**一、拓扑结构**

1 master + 2 minion

k8s-master 10.67.147.217 master

k8s-node1 10.67.147.218 minion

k8s-node2 10.67.147.219 minion

二、准备工作

注：所有指令均切换到root用户状态下执行。

### ****Docker安装****

1、下载docker的bin文件，在三个主机上都安装Docker，版本此处安装的是1.9.1 （如已经安装过docker，可不用安装）

下载：wget https://get.docker.com/builds/Linux/x86\_64/docker-1.9.1

重命名：mv docker-1.9.1 docker

修改权限：chmod 777 docker

拷贝到/usr/bin目录： cp docker /usr/bin

2、运行docker

执行：docker daemon &

### ****SSH登录****

需要所有minion节点都能从master节点直接用root账户登录，不用密码。

1、分别在三个主机上进行如下操作：

# ssh-keygen #连续按三次回车键（默认设置）， 生成SSH连接时所需要的key

# vim /etc/ssh/sshd\_config #配置允许ssh远程root用户登录

屏蔽第28行：

#PermitRootLogin without-password

在下一行增加：

PermitRootLogin yes

重启SSH服务

# service ssh restart

修改root用户密码

# passwd root #注意：通常ubuntu安装之后root用户默认是空密码的，如果不修改root密码，用ssh登录时将会提示权限受限导致无法登录。

2、在master主机上进行如下操作：

# ssh-copy-id root@10.67.147.217

# ssh-copy-id root@10.67.147.218

# ssh-copy-id root@10.67.147.219

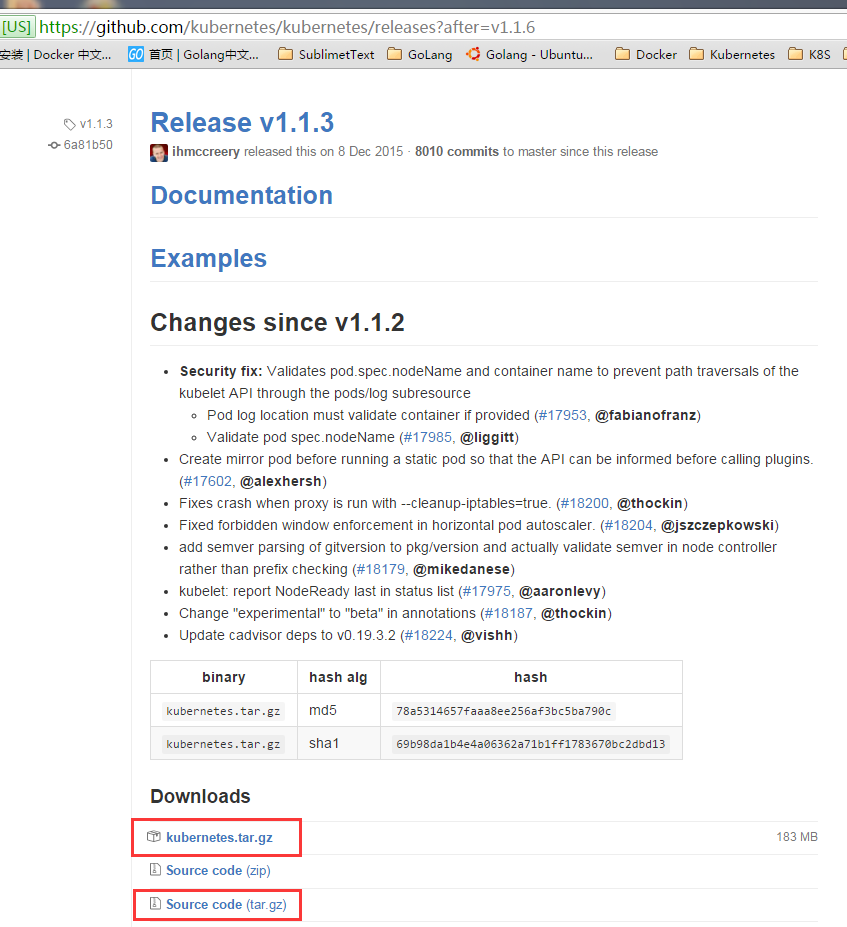
如弹出对话yes/no选项，均选择yes，然后按回车。

三、部署kubernetes环境

# 在master主机下操作

1、下载kubernetes-1.1.3.tar.gz源码包(大小：21.4M Byte)，下载地址https://github.com/kubernetes/kubernetes/releases

下载kubernetes.tar.gz二进制文件（大小：183M Byte），下载地址https://github.com/kubernetes/kubernetes/releases



下载[flannel-0.5.5-linux-amd64.tar.gz](https://github.com/coreos/flannel/releases/download/v0.5.5/flannel-0.5.5-linux-amd64.tar.gz)文件，下载地址https://github.com/coreos/flannel/releases/

下载[etcd-v2.2.1-linux-amd64.tar.gz](https://github.com/coreos/etcd/releases/download/v2.2.1/etcd-v2.2.1-linux-amd64.tar.gz)文件，下载地址https://github.com/coreos/etcd/releases/

解压编译kubernetes

#tar zxvf kubernetes-1.1.3.tar.gz

#mv kubernetes-1.1.3 kubernetes

#cd kubernetes

#make all

编译成功后在kubernetes根目录下会生成\_output文件夹

root@ubuntu:/home/workspace/src/kubernetes# ls

api code-of-conduct.md docs LICENSE Makefile shippable.yml

build contrib examples logo.pdf \_output test

CHANGELOG.md CONTRIB.md Godeps logo.png pkg third\_party

cluster CONTRIBUTING.md hack logo.svg plugin Vagrantfile

cmd DESIGN.md hooks logo\_usage\_guidelines.md README.md www

2、配置各组件文件

# cd cluster/Ubuntu

将下载好的[flannel-0.5.5-linux-amd64.tar.gz](https://github.com/coreos/flannel/releases/download/v0.5.5/flannel-0.5.5-linux-amd64.tar.gz)，[etcd-v2.2.1-linux-amd64.tar.gz](https://github.com/coreos/etcd/releases/download/v2.2.1/etcd-v2.2.1-linux-amd64.tar.gz)**,** kubernetes.tar.gz (183M Byte)文件复制到当前目录

重命名[flannel-0.5.5-linux-amd64.tar.gz](https://github.com/coreos/flannel/releases/download/v0.5.5/flannel-0.5.5-linux-amd64.tar.gz) 为flannel.tar.gz

重命名[etcd-v2.2.1-linux-amd64.tar.gz](https://github.com/coreos/etcd/releases/download/v2.2.1/etcd-v2.2.1-linux-amd64.tar.gz) 为etcd.tar.gz

root@ubuntu:/home/workspace/src/kubernetes/cluster/ubuntu# ls

build.sh config-test.sh etcd.tar.gz kubernetes.tar.gz reconfDocker.sh

config-default.sh deployAddons.sh flannel.tar.gz namespace.yaml util.sh

修改各组件的版本，下载所有组件的二进制文件。此处使用etcd 2.2.1，flannel 0.5.5，kubernetes 1.1.3。

# vim build.sh

#分别修改版本号

FLANNEL\_VERSION=${FLANNEL\_VERSION:-"0.5.5"} #在36行

ETCD\_VERSION=${ETCD\_VERSION:-"2.2.1"} #在46行

KUBE\_VERSION=${KUBE\_VERSION:-"1.1.3"} #在56行

由于相关组件的二进制文件已经手动下载好，所以需要屏蔽build.sh中相关的下载连接脚本

分别屏蔽第38、39、41、49、50、52、58、59、61行的内容，然后保存并退出文件。

在当前目录执行build.sh脚本，生成binaries文件夹

# ./build.sh

拷贝binaries目录的kubectl文件到/usr/bin目录

#cp binaries/kubectl /usr/bin

3、根据实际情况修改config-default.sh

# vim config-default.sh

修改第21行为：

export nodes=${nodes:-"root@10.67.147.217 root@10.67.147.218 root@10.67.147.219"}

修改第24行为：

role=${role:-"a i i"} 修改为: role=${role:-"ai"} #｛maser，minion，minion｝

修改第30行为：

export NUM\_MINIONS=${NUM\_MINIONS:-2} # minion节点的数目改为2

#如果需要master节点也作为minion节点的话，只需将role中的a改为ai即可，NUM\_MINIONS表示minion节点的个数。

在最后一行设置代理（注：在黄区的主机是通过设置代理才能连外网的，在这里必需设置代理，否则不能往下执行,在开放区则不用设置）

PROXY\_SETTING=${PROXY\_SETTING:-"https\_proxy=http://用户名:密码@proxyhk.huawei.com:8080"}

4、分别在三个主机上的/etc/hosts文件中追加一行内容：

10.67.147.217 localhost # 10.67.147.217为master主机IP

5、退回到上一级目录即kubernetes/cluster，执行下面指令

#export KUBERNETES\_PROVIDER=ubuntu

#./kube-up.sh // 出问题重新执行./kube-down.sh

#接下来脚本就会自动将二进制文件拷贝到相应的节点上。

配置DNS

1、进入到kubernetes/cluster/addons/dns/目录,修改sky-rc.yaml.in文件

#vim skydns-rc.yaml.in

在大概49行，即# command = "/kube2sky"的下一行增加一行内容：

- --kube\_master\_url=http://10.67.147.217:8080 # 注：10.67.147.217 为master主机的ip

2、进入到kubernetes/cluster/ubuntu目录

#vim config-default.sh

DNS的默认参数配置在config-default.sh文件中（根据实际情况配置）

DNS\_SERVER\_IP="192.168.3.10"

DNS\_DOMAIN="cluster.local"

DNS\_REPLICAS=1

3、执行下面的指令

# KUBERNETES\_PROVIDER=ubuntu ./deployAddons.sh

查看dns pod状态

#kubectl get pods --namespace "kube-system"

NAME READY STATUS RESTARTS AGE

kube-dns-v9-043v6 4/4 Running 0 1d

kube-ui-v4-zaz75 1/1 Running 0 1d

测试DNS工作是否正常

1、创建busybox.yaml 文件

#vim busybox.yaml #输入下面的内容：

apiVersion: v1

kind: Pod

metadata:

name: busybox

namespace: default

spec:

containers:

- image: busybox

command:

- sleep

- "3600"

imagePullPolicy: IfNotPresent

name: busybox

restartPolicy: Always

2、创建一个busybox pod

#kubectl create -f busybox.yaml

查看busybox pod运行状态

# kubectl get pods busybox

将会看到：

NAME READY REASON RESTARTS AGE

busybox 1/1 Running 0 18s

3、确认DNS正常工作

当busybox处在Running状态之后，执行指令：

#kubectl exec busybox -- nslookup kubernetes.default.svn.cluster.local

将会看到类似下面的内容：

Server: 192.168.3.10

Address 1: 192.168.3.10

Name: kubernetes.default.svn.cluster.local

Address 1: 192.168.3.1

看到上面的信息,说明 DNS已经正常工作。

4、用Guestbook 例子测试kubernetes环境

进入到kubernetes根目录

第一步 建立redis master replication pod

#kubectl create -f examples/guestbook/redis-master-controller.yaml

第二步 创建redis master service

# kubectl create -f examples/guestbook/redis-master-service.yaml

第三步 建立redis slave replication pod

#kubectl create -f examples/guestbook/redis-slave-controller.yaml

第四步 建立redis slave service

#kubectl create -f examples/guestbook/redis-slave-service.yaml

第五步 建立redis frontend replication pod

# kubectl create -f examples/guestbook/frontend-controller.yaml

第六步 建立redis frontend service

# kubectl create -f examples/guestbook/frontend-service.yaml

查看pods，services，rc列表

# kubectl get pods

NAME READY STATUS RESTARTS AGE

frontend-4o11g 1/1 Running 0 2h

frontend-u9aq6 1/1 Running 0 2h

frontend-yga1l 1/1 Running 0 2h

redis-master-dz33o 1/1 Running 0 2h

redis-slave-35mer 1/1 Running 0 2h

redis-slave-iqkhy 1/1 Running 0 2h

# kubectl get services

NAME CLUSTER\_IP EXTERNAL\_IP PORT(S) SELECTOR AGE

frontend 192.168.3.158 <none> 80/TCP name=frontend 13s

kubernetes 192.168.3.1 <none> 443/TCP <none> 1d

redis-master 192.168.3.212 <none> 6379/TCP name=redis-master 1d

redis-slave 192.168.3.233 <none> 6379/TCP name=redis-slave 1d

# kubectl get rc

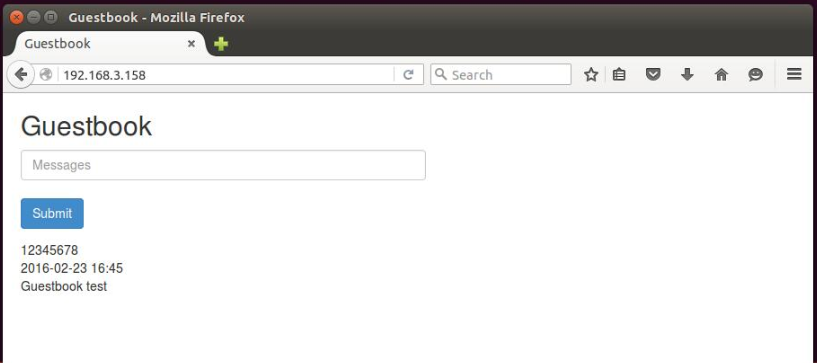
CONTROLLER CONTAINER(S) IMAGE(S) SELECTOR REPLICAS AGE

frontend php-redis gcr.io/google\_samples/gb-frontend:v3 name=frontend 3 1d

redis-master master redis name=redis-master 1 1d

redis-slave worker gcr.io/google\_samples/gb-redisslave:v1 name=redis-slave 2 1d

如果kubernetes是在图形桌面环境下搭建，可以在浏览器输入地址192.168.3.158：80 将会看到下面的界面：



由于搭建环境的各个主机的外网权限限制，所以必需从可以连接外网并且已经下载好镜像文件的主机导出镜像文件，然后在每个node主机导入镜像文件

镜像的导入导出方法以（busybox镜像为例）

导出指令：docker save -o busybox.tar busybox:latest

导入指令：docker load --input busybox.tar

用到的镜像文件列表：

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

busybox latest b175bcb79023 6 weeks ago 1.114 MB

redis latest 8bccd73928d9 8 weeks ago 151.3 MB

gcr.io/google\_containers/kube-ui v4 50405356c714 11 weeks ago 5.677 MB

gcr.io/google\_containers/skydns 2015-10-13-8c72f8c 763c92e53f31 4 months ago 40.55 MB

gcr.io/google\_samples/gb-frontend v3 ef8dcaba5c0f 5 months ago 509.7 MB

gcr.io/google\_containers/kube-ui v2 6286e698abf4 6 months ago 5.673 MB

gcr.io/google\_samples/gb-redisslave v1 f94ba9414e19 6 months ago 109.5 MB

gcr.io/google\_containers/exechealthz 1.0 4f3d04b1d47b 7 months ago 7.099 MB

gcr.io/google\_containers/kube2sky 1.11 e52a547dca17 8 months ago 19.17 MB

gcr.io/google\_containers/etcd 2.0.9 b6b9a86dc06a 11 months ago 12.82 MB

参考文档：

<https://github.com/kubernetes/kubernetes/blob/release-1.1/docs/getting-started-guides/ubuntu.md>

https://github.com/kubernetes/kubernetes/tree/release-1.1/cluster/addons/dns

http://blog.csdn.net/wangtaoking1/article/details/49122977