K8s环境部署实践

目录

**[1.](#_Toc443904118)****[总体说明](#_Toc443904118)** [3](#_Toc443904118)

**[2.](#_Toc443904119)****[安装前环境准备](#_Toc443904119)** [3](#_Toc443904119)

**[3.](#_Toc443904120)****[单板上安装操作系统](#_Toc443904120)** [4](#_Toc443904120)

**[4.](#_Toc443904121)****[虚拟机中安装ubuntu系统](#_Toc443904121)** [6](#_Toc443904121)

[4.1安装virtualbox 6](#_Toc443904122)

[4.2创建虚拟机 6](#_Toc443904123)

[4.3 安装ubuntu系统 8](#_Toc443904124)

**[5.](#_Toc443904125)****[部署Docker容器](#_Toc443904125)** [10](#_Toc443904125)

[5.1 安装Docker 10](#_Toc443904126)

[5.2 安装私有镜像仓库 11](#_Toc443904127)

**[6.](#_Toc443904128)****[部署k8s集群](#_Toc443904128)** [15](#_Toc443904128)

[6.1 部署Master 节点组件 15](#_Toc443904129)

[6.2 部署minion 节点组件 16](#_Toc443904130)

**[7.](#_Toc443904131)****[部署nginx应用](#_Toc443904131)** [17](#_Toc443904131)

[7.1 部署RC和pod 17](#_Toc443904132)

[7.2 部署service 18](#_Toc443904133)

[7.3 访问service 19](#_Toc443904134)

**[8.](#_Toc443904135)****[相关参考资料](#_Toc443904135)** [21](#_Toc443904135)

1. **总体说明**

Kubernetes简称k8s，是google开发的一款开源Docker管理工具，用于跨计算机集群部署容器。本手册基于ubuntu系统下的集群部署，以简单的nginx服务展示了RC（副本复制器）和Service的使用。

本手册中主要包含内容如下：

* 安装前环境准备
* 单板上安装操作系统
* 虚拟机中安装ubuntu系统
* 部署Docker容器
* 部署k8s集群
* 部署nginx应用
* 相关参考资料

1. **安装前环境准备**

环境是在SBCR物理单板IP：10.62.40.129上通过virtualbox创建的2台ubuntu虚拟机，然后安装k8s并进行容器部署。

主要软件版本说明：

|  |  |
| --- | --- |
| 软件 | 版本 |
| Etcd | V2.2.0 |
| VirtualBox | V 5.0.10 |
| Kubernetes | V1.0.1 |
| Docker | V1.6.2 |
| Ubuntu | ubuntu-14.04.1-server-amd64 |

其他工具软件如：

Windows自带命令终端或SecureCRT软件。

FTP上传软件，如FileZilla。

远程图形界面管理工具，如vncviewer。

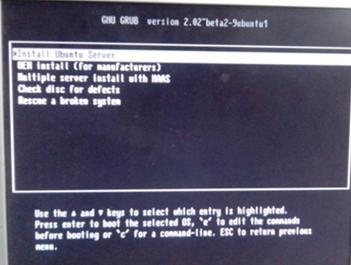
虚拟机环境配置：

其中一个是Master和Minion混合的节点，另一个Minion节点，Minion也称Node节点。

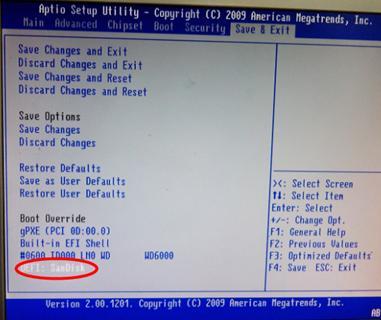
|  |  |  |
| --- | --- | --- |
| 角色 | IP | 组件 |
| Master + Minion | 192.168.56.113 | Etcd apiserver scheduler controller-manager Kubelet kube-proxy |
| Minion | 192.168.56.114 | Kubelet kube-proxy |

1. **单板上安装操作系统**

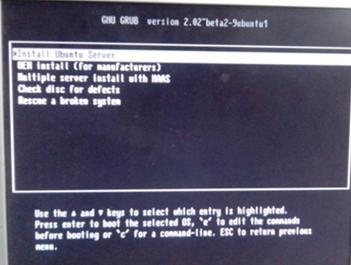
在SBCR单板上安装linux系统，下面以安装ubuntu系统为例，（也可以安装其他linux系统），首先使用ultraiso制作U盘ubuntu系统的启动盘，然后将带有系统安装文件的U盘放到SBCR单板上。重启单板，开机后系统初始化时，不断的点击<delete>键，直到进入BOIS设置

****

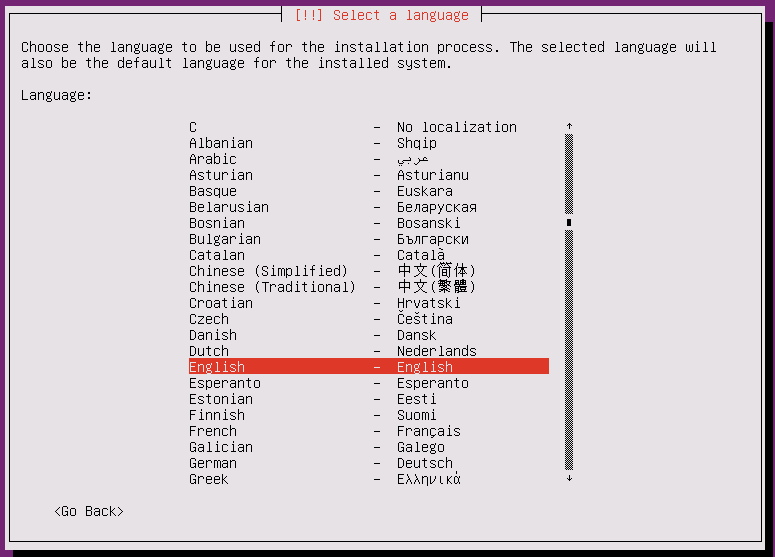
在<Save & Exit>页的<Boot Override>中，选择插入的USB安装盘（不同的U盘，名称不同）



进入ubuntu的界面，选择<Install Ubuntu Server>



选择 <English>



接下来的步骤参考4.3节的《virtualbox下安装ubuntu系统.docx》文档。

1. **虚拟机中安装ubuntu系统**

4.1安装virtualbox

登陆到10.62.40.129单板上：

先备份一下apt原有的sources.list：

cp /etc/apt/sources.list /etc/apt/sources.list.bak

在sources.list中加入下面一行：

deb http://10.62.99.232/virtualbox/debian trusty contrib

然后

apt-get update

最后使用如下命令下载：

apt-get install virtualbox-5.0

验证virtualbox能否正常使用

virtualbox –help

4.2创建虚拟机

下面通过vnc工具，使用图形化方式创建虚拟机，

先在单板上安装vncserver：

root@mcetest:~# apt-get install vnc4server

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following extra packages will be installed:

root@mcetest:~# vncserver :1

You will require a password to access your desktops.

Password:

Verify:

New 'mcetest:1 (root)' desktop is mcetest:1

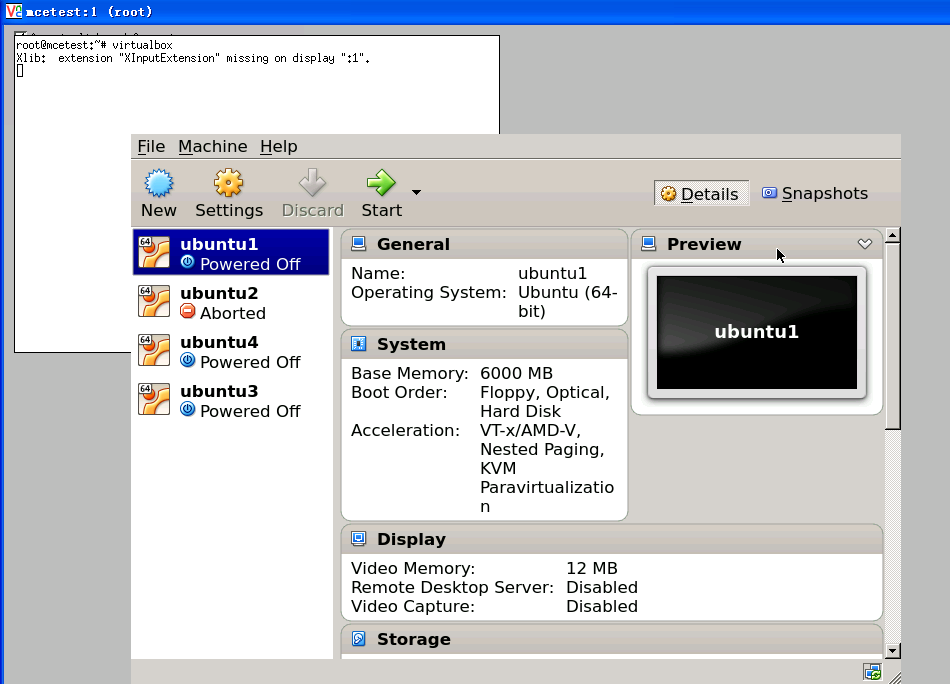
Creating default startup script /home/mcetest/.vnc/xstartup

Starting applications specified in /home/mcetest/.vnc/xstartup

Log file is /home/mcetest/.vnc/mcetest:1.log

使用vncviewer工具登陆，

root@ubuntu:~#virtualbox



4.3 安装ubuntu系统

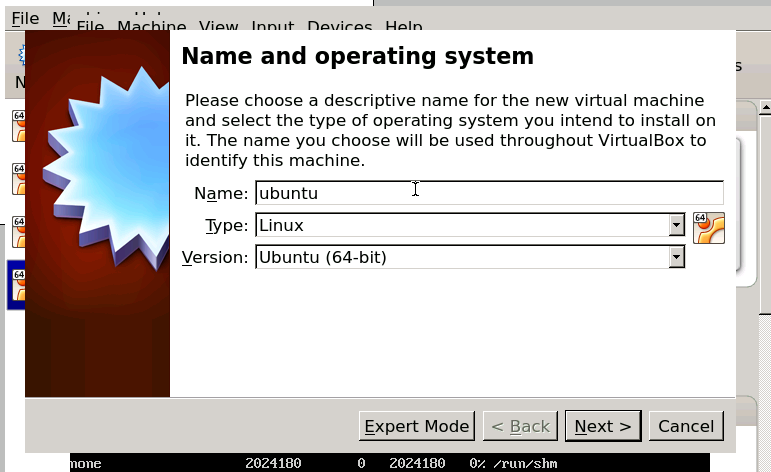
接下来按照以下文档创建虚拟机，生成的ubuntu系统虚拟镜像VDI文件默认存放在/root/VirtualBox VMs目录下。

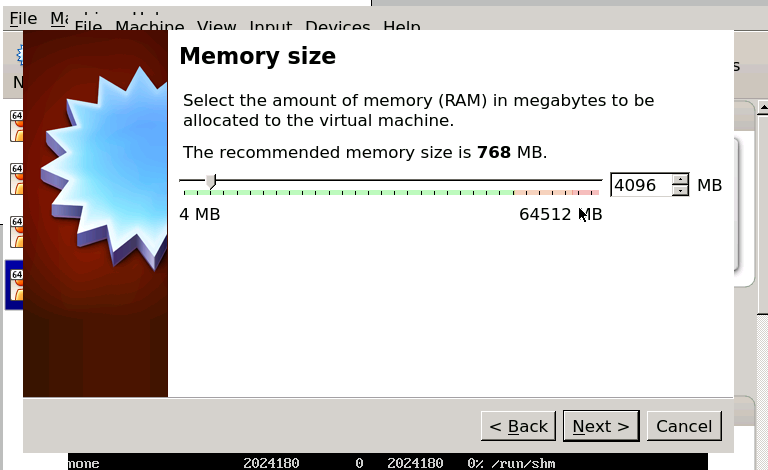


后面根据需要增加虚拟机，先复制已存在的VDI文件，复制的同时可以对新的虚拟硬盘镜像文件重新生成新的uuid。

VBoxManage clonevdi Orig.vdi New.vdi

通过选择已存在新的uuid的VDI文件的方式快速创建虚拟机，完成后虚拟机里面就带有ubuntu系统了。







1. **部署Docker容器**

因为k8s是用来部署和启动docker容器的集群，因此必须保证在master和各minion上都安装了docker。

5.1 安装Docker

分别登陆到master和minion上：

先备份一下apt原有的sources.list：

cp /etc/apt/sources.list /etc/apt/sources.list.bak

编辑/etc/apt/sources.list文件，删除全部内容，然后加入下面内容：

如果仅需要Ubuntu软件包：

deb http://10.62.99.232/common/ubuntu trusty main restricted multiverse universe

deb http://10.62.99.232/common/ubuntu trusty-security main restricted multiverse universe

deb http://10.62.99.232/common/ubuntu trusty-updates main restricted multiverse universe

deb http://10.62.99.232/common/ubuntu trusty-proposed main restricted multiverse universe

deb http://10.62.99.232/common/ubuntu trusty-backports main restricted multiverse universe

安装docker时加入一下信息：

deb http://10.62.99.232/docker/ubuntu docker main

更新一下软件索引列表：

#apt-get update

Docker安装

# sudo apt-get install -y docker.io

# sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker //强行创建软链接

查询是否安装成功

# docker version

5.2 安装私有镜像仓库

在k8s创建启动pod的时候，需要从google的registry中下载一个pause镜像。，其他业务容器需要共享pause容器的网络栈和Volume挂载卷。由于安全原因登陆不了google网站，需要建立自己的私有镜像仓库，将我们需要的镜像放在私有镜像仓库中。

**首先从网上下载registry的docker镜像，并将其导入**

root@ubuntu:/home/zte/share# docker load -i registry.tar

root@ubuntu:/home/zte/share# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

registry latest e94834ac9522 5 months ago 423.3 MB

之后需要修改下docker的配置文件/etc/default/docker，增加下面

# Docker Upstart and SysVinit configuration file

# Customize location of Docker binary (especially for development testing).

#DOCKER="/usr/local/bin/docker"

# Use DOCKER\_OPTS to modify the daemon startup options.

#DOCKER\_OPTS="--dns 8.8.8.8 --dns 8.8.4.4"

DOCKER\_OPTS="--insecure-registry 192.168.56.113:5000"

# If you need Docker to use an HTTP proxy, it can also be specified here.

#export http\_proxy="http://127.0.0.1:3128/"

# This is also a handy place to tweak where Docker's temporary files go.

#export TMPDIR="/mnt/bigdrive/docker-tmp"

**重启docker服务**

service docker restart

**运行私有镜像仓库**

docker run -d -p 5000:5000 registry（不建议使用）

默认情况下，会将仓库存放于容器的/tmp/registry目录下，这样如果容器被删除，则存放于容器中的镜像也会丢失，所以我们一般情况下会指定本地一个目录挂载到容器的/tmp/registry下，如下：

docker run -d -p 5000:5000 -v /opt/data/registry:/tmp/registry registry

这样我们就启动了私有镜像仓库。参数-p 5000:5000表示我们将容器的5000端口映射到宿主机(也就是虚机)的5000端口。换句话说，通过访问虚机的5000端口，就可以访问到私有镜像仓库了。

root@ubuntu:/home/zte/share# curl http://127.0.0.1:5000

"\"docker-registry server\""

接下来，需要将我们用到的镜像先传到私有镜像仓库中。其他项目组已经从网上下载好了镜像放到到本地服务器，可以在10.62.97.238:5000下载pause和aerospike-server镜像，10.62.97.231:500下载nginx镜像。（备注：需要先将/etc/default/docker中DOCKER\_OPTS="--insecure-registry 192.168.56.113:5000" 改成相应的IP，然后使用docker pull xxxx/nginx）下载镜像）。

假设已经本地下载了pause和aerospike-server镜像

# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

registry latest e94834ac9522 5 months ago 423.3 MB

10.62.97.238:5000/aerospike-server latest b6e4942e88ab 5 months ago 147.6 MB

10.62.97.238:5000/kubernetes/pause latest 6c4579af347b 18 months ago 239.8 kB

**先给这两个镜像贴上标签**

root@ubuntu:/etc/init# docker tag b6e4 192.168.56.113:5000/aerospike-server

root@ubuntu:/etc/init# docker tag 6c45 192.168.56.113:5000/kubernetes/pause

上面的6c45和b6e4分别是pause和aerospike-server这两个镜像的ID的前4位。

root@ubuntu:/etc/init# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

registry latest e94834ac9522 5 months ago 423.3 MB

10.62.97.238:5000/aerospike-server latest b6e4942e88ab 5 months ago 147.6 MB

192.168.56.113:5000/aerospike-server latest b6e4942e88ab 5 months ago 147.6 MB

10.62.97.238:5000/kubernetes/pause latest 6c4579af347b 19 months ago 239.8 kB

192.168.56.113:5000/kubernetes/pause latest 6c4579af347b 19 months ago 239.8 kB

192.168.56.113:5000/kubernetes/pause和192.168.56.113:5000/aerospike-server这两个新产生的镜像和原来的pause和aerospike-server的IMAGE ID是一样的，是同一个镜像只是标签不一样而已。

**然后开始上传镜像**

root@ubuntu:/etc/init# docker push 192.168.56.113:5000/aerospike-server

The push refers to a repository [192.168.56.113:5000/aerospike-server] (len: 1)

Sending image list

Pushing repository 192.168.56.113:5000/aerospike-server (1 tags)

9d3ceacde91b: Image successfully pushed

bbe78c1a5a53: Image successfully pushed

ce5dafd76f08: Image successfully pushed

b1d7f0a60acf: Image successfully pushed

e27000943065: Image successfully pushed

0cfe483ee2dc: Image successfully pushed

7ccd36d7a74f: Image successfully pushed

8525bca90b6c: Image successfully pushed

b6e4942e88ab: Image successfully pushed

Pushing tag for rev [b6e4942e88ab] on {http://192.168.56.113:5000/v1/repositories/aerospike-server/tags/latest}

root@ubuntu:/etc/init# docker push 192.168.56.113:5000/kubernetes/pause

The push refers to a repository [192.168.56.113:5000/kubernetes/pause] (len: 1)

Sending image list

Pushing repository 192.168.56.113:5000/kubernetes/pause (1 tags)

511136ea3c5a: Image successfully pushed

e244e638e26e: Image successfully pushed

6c4579af347b: Image successfully pushed

Pushing tag for rev [6c4579af347b] on {http://192.168.56.113:5000/v1/repositories/kubernetes/pause/tags/latest}

**确认私有仓库是否有刚上传的镜像。**

root@ubuntu:/etc/init# curl http://127.0.0.1:5000/v1/search

{"num\_results": 2, "query": "", "results": [{"description": "", "name": "library/aerospike-server"}, {"description": "", "name": "kubernetes/pause"}]}root@ubuntu:/etc/init#

1. **部署k8s集群**

下面采用命令的方式进行手动部署。

6.1 部署Master 节点组件

192.168.56.113虚机

解压kubernetes和etcd安装文件，将二进制可执行文件放入/opt/kubernetes/bin，方便统一执行和管理。

root@ubuntu:/home/zte/share # mkdir -p /opt/kubernetes/bin

root@ubuntu:/home/zte/share #tar xf kubernetes.tar.gz

root@ubuntu:/home/zte/share/kubernetes/server# tar xf kubernetes-server-linux-amd64.tar.gz

root@ubuntu:/home/zte/share/kubernetes/server/kubernetes/server/bin# cp kube-apiserver kube-controller-manager kube-scheduler kubelet kube-proxy /opt/kubernetes/bin/

root@ubuntu:/home/zte/share# tar xf etcd-v2.2.0-linux-amd64.tar.gz

root@ubuntu:/home/zte/share/etcd-v2.2.0-linux-amd64# cp etcd etcdctl /opt/kubernetes/bin/

**拷贝kubelet和kube-proxy文件到192.168.56.114虚拟机minion**

root@ubuntu:/home/zte/share/kubernetes/server/kubernetes/server/bin# scp kubelet kube-proxy root@192.168.56.114:/opt/kubernetes/bin

**启动etcd-server和k8s的master组件apiserver，controller-manager，scheduler。**

root@ubuntu:/opt/kubernetes/bin#./etcd --initial-advertise-peer-urls http://192.168.56.113:7001 --advertise-client-urls http://192.168.56.113:4001 --listen-peer-urls http://0.0.0.0:7001 --listen-client-urls http://0.0.0.0:4001 --initial-cluster default=http://192.168.56.2:7001

root@ubuntu:/opt/kubernetes/bin#./kube-apiserver --etcd-servers=http://192.168.56.113:4001 --service-cluster-ip-range=192.168.56.0/24 --address=0.0.0.0

root@ubuntu:/opt/kubernetes/bin#./kube-controller-manager --master=192.168.56.113:8080

root@ubuntu:/opt/kubernetes/bin#./kube-scheduler --master=192.168.56.113:8080

6.2 部署minion 节点组件

192.168.56.113虚拟机

启动k8s的minion组件kubelet，kube-proxy

root@ubuntu:/opt/kubernetes/bin#./kubelet --api-servers=192.168.56.113:8080 --pod-infra-container-image=192.168.56.113:5000/kubernetes/pause --hostname-override="192.168.56.113"

root@ubuntu:/opt/kubernetes/bin#./kube-proxy --master=192.168.56.113:8080

192.168.56.114虚拟机

启动k8s的minion组件kubelet，kube-proxy

root@ubuntu:/opt/kubernetes/bin#./kubelet --api-servers=192.168.56.113:8080 --pod-infra-container-image=192.168.56.113:5000/kubernetes/pause --hostname-override="192.168.56.114"

root@ubuntu:/opt/kubernetes/bin#./kube-proxy --master=192.168.56.113:8080

**在master节点(192.168.56.113)，查看k8s集群节点**

root@ubuntu:~# kubectl get nodes

NAME LABELS STATUS

192.168.56.113 kubernetes.io/hostname=192.168.56.113 Ready

192.168.56.114 kubernetes.io/hostname=192.168.56.114 Ready

表明集群构建成功。

1. **部署nginx应用**

接下来进行nginx应用的部署。先要下载好了nginx镜像到本地。Pod可以单独创建，也可以通过RC来创建，根据官方推荐使用RC创建pods。

* 1. 部署RC和pod

通过命令工具进行快速部署RC。

kubectl run nginx --image=192.168.56.113:5000/library/nginx --replicas=2

或者通过yaml来部署RC

root@ubuntu:/home/zte/share# cat nginx-rc.yaml

apiVersion: v1

kind: ReplicationController

metadata:

name: nginx-controller

spec:

replicas: 2

selector:

name: nginx

template:

metadata:

labels:

name: nginx

spec:

containers:

- name: nginx

image: 192.168.56.113:5000/library/nginx

ports:

- containerPort: 80

定义了一个nginx pod复制器，复制份数为2，使用nginx 的docker镜像

root@ubuntu:/home/zte/share# kubectl create -f nginx-rc.yaml

**查看rc**

root@ubuntu:/home/zte/share# kubectl get rc

CONTROLLER CONTAINER(S) IMAGE(S) SELECTOR REPLICAS

nginx-controller nginx 192.168.56.113:5000/library/nginx name=nginx 2

**查看pods**

root@ubuntu:/home/zte/share# kubectl get pods

NAME READY STATUS RESTARTS AGE

nginx-controller-gf10m 1/1 Running 0 18h

nginx-controller-xborx 1/1 Running 0 18h

7.2 部署service

通过yaml来部署service

root@ubuntu:/home/zte/share# cat nginx-service.yaml

apiVersion: v1

kind: Service

metadata:

name: nginx-service-nodeport

spec:

ports:

- port: 8000

targetPort: 80

nodeport: 30001

protocol: TCP

type: NodePort

selector:

name: nginx

root@ubuntu:/home/zte/share# kubectl create -f nginx-service.yaml

K8s支持两种对外提供服务的service的type定义，Nodeport和LoadBalancer，这里使用到的是Nodeport方式，Nodeport默认范围是30000—32767。

**查看service**

root@ubuntu:/home/zte/share# kubectl get se

NAME LABELS SELECTOR IP(S) PORT(S)

kubernetes component=apiserver,provider=kubernetes <none> 192.168.56.1 443/TCP

nginx-service-nodeport <none> name=nginx 192.168.56.122 8000/TCP

7.3 访问service

**通过Node IP 和Nodeport方式访问service**

nodeIP:nodePort 是提供给集群外部客户访问service的入口，是kubernetes提供给集群外部客户访问service入口的一种方式（另一种方式是LoadBalancer）

root@ubuntu:/home/zte/share# curl –s 192.168.56.114:30001

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

body {

width: 35em;

margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

**通过cluster IP 和port方式访问service**

cluster ip:port 是提供给集群内部客户访问service的入口。

root@ubuntu:~# ssh 192.168.56.114 curl -s 192.168.56.122:8000

root@192.168.56.114's password:

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

body {

width: 35em;

margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

经过上面的部署，已经可以成功访问到service，至此已经完成k8s容器nginx应用的部署。

1. **相关参考资料**

<http://w23ta0.sinaapp.com/948.html> Docker搭建本地私有仓库

<http://www.csdn.net/article/2015-06-12/2824937> Kubernetes应用部署模型解析