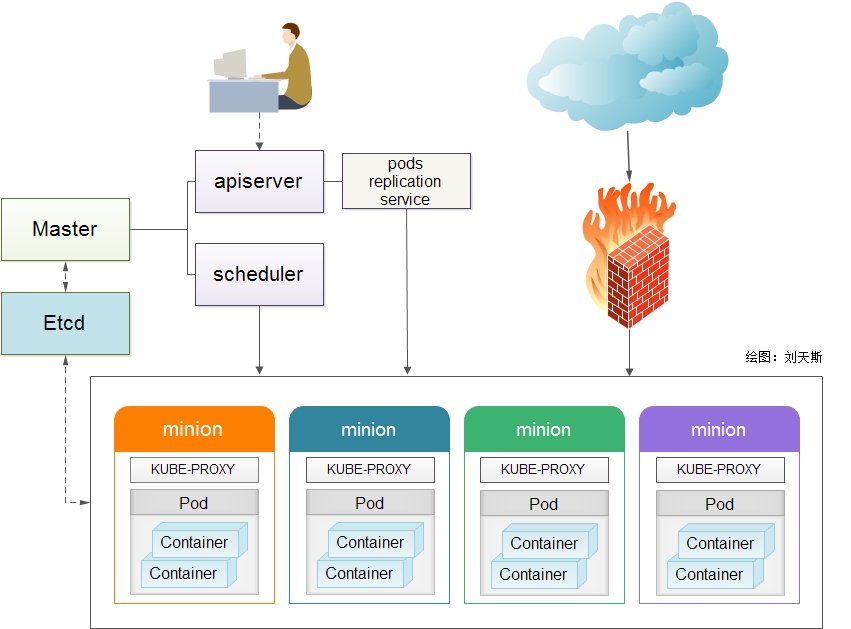
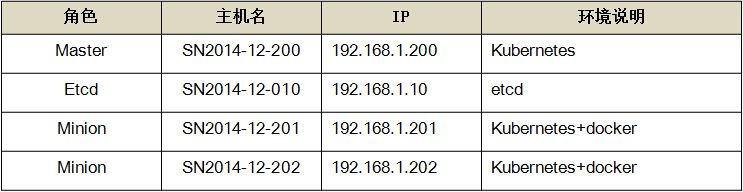
**一、前言**  
        Kubernetes 是Google开源的容器集群管理系统，基于Docker构建一个容器的调度服务，提供资源调度、均衡容灾、服务注册、动态扩缩容等功能套件，目前最新版本为0.6.2。本文介绍如何基于Centos7.0构建Kubernetes平台，在正式介绍之前，大家有必要先理解Kubernetes几个核心概念及其承担的功能。以下为Kubernetes的架构设计图：  
[](http://blog.liuts.com/attachment.php?fid=368)  
1. Pods  
        在Kubernetes系统中，调度的最小颗粒不是单纯的容器，而是抽象成一个Pod，Pod是一个可以被创建、销毁、调度、管理的最小的部署单元。比如一个或一组容器。  
2. Replication Controllers  
        Replication Controller是Kubernetes系统中最有用的功能，实现复制多个Pod副本，往往一个应用需要多个Pod来支撑，并且可以保证其复制的副本数，即使副本所调度分配的主宿机出现异常，通过Replication Controller可以保证在其它主宿机启用同等数量的Pod。Replication Controller可以通过repcon模板来创建多个Pod副本，同样也可以直接复制已存在Pod，需要通过Label selector来关联。  
3、Services  
        Services是Kubernetes最外围的单元，通过虚拟一个访问IP及服务端口，可以访问我们定义好的Pod资源，目前的版本是通过iptables的nat转发来实现，转发的目标端口为Kube\_proxy生成的随机端口，目前只提供GOOGLE云上的访问调度，如GCE。如果与我们自建的平台进行整合？请关注下篇《kubernetes与HECD架构的整合》文章。  
4、Labels  
        Labels是用于区分Pod、Service、Replication Controller的key/value键值对，仅使用在Pod、Service、 Replication Controller之间的关系识别，但对这些单元本身进行操作时得使用name标签。  
5、Proxy  
        Proxy不但解决了同一主宿机相同服务端口冲突的问题，还提供了Service转发服务端口对外提供服务的能力，Proxy后端使用了随机、轮循负载均衡算法。  
  
        说说个人一点看法，目前Kubernetes 保持一周一小版本、一个月一大版本的节奏，迭代速度极快，同时也带来了不同版本操作方法的差异，另外官网文档更新速度相对滞后及欠缺，给初学者带来一定挑战。在上游接入层官方侧重点还放在GCE（Google Compute Engine）的对接优化，针对个人私有云还未推出一套可行的接入解决方案。在v0.5版本中才引用service代理转发的机制，且是通过iptables来实现，在高并发下性能令人担忧。但作者依然看好Kubernetes未来的发展，至少目前还未看到另外一个成体系、具备良好生态圈的平台，相信在V1.0时就会具备生产环境的服务支撑能力。  
  
**一、环境部署**  
1、平台版本说明  
    1）Centos7.0 OS  
    2）Kubernetes V0.6.2  
    3）etcd version 0.4.6  
    4）Docker version 1.3.2  
  
2、平台环境说明  
[](http://blog.liuts.com/attachment.php?fid=369)  
3、环境安装  
    1）系统初始化工作（所有主机）  
    系统安装-选择[最小化安装]

**引用**

    # yum -y install wget ntpdate bind-utils  
    # wget http://mirror.centos.org/centos/7/extras/x86\_64/Packages/epel-release-7-2.noarch.rpm  
    # yum update

    CentOS 7.0默认使用的是firewall作为防火墙，这里改为iptables防火墙（熟悉度更高，非必须）。  
    1.1、关闭firewall：

**引用**

    # systemctl stop firewalld.service #停止firewall  
    # systemctl disable firewalld.service #禁止firewall开机启动

    1.2、安装iptables防火墙

**引用**

    # yum install iptables-services #安装  
    # systemctl start iptables.service #最后重启防火墙使配置生效  
    # systemctl enable iptables.service #设置防火墙开机启动

    2）安装Etcd（192.168.1.10主机）

**引用**

    # mkdir -p /home/install && cd /home/install    
    # wget https://github.com/coreos/etcd/releases/download/v0.4.6/etcd-v0.4.6-linux-amd64.tar.gz    
    # tar -zxvf etcd-v0.4.6-linux-amd64.tar.gz    
    # cd etcd-v0.4.6-linux-amd64    
    # cp etcd\* /bin/    
    # /bin/etcd -version    
    etcd version 0.4.6  

    启动服务etcd服务，如有提供第三方管理需求，另需在启动参数中添加“-cors='\*'”参数。

**引用**

    # mkdir /data/etcd    
    # /bin/etcd -name etcdserver -peer-addr 192.168.1.10:7001 -addr 192.168.1.10:4001 -data-dir /data/etcd -peer-bind-addr 0.0.0.0:7001 -bind-addr 0.0.0.0:4001 &

    配置etcd服务防火墙，其中4001为服务端口，7001为集群数据交互端口。

**引用**

    # iptables -I INPUT -s 192.168.1.0/24 -p tcp --dport 4001 -j ACCEPT  
    # iptables -I INPUT -s 192.168.1.0/24 -p tcp --dport 7001 -j ACCEPT  
    
    3）安装Kubernetes（涉及所有Master、Minion主机）  
    通过yum源方式安装，默认将安装etcd, docker, and cadvisor相关包。

**引用**

    # curl https://copr.fedoraproject.org/coprs/eparis/kubernetes-epel-7/repo/epel-7/eparis-kubernetes-epel-7-epel-7.repo -o /etc/yum.repos.d/eparis-kubernetes-epel-7-epel-7.repo  
    #yum -y install kubernetes

    升级至v0.6.2，覆盖bin文件即可，方法如下：

**引用**

    # mkdir -p /home/install && cd /home/install  
    # wget https://github.com/GoogleCloudPlatform/kubernetes/releases/download/v0.6.2/kubernetes.tar.gz  
    # tar -zxvf kubernetes.tar.gz  
    # tar -zxvf kubernetes/server/kubernetes-server-linux-amd64.tar.gz  
    # cp kubernetes/server/bin/kube\* /usr/bin

    校验安装结果，出版以下信息说明安装正常。

**引用**

    [root@SN2014-12-200 bin]# /usr/bin/kubectl version  
    Client Version: version.Info{Major:"0", Minor:"6+", GitVersion:"v0.6.2", GitCommit:"729fde276613eedcd99ecf5b93f095b8deb64eb4", GitTreeState:"clean"}  
    Server Version: &version.Info{Major:"0", Minor:"6+", GitVersion:"v0.6.2", GitCommit:"729fde276613eedcd99ecf5b93f095b8deb64eb4", GitTreeState:"clean"}

    4）Kubernetes配置（仅Master主机）  
    master运行三个组件,包括apiserver、scheduler、controller-manager，相关配置项也只涉及这三块。  
4.1、【/etc/kubernetes/config】

[view plainprint?](http://blog.liuts.com/)

1. **# Comma seperated list of nodes in the etcd cluster**
2. KUBE\_ETCD\_SERVERS="--etcd\_servers=http://192.168.1.10:4001"
4. **# logging to stderr means we get it in the systemd journal**
5. KUBE\_LOGTOSTDERR="--logtostderr=true"
7. **# journal message level, 0 is debug**
8. KUBE\_LOG\_LEVEL="--v=0"
10. **# Should this cluster be allowed to run privleged docker containers**
11. KUBE\_ALLOW\_PRIV="--allow\_privileged=false"

4.2、【/etc/kubernetes/apiserver】

[view plainprint?](http://blog.liuts.com/)

1. **# The address on the local server to listen to.**
2. KUBE\_API\_ADDRESS="--address=0.0.0.0"
4. **# The port on the local server to listen on.**
5. KUBE\_API\_PORT="--port=8080"
7. **# How the replication controller and scheduler find the kube-apiserver**
8. KUBE\_MASTER="--master=192.168.1.200:8080"
10. **# Port minions listen on**
11. KUBELET\_PORT="--kubelet\_port=10250"
13. **# Address range to use for services**
14. KUBE\_SERVICE\_ADDRESSES="--portal\_net=10.254.0.0/16"
16. **# Add you own!**
17. KUBE\_API\_ARGS=""

4.3、【/etc/kubernetes/controller-manager】

[view plainprint?](http://blog.liuts.com/)

1. **# Comma seperated list of minions**
2. KUBELET\_ADDRESSES="--machines= 192.168.1.201,192.168.1.202"
4. **# Add you own!**
5. KUBE\_CONTROLLER\_MANAGER\_ARGS=""

4.4、【/etc/kubernetes/scheduler】

[view plainprint?](http://blog.liuts.com/)

1. **# Add your own!**
2. KUBE\_SCHEDULER\_ARGS=""

    启动master侧相关服务

**引用**

    # systemctl daemon-reload  
    # systemctl start kube-apiserver.service kube-controller-manager.service kube-scheduler.service  
    # systemctl enable kube-apiserver.service kube-controller-manager.service kube-scheduler.service

    5）Kubernetes配置（仅minion主机）  
        minion运行两个组件,包括kubelet、proxy，相关配置项也只涉及这两块。  
    Docker启动脚本更新  
    # vi /etc/sysconfig/docker  
    添加：-H tcp://0.0.0.0:2375，最终配置如下，以便以后提供远程API维护。  
    OPTIONS=--selinux-enabled -H tcp://0.0.0.0:2375 -H fd://  
  
    修改minion防火墙配置，通常master找不到minion主机多半是由于端口没有连通。  
    iptables -I INPUT -s 192.168.1.200 -p tcp --dport 10250 -j ACCEPT  
  
    修改kubernetes minion端配置，以192.168.1.201主机为例，其它minion主机同理。  
5.1、【/etc/kubernetes/config】

[view plainprint?](http://blog.liuts.com/)

1. **# Comma seperated list of nodes in the etcd cluster**
2. KUBE\_ETCD\_SERVERS="--etcd\_servers=http://192.168.1.10:4001"
4. **# logging to stderr means we get it in the systemd journal**
5. KUBE\_LOGTOSTDERR="--logtostderr=true"
7. **# journal message level, 0 is debug**
8. KUBE\_LOG\_LEVEL="--v=0"
10. **# Should this cluster be allowed to run privleged docker containers**
11. KUBE\_ALLOW\_PRIV="--allow\_privileged=false"

5.2、【/etc/kubernetes/kubelet】

[view plainprint?](http://blog.liuts.com/)

1. **###**
2. **# kubernetes kubelet (minion) config**
4. **# The address for the info server to serve on (set to 0.0.0.0 or "" for all interfaces)**
5. KUBELET\_ADDRESS="--address=0.0.0.0"
7. **# The port for the info server to serve on**
8. KUBELET\_PORT="--port=10250"
10. **# You may leave this blank to use the actual hostname**
11. KUBELET\_HOSTNAME="--hostname\_override=192.168.1.201"
13. **# Add your own!**
14. KUBELET\_ARGS=""

5.3、【/etc/kubernetes/proxy】

[view plainprint?](http://blog.liuts.com/)

1. KUBE\_PROXY\_ARGS=""

启动kubernetes服务

**引用**

# systemctl daemon-reload  
# systemctl enable docker.service kubelet.service kube-proxy.service  
# systemctl start docker.service kubelet.service kube-proxy.service

3、校验安装(在master主机操作，或可访问master主机8080端口的client api主机)  
  1) kubernetes常用命令

**引用**

# kubectl get minions    #查查看minion主机  
# kubectl get pods    #查看pods清单  
# kubectl get services 或 kubectl get services -o json    #查看service清单  
# kubectl get replicationControllers    #查看replicationControllers清单  
# for i in `kubectl get pod|tail -n +2|awk '{print $1}'`; do kubectl delete pod $i; done    #删除所有pods

    或者通过Server api for REST方式（推荐，及时性更高）：

**引用**

# curl -s -L http://192.168.1.200:8080/api/v1beta1/version | python -mjson.tool    #查看kubernetes版本  
# curl -s -L http://192.168.1.200:8080/api/v1beta1/pods | python -mjson.tool    #查看pods清单  
# curl -s -L http://192.168.1.200:8080/api/v1beta1/replicationControllers | python -mjson.tool    #查看replicationControllers清单  
# curl -s -L http://192.168.1.200:8080/api/v1beta1/minions | python -m json.tool    #查查看minion主机  
# curl -s -L http://192.168.1.200:8080/api/v1beta1/services | python -m json.tool    #查看service清单

注：在新版kubernetes中，所有的操作命令都整合至kubectl，包括kubecfg、kubectl.sh、kubecfg.sh等  
  
  2）创建测试pod单元  
   # /home/kubermange/pods && cd /home/kubermange/pods  
   # vi apache-pod.json

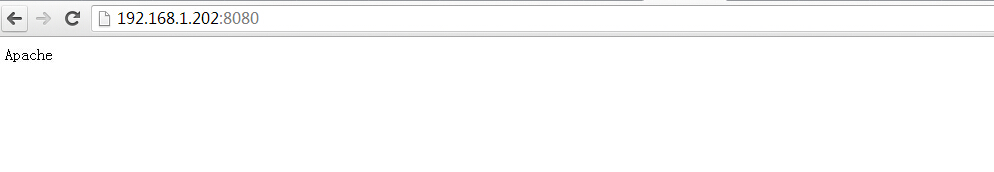
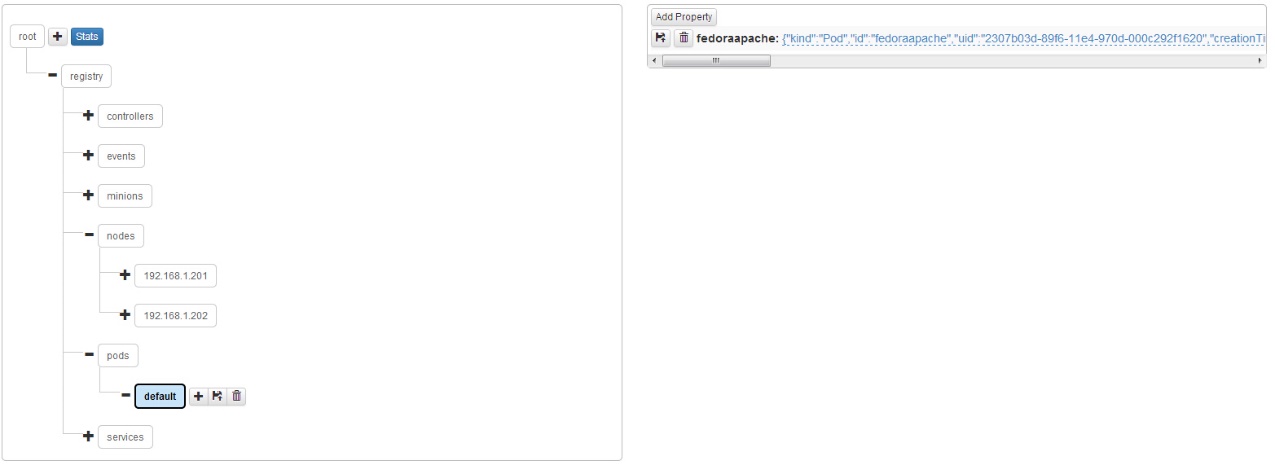
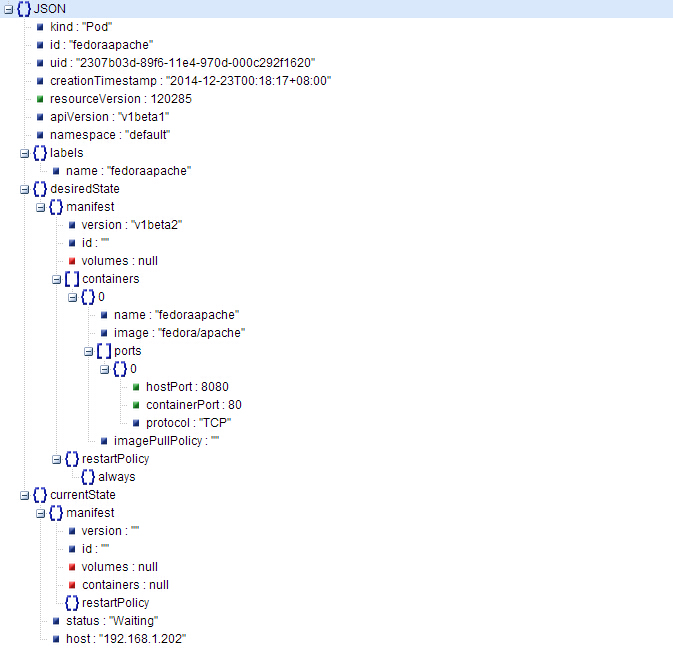
[view plainprint?](http://blog.liuts.com/)

1. {
2. "id": "fedoraapache",
3. "kind": "Pod",
4. "apiVersion": "v1beta1",
5. "desiredState": {
6. "manifest": {
7. "version": "v1beta1",
8. "id": "fedoraapache",
9. "containers": [{
10. "name": "fedoraapache",
11. "image": "fedora/apache",
12. "ports": [{
13. "containerPort": 80,
14. "hostPort": 8080
15. }]
16. }]
17. }
18. },
19. "labels": {
20. "name": "fedoraapache"
21. }
22. }

    # kubectl create -f apache-pod.json  
    # kubectl get pod

**引用**

NAME                IMAGE(S)            HOST                LABELS              STATUS  
fedoraapache        fedora/apache       192.168.1.202/      name=fedoraapache   Running

    启动浏览器访问http://192.168.1.202:8080/，对应的服务端口切记在iptables中已添加。效果图如下：  
[](http://blog.liuts.com/attachment.php?fid=370)  
    观察kubernetes在etcd中的数据存储结构  
[](http://blog.liuts.com/attachment.php?fid=373)  
  
    观察单个pods的数据存储结构，以json的格式存储。  
[](http://blog.liuts.com/attachment.php?fid=374)  
  
**二、实战操作**  
    任务：通过Kubernetes创建一个LNMP架构的服务集群，以及观察其负载均衡，涉及镜像“yorko/webserver”已经push至registry.hub.docker.com，大家可以通过“docker pull yorko/webserver”下载。

**引用**

    # mkdir -p /home/kubermange/replication && mkdir -p /home/kubermange/service  
    # cd /home/kubermange/replication 

1、 创建一个replication ，本例直接在replication模板中创建pod并复制，也可独立创建pod再通过replication来复制。  
【replication/lnmp-replication.json】

[view plainprint?](http://blog.liuts.com/)

1. {
2. "id": "webserverController",
3. "kind": "ReplicationController",
4. "apiVersion": "v1beta1",
5. "labels": {"name": "webserver"},
6. "desiredState": {
7. "replicas": 2,
8. "replicaSelector": {"name": "webserver\_pod"},
9. "podTemplate": {
10. "desiredState": {
11. "manifest": {
12. "version": "v1beta1",
13. "id": "webserver",
14. "volumes": [
15. {"name":"httpconf", "source":{"hostDir":{"path":"/etc/httpd/conf"}}},
16. {"name":"httpconfd", "source":{"hostDir":{"path":"/etc/httpd/conf.d"}}},
17. {"name":"httproot", "source":{"hostDir":{"path":"/data"}}}
18. ],
19. "containers": [{
20. "name": "webserver",
21. "image": "yorko/webserver",
22. "command": ["/bin/sh", "-c", "/usr/bin/supervisord -c /etc/supervisord.conf"],
23. "volumeMounts": [
24. {"name":"httpconf", "mountPath":"/etc/httpd/conf"},
25. {"name":"httpconfd", "mountPath":"/etc/httpd/conf.d"},
26. {"name":"httproot", "mountPath":"/data"}
27. ],
28. "cpu": 100,
29. "memory": 50000000,
30. "ports": [{
31. "containerPort": 80,
32. },{
33. "containerPort": 22,
34. }]
35. }]
36. }
37. },
38. "labels": {"name": "webserver\_pod"},
39. },
40. }
41. }

    执行创建命令  
    #kubectl create -f lnmp-replication.json  
    观察生成的pod副本清单：  
[root@SN2014-12-200 replication]# kubectl get pod

**引用**

NAME                                   IMAGE(S)            HOST                LABELS               STATUS  
84150ab7-89f8-11e4-970d-000c292f1620   yorko/webserver     192.168.1.202/      name=webserver\_pod   Running  
84154ed5-89f8-11e4-970d-000c292f1620   yorko/webserver     192.168.1.201/      name=webserver\_pod   Running  
840beb1b-89f8-11e4-970d-000c292f1620   yorko/webserver     192.168.1.202/      name=webserver\_pod   Running  
84152d93-89f8-11e4-970d-000c292f1620   yorko/webserver     192.168.1.202/      name=webserver\_pod   Running  
840db120-89f8-11e4-970d-000c292f1620   yorko/webserver     192.168.1.201/      name=webserver\_pod   Running  
8413b4f3-89f8-11e4-970d-000c292f1620   yorko/webserver     192.168.1.201/      name=webserver\_pod   Running

2、创建一个service，通过selector指定 "name": "webserver\_pod"与pods关联。  
【service/lnmp-service.json】

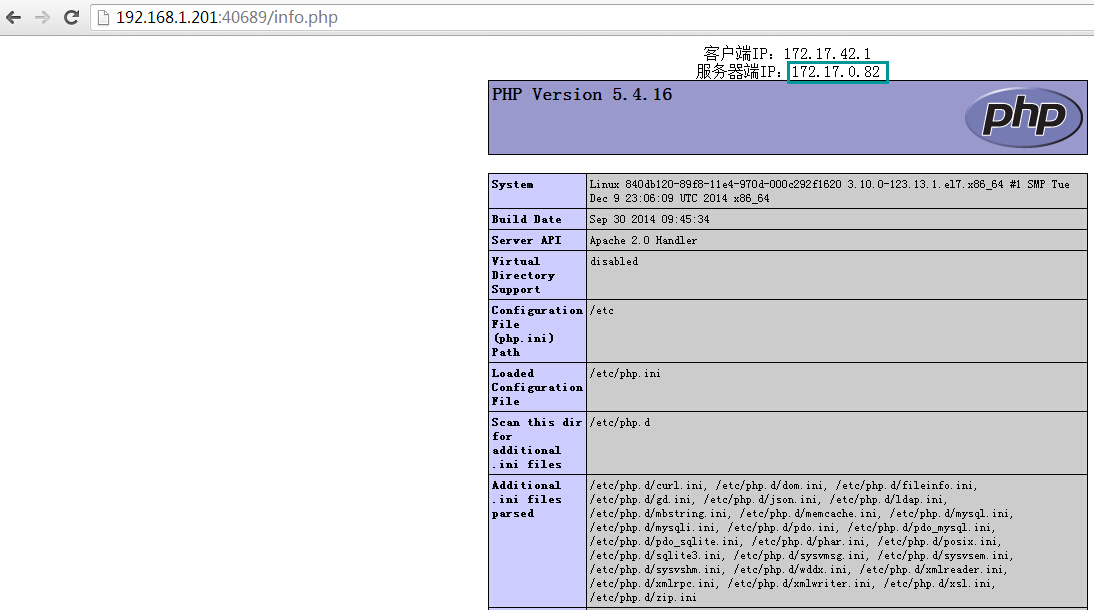
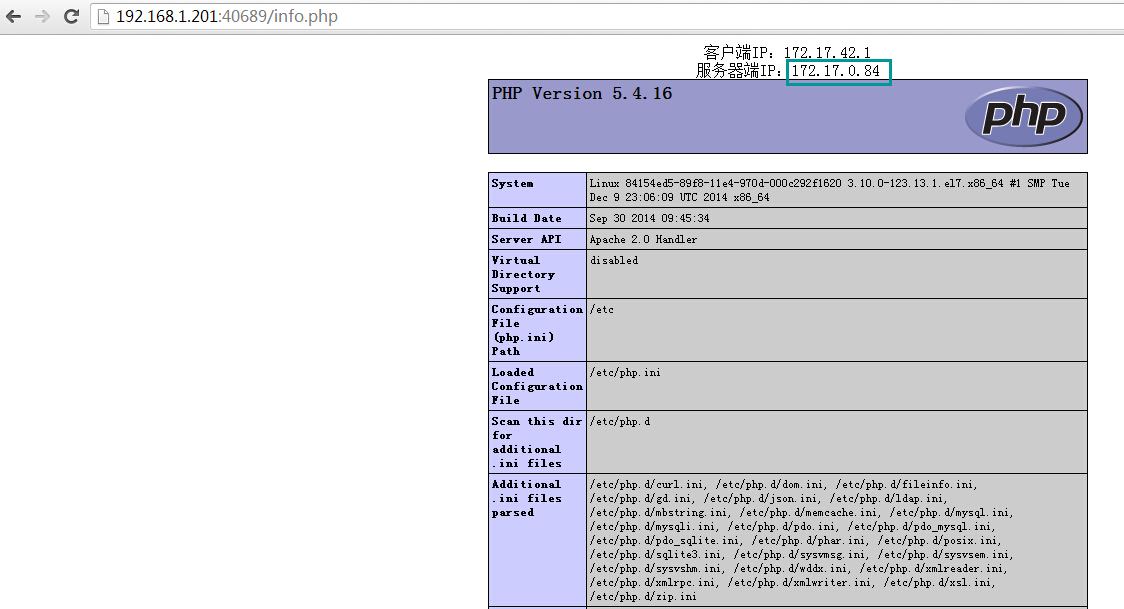
[view plainprint?](http://blog.liuts.com/)

1. {
2. "id": "webserver",
3. "kind": "Service",
4. "apiVersion": "v1beta1",
5. "selector": {
6. "name": "webserver\_pod",
7. },
8. "protocol": "TCP",
9. "containerPort": 80,
10. "port": 8080
11. }

    执行创建命令：  
    # kubectl create -f lnmp-service.json  
  
    登录minion主机（192.168.1.201），查询主宿机生成的iptables转发规则（最后一行）  
    # iptables -nvL -t nat

**引用**

Chain KUBE-PROXY (2 references)  
pkts bytes target     prot opt in     out     source               destination           
    2   120 REDIRECT   tcp  --  \*      \*       0.0.0.0/0            10.254.102.162       /\* kubernetes \*/ tcp dpt:443 redir ports 47700  
    1    60 REDIRECT   tcp  --  \*      \*       0.0.0.0/0            10.254.28.74         /\* kubernetes-ro \*/ tcp dpt:80 redir ports 60099  
    0     0 REDIRECT   tcp  --  \*      \*       0.0.0.0/0            10.254.216.51        /\* webserver \*/ tcp dpt:8080 redir ports 40689

    访问测试，http://192.168.1.201:40689/info.php，刷新浏览器发现proxy后端的变化，默认为随机轮循算法。  
[](http://blog.liuts.com/attachment.php?fid=371)  
[](http://blog.liuts.com/attachment.php?fid=372)  
  
**三、测试过程**  
    1、pods自动复制、销毁测试，观察kubernetes自动保持副本数（6份）  
删除replicationcontrollers中一个副本fedoraapache  
[root@SN2014-12-200 pods]# kubectl delete pods fedoraapache  
I1219 23:59:39.305730    9516 restclient.go:133] Waiting for completion of operation 142530  
fedoraapache

**引用**

[root@SN2014-12-200 pods]# kubectl get pods  
NAME                                   IMAGE(S)            HOST                LABELS              STATUS  
5d70892e-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.201/      name=fedoraapache   Running  
5d715e56-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.202/      name=fedoraapache   Running  
5d717f8d-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.202/      name=fedoraapache   Running  
5d71c584-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.201/      name=fedoraapache   Running  
5d71a494-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.202/      name=fedoraapache   Running

#自动生成出一个副本，保持6份的效果

**引用**

[root@SN2014-12-200 pods]# kubectl get pods  
NAME                                   IMAGE(S)            HOST                LABELS              STATUS  
5d717f8d-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.202/      name=fedoraapache   Running  
5d71c584-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.201/      name=fedoraapache   Running  
5d71a494-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.202/      name=fedoraapache   Running  
2a8fb993-8798-11e4-970d-000c292f1620   fedora/apache       192.168.1.201/      name=fedoraapache   Running  
5d70892e-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.201/      name=fedoraapache   Running  
5d715e56-8794-11e4-970d-000c292f1620   fedora/apache       192.168.1.202/      name=fedoraapache   Running

2、测试不同角色模块中的hostPort  
    1）pod中hostPort为空，而replicationcontrollers为指定端口，则异常；两侧都指定端口，相同或不同时都异常；pod的hostport为指定，另replicationcon为空，则正常；pod的hostport为空，另replicationcon为空，则正常；结论是在replicationcontrollers场景不能指定hostport，否则异常，待持续测试。  
    2）结论：在replicationcontronllers.json中，"replicaSelector": {"name": "webserver\_pod"}要与"labels": {"name": "webserver\_pod"}以及service中的"selector": {"name": "webserver\_pod"｝保持一致；  
  
请关注下篇《kubernetes与HECD架构的整合》，近期推出。  
  
参考文献：  
https://github.com/GoogleCloudPlatform/kubernetes/blob/master/docs/getting-started-guides/fedora/fedora\_manual\_config.md  
https://github.com/GoogleCloudPlatform/kubernetes/blob/master/DESIGN.md  
http://www.infoq.com/cn/articles/Kubernetes-system-architecture-introduction  
  
转载请注明来源 http://blog.liuts.com/post/247/