是最新的v1.13，所以我们要修改下：

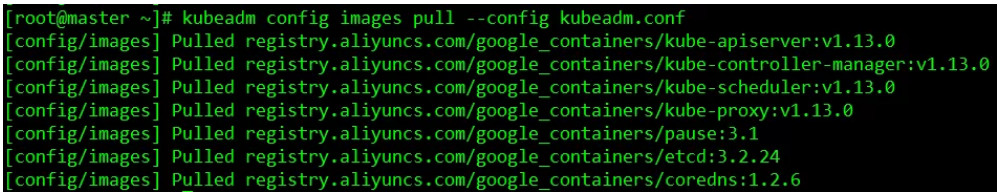
$ sed -i "s/kubernetesVersion: .*/kubernetesVersion: v1.13.0/g" kubeadm.conf



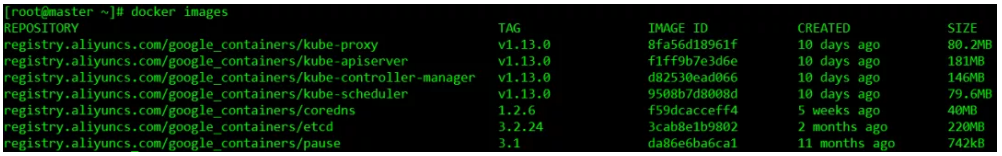
1.4 下载需要用到的镜像

kubeadm.conf修改好后，我们执行下面命令就可以自动从国内下载需要用到的镜像了：

$ kubeadm config images pull --config kubeadm.conf



自动下载v1.13需要用到的镜像，执行 docker images 可以看到下载好的镜像列表：



注：除了上面的方法，还有一种方式是搭建自己的镜像仓库。不过前提你得下载好对应的版本镜像，然后上传到你镜像仓库里，然后pull下载。不过上面我提到的方法更加方便省事。

1.5 docker tag 镜像

镜像下载好后，我们还需要tag下载好的镜像，让下载好的镜像都是带有 k8s.gcr.io 标识的，目前我们从阿里下载的镜像 标识都是，如果不打tag变成k8s.gcr.io，那么后面用kubeadm安装会出现问题，因为kubeadm里面只认google自身的模式。我们执行下面命令即可完成tag标识更换：

$ docker tag registry.aliyuncs.com/google_containers/kube-apiserver:v1.13.0 k8s.gcr.io/kube-apiserver:v1.13.0

$ docker tag registry.aliyuncs.com/google_containers/kube-controller-manager:v1.13.0 k8s.gcr.io/kube-controller-manager:v1.13.0

$ docker tag registry.aliyuncs.com/google_containers/kube-scheduler:v1.13.0 k8s.gcr.io/kube-scheduler:v1.13.0

$ docker tag registry.aliyuncs.com/google_containers/kube-proxy:v1.13.0 k8s.gcr.io/kube-proxy:v1.13.0

$ docker tag registry.aliyuncs.com/google_containers/pause:3.1 k8s.gcr.io/pause:3.1

$ docker tag registry.aliyuncs.com/google_containers/etcd:3.2.24 k8s.gcr.io/etcd:3.2.24

$ docker tag registry.aliyuncs.com/google_containers/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6

1.6 docker rmi 清理下载的镜像

执行完上面tag镜像的命令，我们还需要把带有 registry.aliyuncs.com 标识的镜像删除，执行：

$ docker rmi registry.aliyuncs.com/google_containers/kube-apiserver:v1.13.0

$ docker rmi registry.aliyuncs.com/google_containers/kube-controller-manager:v1.13.0

$ docker rmi registry.aliyuncs.com/google_containers/kube-scheduler:v1.13.0

$ docker rmi registry.aliyuncs.com/google_containers/kube-proxy:v1.13.0

$ docker rmi registry.aliyuncs.com/google_containers/pause:3.1

$ docker rmi registry.aliyuncs.com/google_containers/etcd:3.2.24

$ docker rmi registry.aliyuncs.com/google_containers/coredns:1.2.6

1.7 查看下载的镜像列表

执行docker images命令，即可查看到，这里结果如下，您下载处理后，结果需要跟这里的一致：

```
[root@master ~]# docker images
REPOSITORY                           TAG                 IMAGE ID            CREATED             SIZE
k8s.gcr.io/kube-proxy                v1.13.0             8fa56d18961f        10 days ago         80.2MB
k8s.gcr.io/kube-apiserver            v1.13.0             f1ff9b7e3d6e        10 days ago         181MB
k8s.gcr.io/kube-controller-manager   v1.13.0             d82530ead066        10 days ago         146MB
k8s.gcr.io/kube-scheduler            v1.13.0             9508b7d8008d        10 days ago         79.6MB
k8s.gcr.io/coredns                   1.2.6               f59dcacceff4        5 weeks ago         40MB
k8s.gcr.io/etcd                      3.2.24              3cab8e1b9802        2 months ago        220MB
k8s.gcr.io/pause                     3.1                 da86e6ba6ca1        11 months ago       742kB
```

注：以上操作其实可以写到一个脚本里，然后自动处理。另外两个master节点，重复上面的操作下载即可。

五、部署master节点

1. kubeadm init 初始化master节点

$ kubeadm init --kubernetes-version=v1.13.0 --pod-network-cidr=172.22.0.0/16 --apiserver-advertise-address=192.168.1.11

这里我们定义POD的网段为: 172.22.0.0/16 ，然后api server地址就是master本机IP地址。

```
[root@master ~]# kubeadm init --kubernetes-version=v1.13.0 --pod-network-cidr=172.22.0.0/16 --apiserver-advertise-address=192.168.1.11
[init] Using Kubernetes version: v1.13.0
[preflight] Running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 18.09.0. Latest validated version: 18.06
[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'
[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] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [master kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.11]
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [master localhost] and IPs [192.168.1.11 127.0.0.1 ::1]
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [master localhost] and IPs [192.168.1.11 127.0.0.1 ::1]
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
```

2. 初始化成功后，/etc/kubernetes/ 会生成下面文件

```
[root@master01 ~]# cd /etc/kubernetes/
[root@master01 kubernetes]# ll
total 36
-rw------- 1 root root 5448 Dec  6 11:49 admin.conf
-rw------- 1 root root 5484 Dec  6 11:49 controller-manager.conf
-rw------- 1 root root 5472 Dec  6 11:49 kubelet.conf
drwxr-xr-x 2 root root  113 Dec  6 11:49 manifests
drwxr-xr-x 3 root root 4096 Dec  6 11:49 pki
-rw------- 1 root root 5436 Dec  6 11:49 scheduler.conf
```

3. 同时最后会生成一句话

kubeadm join 192.168.1.11:6443 --token zs4s82.r9svwuj78jc3px43 --discovery-token-ca-cert-hash sha256:45063078d23b3e8d33ff1d81e903fac16fe6c8096189600c709e3bf0ce051ae8

这个我们记录下，到时候添加node的时候要用到。

4. 验证测试

配置kubectl命令

$ mkdir -p /root/.kube

$ cp /etc/kubernetes/admin.conf /root/.kube/config

执行获取pods列表命令，查看相关状态

$ kubectl get pods --all-namespaces

```
[root@master kubernetes]# kubectl get pods --all-namespaces
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-86c58d9df4-5vnbq          0/1     Pending   0          2m18s
kube-system   coredns-86c58d9df4-ks5vd          0/1     Pending   0          2m18s
kube-system   etcd-master                       1/1     Running   0          84s
kube-system   kube-apiserver-master             1/1     Running   0          80s
kube-system   kube-controller-manager-master    1/1     Running   0          83s
kube-system   kube-proxy-mw7gf                  1/1     Running   0          2m18s
kube-system   kube-scheduler-master             1/1     Running   0          88s
```

其中coredns pod处于Pending状态，这个先不管。

我们也可以执行 kubectl get cs 查看集群的健康状态：

```
[root@master kubernetes]# kubectl get cs
NAME                 STATUS    MESSAGE             ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health": "true"}
```

六、部署calico网络 （在master上执行）

calico介绍：Calico是一个纯三层的方案，其好处是它整合了各种云原生平台(Docker、Mesos 与 OpenStack 等)，每个 Kubernetes 节点上通过 Linux Kernel 现有的 L3 forwarding 功能来实现 vRouter 功能。

1. 下载calico 官方镜像

我们这里要下载三个镜像，分别是calico-node:v3.1.4、calico-cni:v3.1.4、calico-typha:v3.1.4

直接运行 docker pull 下载即可

$ docker pull calico/node:v3.1.4

$ docker pull calico/cni:v3.1.4
$ docker pull calico/typha:v3.1.4

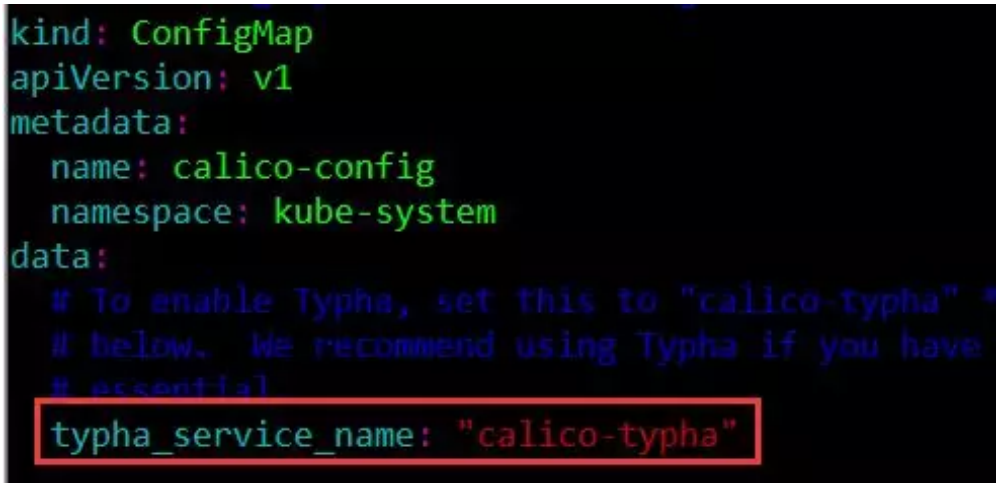
2. tag 这三个calico镜像
$ docker tag calico/node:v3.1.4 quay.io/calico/node:v3.1.4
$ docker tag calico/cni:v3.1.4 quay.io/calico/cni:v3.1.4
$ docker tag calico/typha:v3.1.4 quay.io/calico/typha:v3.1.4

3. 删除原有镜像
$ docker rmi calico/node:v3.1.4$ docker rmi calico/cni:v3.1.4
$ docker rmi calico/typha:v3.1.4

4. 部署calico
4.1 下载执行rbac-kdd.yaml文件
$ curl https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml -O
$ kubectl apply -f rbac-kdd.yaml

4.2 下载、配置calico.yaml文件
$ curl https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/kubernetes-datastore/policy-only/1.7/calico.yaml -O

把ConfigMap 下的 typha_service_name 值由none变成 calico-typha



设置 Deployment 类目的 spec 下的replicas值

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: calico-typha
  namespace: kube-system
  labels:
    k8s-app: calico-typha
spec:
  # Number of Typha replicas.  To enable Typ
  # typha_service_name variable in the calic
  #
  # We recommend using Typha if you have mor
  # (when using the Kubernetes datastore).
  # production, we recommend running at leas
  replicas: 1
  revisionHistoryLimit: 2
  template:
    metadata:
      labels:
```

我们这里设置为 1。

4.3 定义POD网段

我们找到CALICO_IPV4POOL_CIDR，然后值修改成之前定义好的POD网段，我这里是172.22.0.0/16

```
  - name: CALICO_IPV4POOL_CIDR
    value: "172.22.0.0/16"
```

4.4 开启bird模式

把 CALICO_NETWORKING_BACKEND 值设置为 bird ， 这个值是设置BGP网络后端模式

```
- name: CALICO_NETWORKING_BACKEND
  value: "bird"
```

4.5 部署calico.yaml文件

上面参数设置调优完毕，我们执行下面命令彻底部署calico

$ kubectl apply -f calico.yaml

查看状态

$ kubectl get pods --all-namespaces

```
[root@master ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   calico-node-fqkvn                 1/2     Running   1          59s
kube-system   calico-typha-588d4f649b-qp8wg     0/1     Pending   0          59s
kube-system   coredns-86c58d9df4-5vnbq          1/1     Running   0          11m
kube-system   coredns-86c58d9df4-ks5vd          1/1     Running   0          11m
kube-system   etcd-master                       1/1     Running   0          10m
kube-system   kube-apiserver-master             1/1     Running   0          10m
kube-system   kube-controller-manager-master    1/1     Running   0          10m
kube-system   kube-proxy-mw7gf                  1/1     Running   0          11m
kube-system   kube-scheduler-master             1/1     Running   0          11m
```

这里calico-typha 没起来，那是因为我们的node 计算节点还没启动和安装。

七、部署node节点

1. 下载安装镜像（在node上执行）

node上也是需要下载安装一些镜像的，需要下载的镜像为：kube-proxy:v1.13、pause:3.1、calico-node:v3.1.4、calico-cni:v3.1.4、calico-typha:v3.1.4

1.1 下载镜像

$ docker pull registry.aliyuncs.com/google_containers/kube-proxy:v1.13.0

$ docker pull registry.aliyuncs.com/google_containers/pause:3.1

$ docker pull calico/node:v3.1.4$ docker pull calico/cni:v3.1.4

$ docker pull calico/typha:v3.1.4

$ docker tag registry.aliyuncs.com/google_containers/kube-proxy:v1.13.0 k8s.gcr.io/kube-proxy:v1.13.0

$ docker tag registry.aliyuncs.com/google_containers/pause:3.1 k8s.gcr.io/pause:3.1

$ docker tag calico/node:v3.1.4 quay.io/calico/node:v3.1.4

$ docker tag calico/cni:v3.1.4 quay.io/calico/cni:v3.1.4

$ docker tag calico/typha:v3.1.4 quay.io/calico/typha:v3.1.4

$ docker rmi registry.aliyuncs.com/google_containers/kube-proxy:v1.13.0

$ docker rmi registry.aliyuncs.com/google_containers/pause:3.1

$ docker rmi calico/node:v3.1.4

$ docker rmi calico/cni:v3.1.4

$ docker rmi calico/typha:v3.1.4

1.2. 把node加入集群里

加node计算节点非常简单，在node上运行：

$ kubeadm join 192.168.1.11:6443 --token zs4s82.r9svwuj78jc3px43 --discovery-token-ca-cert-hash sha256:45063078d23b3e8d33ff1d81e903fac16fe6c8096189600c709e3bf0ce051ae8

两个节点运行的参数命令一样，运行完后，我们在master节点上运行 kubectl get nodes 命令查看node是否正常

```
[root@master ~]# kubectl get nodes
NAME      STATUS   ROLES    AGE     VERSION
master    Ready    master   21m     v1.13.0
node      Ready    <none>   4m33s   v1.13.0
```

到此，集群的搭建完成了90%，剩下一个是搭建dashboard。

八、部署dashboard

部署dashboard之前，我们需要生成证书，不然后面会https访问登录不了。

1. 生成私钥和证书签名请求

$ mkdir -p /etc/kubernetes/certs $ cd /etc/kubernetes/certs

$ openssl genrsa -des3 -passout pass:x -out dashboard.pass.key 2048

```
[root@master01 certs]#  openssl genrsa -des3 -passout pass:x -out dashboard.pass.key 2048
Generating RSA private key, 2048 bit long modulus
......+++
.................+++
e is 65537 (0x10001)
[root@master01 certs]# ll
total 4
-rw-r--r-- 1 root root 1751 Dec  6 23:30 dashboard.pass.key
```

$ openssl rsa -passin pass:x -in dashboard.pass.key -out dashboard.key

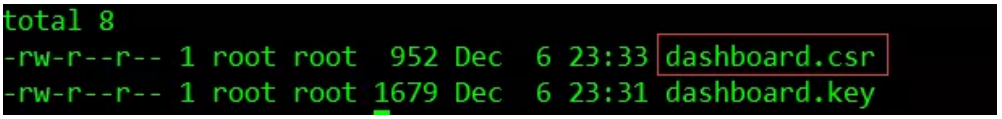
```
[root@master01 certs]# openssl rsa -passin pass:x -in dashboard.pass.key -out dashboard.key
writing RSA key
[root@master01 certs]# ll
total 8
-rw-r--r-- 1 root root 1679 Dec  6 23:31 dashboard.key
-rw-r--r-- 1 root root 1751 Dec  6 23:30 dashboard.pass.key
```

删除刚才生成的dashboard.pass.key

$ rm -rf dashboard.pass.key$ openssl req -new -key dashboard.key -out dashboard.csr

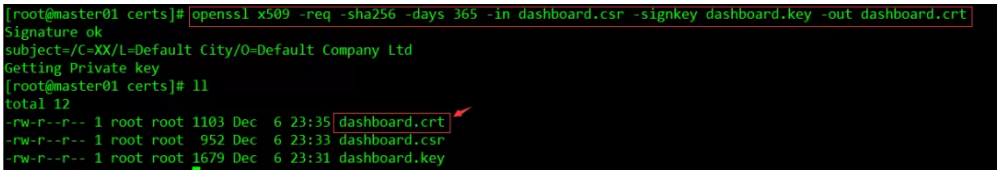
```
[root@master01 certs]# openssl req -new -key dashboard.key -out dashboard.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:                    一路回车
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

生成了dashboard.csr



生成SSL证书

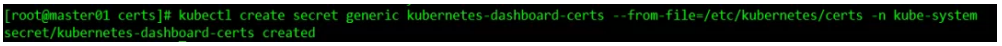
$ openssl x509 -req -sha256 -days 365 -in dashboard.csr -signkey dashboard.key -out dashboard.crt



dashboard.crt文件是适用于仪表板和dashboard.key私钥的证书。

创建secret

$ kubectl create secret generic kubernetes-dashboard-certs --from-file=/etc/kubernetes/certs -n kube-system



注意/etc/kubernetes/certs 是之前创建crt、csr、key 证书文件存放的路径

2. 下载dashboard镜像、tag镜像（在全部节点上）

$ docker pull registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0

$ docker tag registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0 k8s.gcr.io/kubernetes-dashboard:v1.10.0

$ docker rmi registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0

下载 kubernetes-dashboard.yaml 部署文件（在master上执行）

curl https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml -O

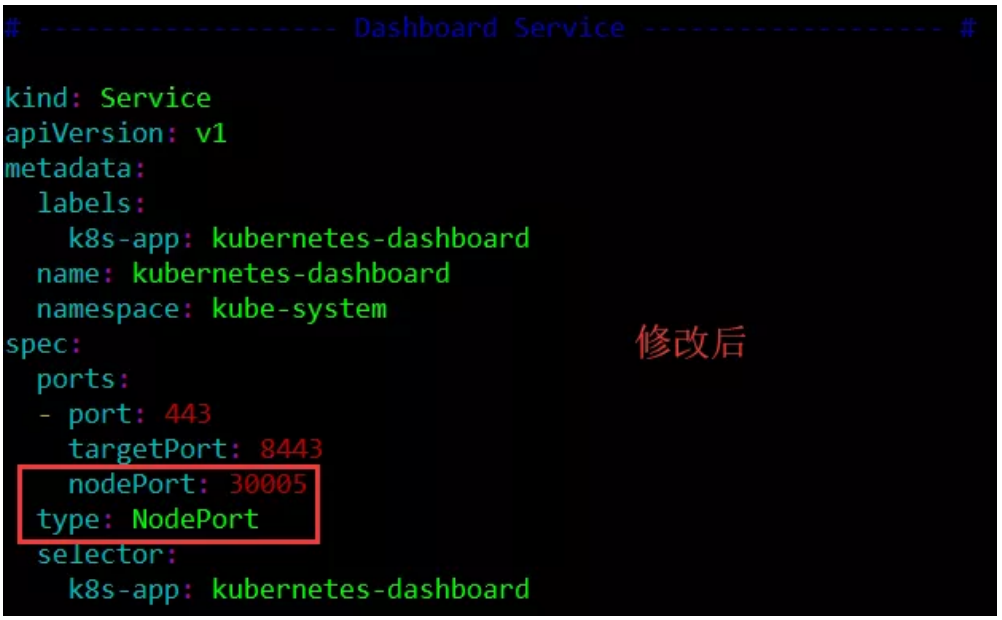
修改 kubernetes-dashboard-amd64:v1.10.0 为 kubernetes-dashboard:v1.10.0，不然会去墙外下载dashboard镜像

$ sed -i "s/kubernetes-dashboard-amd64:v1.10.0/kubernetes-dashboard:v1.10.0/g" kubernetes-dashboard.yaml

3. 把Secret 注释

因为上面我们已经生成了密钥认证了，我们用我们自己生成的。

4. 配置443端口映射到外部主机30005上



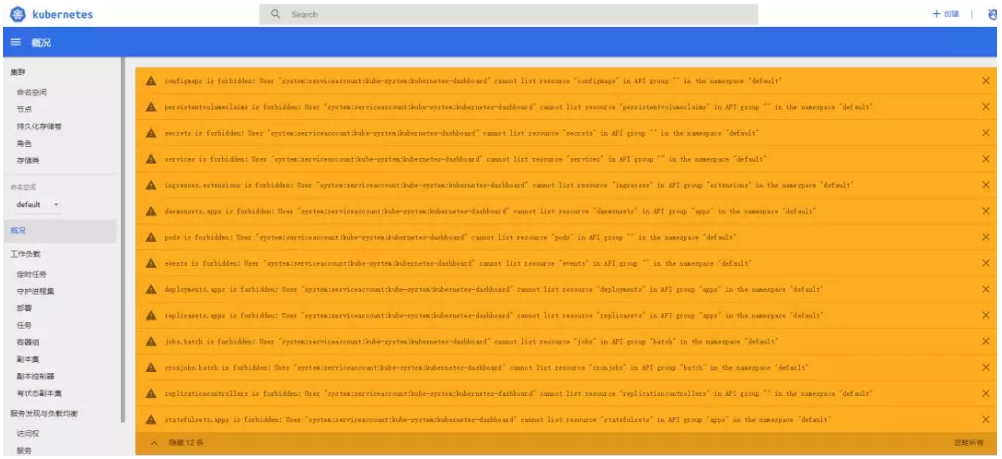
5. 创建dashboard的pod

$ kubectl create -f kubernetes-dashboard.yaml

6. 查看服务运行情况

$ kubectl get deployment kubernetes-dashboard -n kube-system$ kubectl --namespace kube-system get pods -o wide
$ kubectl get services kubernetes-dashboard -n kube-system
$ netstat -ntlp|grep 30005

7. Dashboard BUG处理

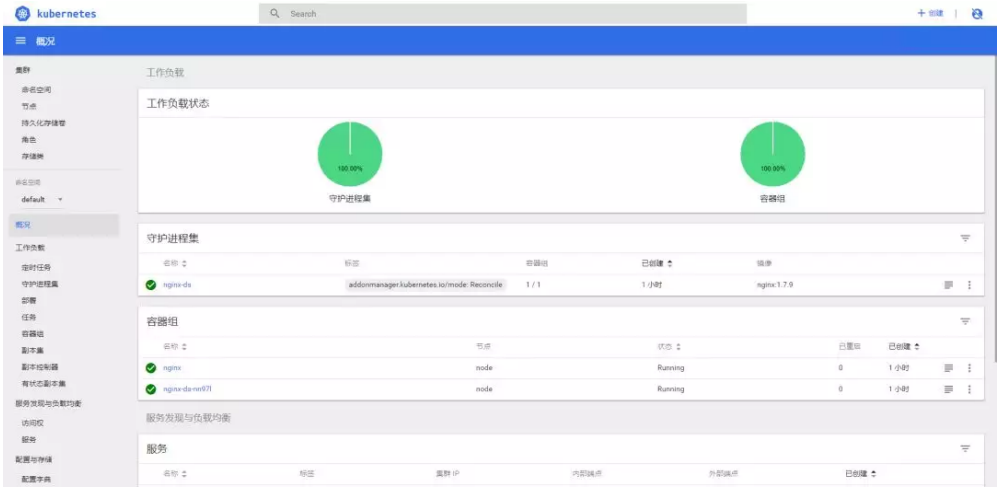


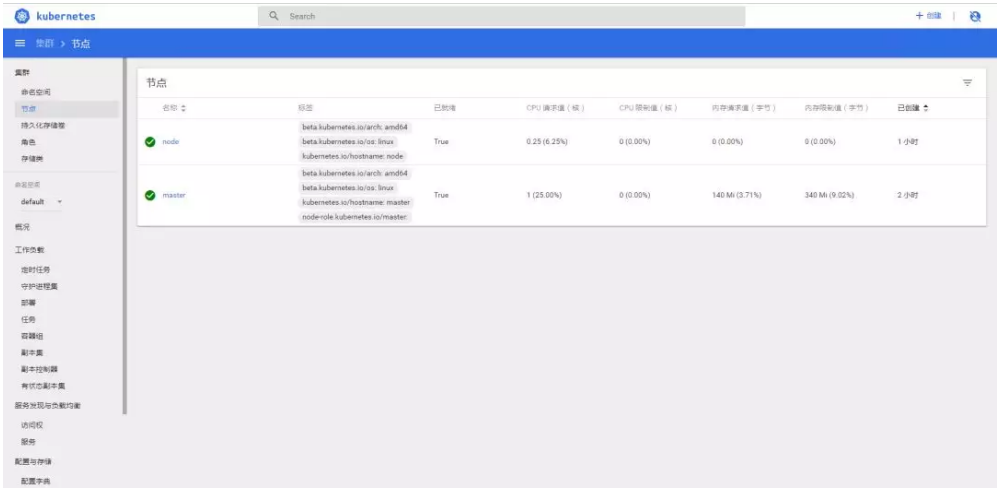
$ vim kube-dashboard-access.yaml

添加下面内容：
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
name: kubernetes-dashboard
labels:
k8s-app: kubernetes-dashboard
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: cluster-admin
subjects:
- kind: ServiceAccount
name: kubernetes-dashboard
namespace: kube-system

执行让kube-dashboard-access.yaml 生效
$ kubectl create -f kube-dashboard-access.yaml

然后重新登录界面刷新下，这个问题即可解决。


kubernetes
Search
概况
工作负载
工作负载状态
守护进程集
容器组
服务发现与负载均衡
服务


kubernetes
Search
集群 › 节点
节点
node
master