Kubernetes安装

1. **环境准备**

* Ubuntu 14.04 LTS
* Kubernetes 1.1.1
* Etcd 2.2.1
* Flannel 0.5.5
* Docker 1.9.0

Kubernetes集群:

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Hostname | Interface | IP Adress |
| Kube master | db06b03 | Eth0 | 9.21.54.63 |
| Kuber minion | db06b03 | Eth0 | 9.21.54.63 |
| Kuber minion | db06b05 | Eth0 | 9.21.54.65 |
| Kuber minion | db06b06 | Eth0 | 9.21.54.66 |

1. **在minion上安装docker**

从Docker官方源安装最新版本的docker，首先需要安装apt-transport-https,并添加Docker官方源：

**$ sudo apt-get install apt-transport-https**

**$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F9**

**66E92D8576A8BA88D21E9**

**$ sudo bash –c “echo deb** [**https://get.docker.io/ubuntu**](https://get.docker.io/ubuntu) **docker main > /etc/apt/sources.list.d/docker.list”**

**$ sudo apt-get update**

之后可以通过下面的命令来安装uzixinbanben的Docker：

**$ sudo apt-get install –y lxc-docker**

1. **安装etcd**

首先我们要在master上安装并启动etcd服务，步骤如下。

1. 下载etcd最新的tar包，地址为:

<https://github.com/coreos/etcd/releases/download/v2.2.1/etcd-v2.2.1-linux-amd64.tar.gz> ，解压etcd和etcdctl到/usr/bin/目录下。

**# tar zxvf etcd-v2.2.1-linux-amd64.tar.gz**

**# cd etcd-v2.2.1-linux-amd64/**

**#cp etcd etcdctl /usr/bin/**

1. 启动etcd服务，我们的kube master和etcd运行在同一台机器上所以使用默认配置就可以。

**# etcd &**

1. **安装配置flannel**
2. 启动flannel server

我们把kube master作为flannel server。

* 首先向etcd注册flannel维护的网段，在master上执行命令：

**# curl -L http://127.0.0.1:4001/v2/keys/coreos.com/network/config -XPUT -d value=**

**'{ "Network": "10.100.0.0/16"}**

* 之后，启动flannel server服务，我们设置端口为1234：

**# flanneld –listen=0.0.0.0:1234**

1. 启动flannel client

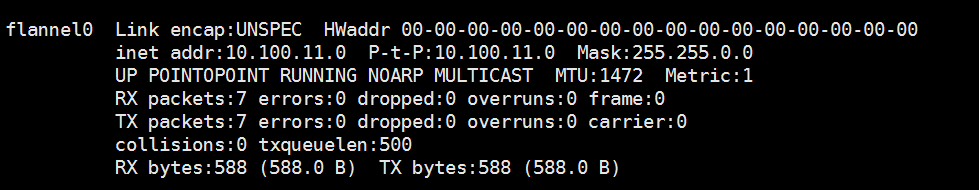
在每个minion node上执行以下操作：

**# flanneld –remote=9.21.54.63:1234 &**

【注】：在master这台机器上也可以使用 flannel –remote=127.0.0.1:1234 &

1. 配置docker0网桥

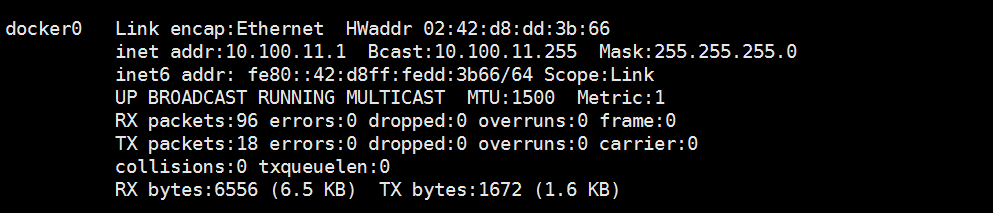
我们可以使用命令ifconfig来查看各个minion分配到的子网，如下图，这个机器分到的子网为10.100.11.0/18.



然后我们把docker0的网络设置成这个子网，运行下面的命令修改docker0的ip、netmask和MTU:

**# ifconfig docker0 10.100.11.1 netmask 255.255.255.0 mtu 1472**

之后可以再次执行ifconfig检查是否修改成功。



1. 之后需要重启docker服务，并指定网桥为docker0

**# service docker stop**

**# docker –d –b=docker0**

1. 按照上面的步骤将其余两台机器配置好，三台机器之间的container就可以互相通信了。
2. **安装配置kubernetes**

这里是使用的binary直接启动服务，没有配置成service。

Kubernetes下载地址：https://github.com/kubernetes/kubernetes/releases/download/v1.1.1/kubernetes.tar.gz

1. 在master上解压安装包并复制api-server、scheduler和controller-manager到/usr/bin,接下来开始启动服务。首先保证etcd服务正常，然后执行：

**# kube-apiserver --etcd-servers=http://127.0.0.1:2379 --insecure-bind-address=0.0.0.0 --insecure-port=8080 --logtostderr=true --service-cluster-ip-range=10.200.20.0/24 --v=2 &**

**# kube-controller-manager --logtostderr=true --master=http://127.0.0.1:8080 --v=2 &**

**# kube-scheduler --master=http://127.0.0.1:8080 --v=2 &**

1. 在3台minion上启动kubelet:首先解压kubelet到/usr/bin/

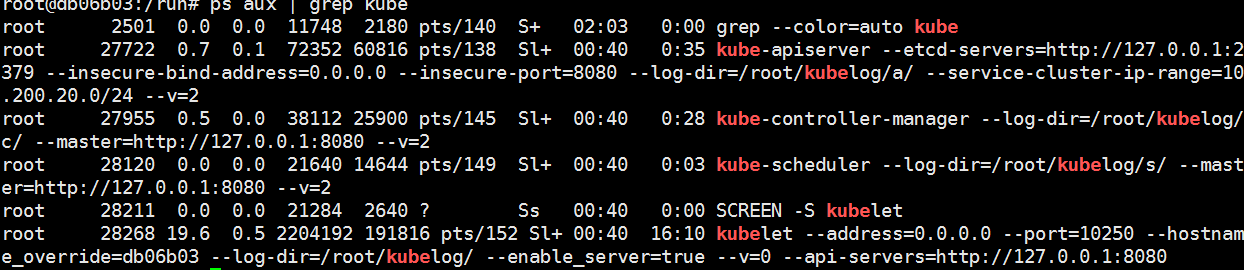
**# kubelet --address=0.0.0.0 --port=10250 --hostname\_override=db06b03 --enable\_server=true --v=0 --api-servers=http://9.21.54.63:8080**

**# kubelet --address=0.0.0.0 --port=10250 --hostname\_override=db06b05 --enable\_server=true --v=0 --api-servers=http://9.21.54.63:8080**

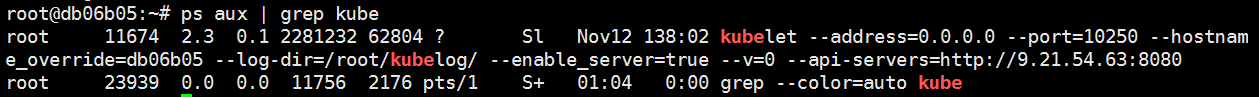
**# kubelet --address=0.0.0.0 --port=10250 --hostname\_override=db06b06 --enable\_server=true --v=0 --api-servers=http://9.21.54.63:8080**

到此，kubernetes就安装完成了。可以使用命令 ps aux | grep kube 查看进程是否启动：

1、master和minion1



2、其余两个minion：



1. **导入pause镜像**

在开始使用kubernetes之前需要导入一个镜像——gcr.io/google\_containers/pause:0.8.0

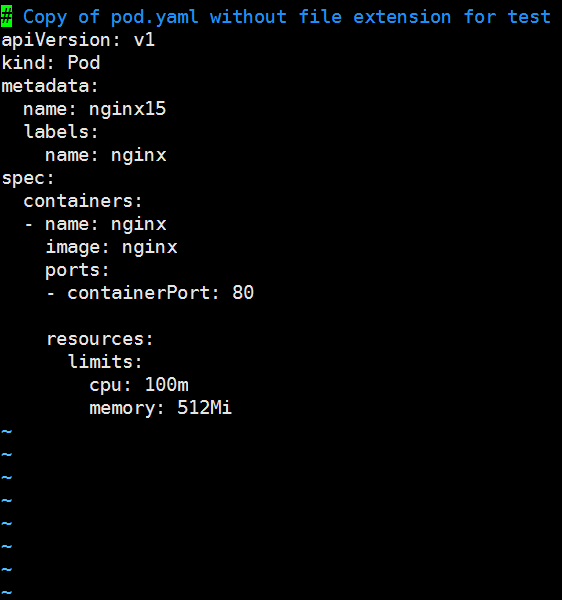
可能用到的命令 ： **# docker load –-input= <pause\_image\_name.tar>**

**# docker tag <pause\_image\_name.tar> gcr.io/google\_containers/pause:0.8.0**

1. **创建pod**

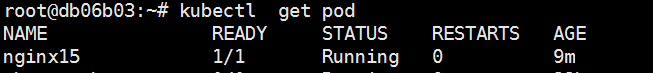
我们用下面的yaml文件创建一个pod：

**# kubectl create –f pod1.yaml**

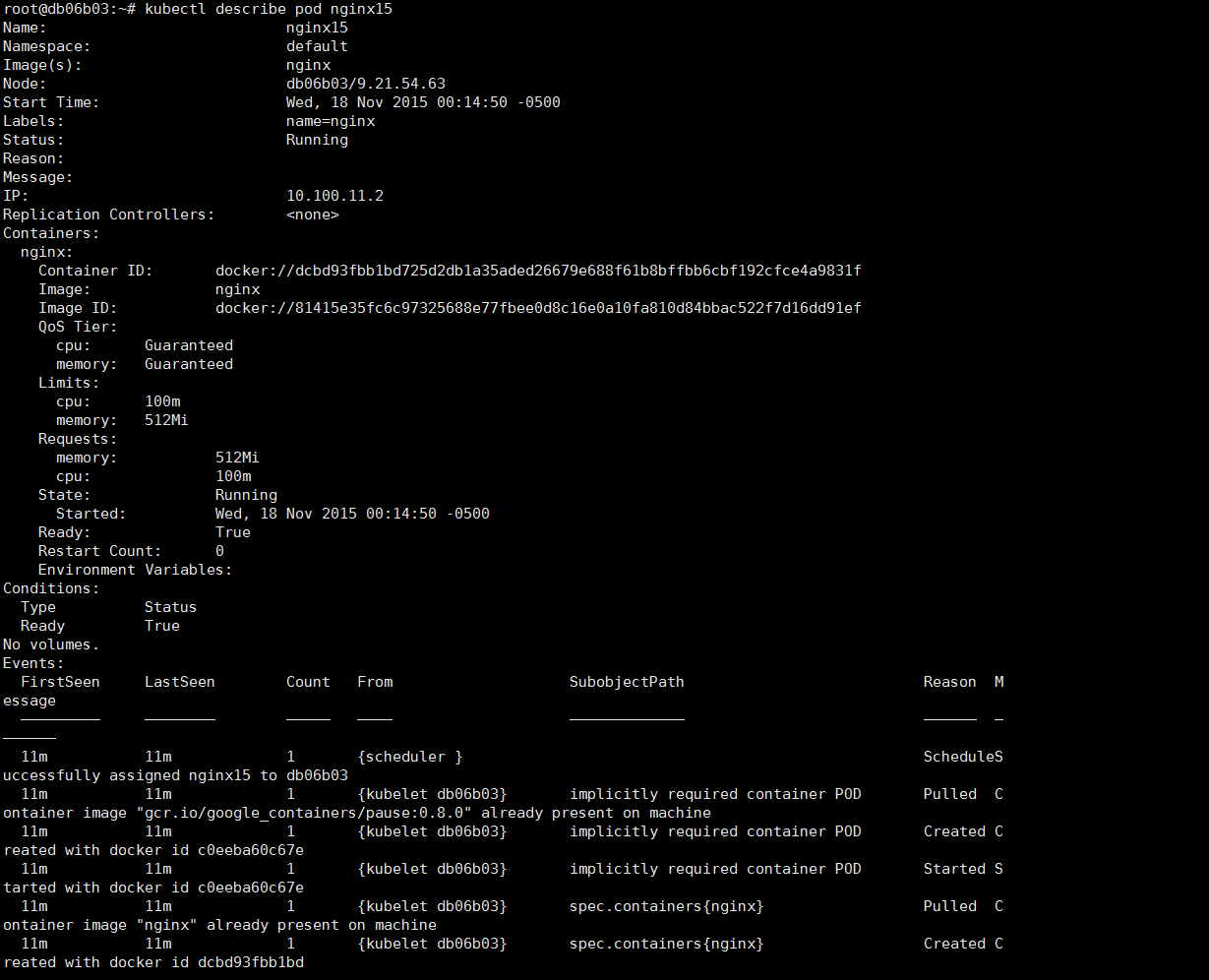


检查pod运行情况：

**# kubectl get pod**



**# kubectl describe pod nginx15**



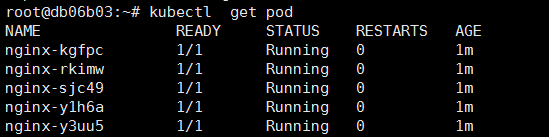
1. ReplicationController测试

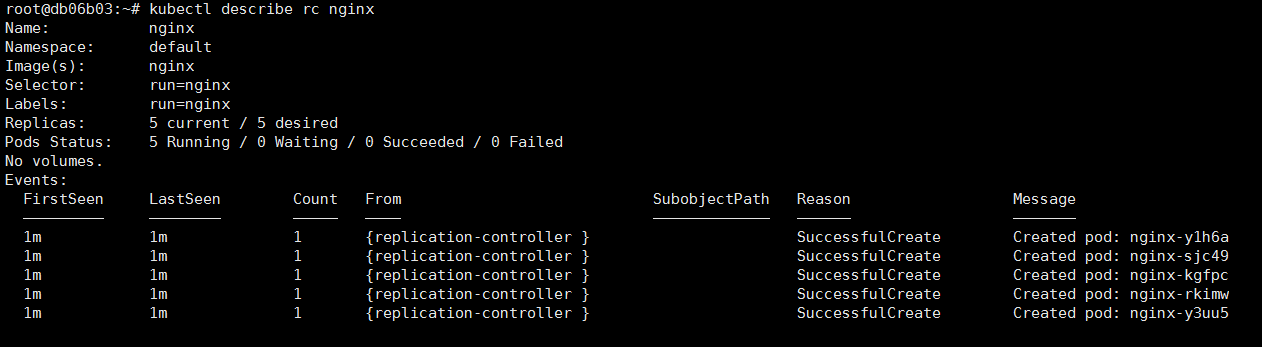
* 首先创建一个replicationController(简称rc)，副本份数为5，使用nginx镜像：

# kubectl run nginx –image=nginx –replicas=5

* 查看一下rc和pod的状态：# kubectl get pod

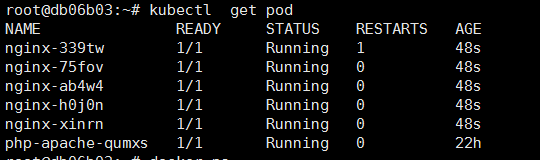
# kubectl describe rc nginx





* 去各个node上看一下这5个pod的分布情况： # docker ps
* 结果为1：2：2.可见系统是按照负载均衡算法分配的节点。下面在一个节点删除其中一个pod然后观察pod情况：# docker stop xxxx (podID)

# kubectl get pod



* 可以看到restarts变成了1，说明这个pod已经重新被建立。ReplicationController功能正常。