Grupo: Gilmar, Victor, Tamires

1 - Introdução

Kubernetes é um sistema Open Source desenvolvido pelo Google para o gerenciamento de cluster de containeres tendo como características auto-scaling de serviços e containeres, auto-monitoração de containeres, permite o deploy de containers e serviços, load balancer, orquestração de containers, e orquestração de volumes de armazenamento 'storages'.

Minions – São todos os hosts que fazem parte do cluster gerenciado pelo Kubernetes.
Serviços rodando nos Minios: kube-proxy kubelet docker e flannel
Flannel – É uma rede virtual que vai provisionar uma sub-rede para cada host do cluster, está sub-rede será utilizada para alocar um ip para cada container e fazer a comunicação entre os containers.
Etcd – É um sistema distribuído de armazenamento do tipo key valor ou key=valor, utilizado pelo Kubernetes para armazenar todas informações sobre os Pods, containers, serviços, redes, nós do cluster, localização dos containeres, Cpus, Memória, versões de aplicativos e metados no geral.
Pod – É um grupo de um ou mais containers rodando dentro de um host/minion do cluster.
Replication Controller "RC" – Vai garantir que um determinado número de Pods estejam sempre rodando, ele monitora o cluster e em caso de algum Pod falhar, ficar off-line e etc, o replication controller tratará de subir um novo container ao Pod, mantendo o cluster funcional.

2 - Pré-requisitos e Infraestrutura

Sistema Operacional **Centos 7 x86_64** instalação mínima.
Hosts – Infraestrutura
Ex: um host para ser o Kubernetes e 3 Minions. (Minions podem ser a quantidade que quiser).

10.0.2.30 kubernetes-master1.devopslab.com.br
10.0.2.31 kubernetes-minion1.devopslab.com.br
10.0.2.32 kubernetes-minion2.devopslab.com.br
10.0.2.33 kubernetes-minion3.devopslab.com.br

3 - DNS – Configuração do DNS

Configure o /etc/hosts de todos os nós do cluster "minions" para ter os apontamentos de DNS. Basicamente copie as informações de dns acima para todos os hosts.

4 - Configuração do repositório para a instalação de pacotes

Crie o repositório em todos os hosts MASTER e MINIONS.

#vi /etc/yum.repos.d/virt7-docker-common-release.repo
Ex:
[virt7-docker-common-release]
name=virt7-docker-common-release
baseurl=http://cbs.centos.org/repos/virt7-docker-common-release/x86_64/os/
gpgcheck=0

5 - Instalação do Kubernetes e Docker

MASTER
#yum install kubernetes flannel etcd
O pacote kubernetes também vai instalar o Docker entre outras dependências.

MINIONS
#yum install kubernetes flannel

6 - Configuração do Kubernetes MASTER e os MINIONS

MASTER – Kubernetes Master.
MINIONS – Hosts Minions.
Desative o Firewall e Selinux em todos os hosts MASTER e MINIONS do cluster.

Ex:

#systemctl stop firewalld.service
#systemctl disable firewalld.service
#sed -i s/SELINUX=enforcing/SELINUX=disabled/g /etc/selinux/config
#setenforce 0

6.1 – Configuração do Kubernetes Master

MASTER – Kubernetes Config
#vi /etc/kubernetes/config
Configure a linha **KUBE_MASTER** para apontar para o ip MASTER.
Ex:
# logging to stderr means we get it in the systemd journal
KUBE_LOGTOSTDERR="--logtostderr=true"

# journal message level, 0 is debug
KUBE_LOG_LEVEL="--v=0"

# Should this cluster be allowed to run privileged docker containers
KUBE_ALLOW_PRIV="--allow-privileged=false"

# How the controller-manager, scheduler, and proxy find the apiserver
KUBE_MASTER="--master=http://kubernetes-master1.devopslab.com.br:8080"

MASTER – Apiserver
Comente a linha **KUBE_ADMISSION_CONTROL**.
#vi /etc/kubernetes/apiserver
Ex:

```
# The address on the local server to listen to.
KUBE_API_ADDRESS="--address=0.0.0.0"

# The port on the local server to listen on.
KUBE_API_PORT="--port=8080"

# Port minions listen on
KUBELET_PORT="--kubelet-port=10250"

# Comma separated list of nodes in the etcd cluster
KUBE_ETCD_SERVERS="--etcd-servers=http://127.0.0.1:2379"

# Address range to use for services
KUBE_SERVICE_ADDRESSES="--service-cluster-ip-range=10.254.0.0/16"

# default admission control policies
#KUBE_ADMISSION_CONTROL="--admission-
#control=NamespaceLifecycle,NamespaceExists,LimitRanger,SecurityContextD
eny,ServiceAccount,ResourceQuota"

# Add your own!
KUBE_API_ARGS=""
```

MASTER – ETCD
Verifique se a porta do Etcd é 2379.
Configure o bind para 0.0.0.0:2379.
#vi /etc/etcd/etcd.conf

Ex:

```
ETCD_NAME=default
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_CLIENT_URLS="http://0.0.0.0:2379"
ETCD_ADVERTISE_CLIENT_URLS="http://localhost:2379"
```

MASTER – Restart dos serviços e habilitação do start no boot.

Ex:

```
[root@kubernetes-master1 ~]# for SERVICES in etcd kube-apiserver kube-
controller-manager kube-scheduler; do
        systemctl restart $SERVICES
        systemctl enable $SERVICES
        systemctl status $SERVICES
done
```

MASTER – Definição de uma rede FLANNEL
Vamos criar uma rede para que seja alocada para cada novo container do cluster.
Normalmente é uma rede 172.17.0.0/16, porém vou mudar para uma do meu interesse.
Então vamos criar uma chave valor no Etcd.
Crie um arquivo .json em qualquer pasta do servidor Master e defina a rede.

#vi flannel-config.json
Ex:

```
{
    "Network": "10.0.10.0/16",
    "SubnetLen": 24,
    "Backend": {
        "Type": "vxlan",
        "VNI": 1
     }
}
```

Agora faça a criação a Key no Etcd.

#etcdctl set /atomic.io/network/config < flannel-config.json

Verificando se a key foi criada com o comando
#etcdctl get /atomic.io/network/config

Ex:

```
#etcdctl get /atomic.io/network/config
{
    "Network": "10.0.10.0/16",
    "SubnetLen": 24,
    "Backend": {
        "Type": "vxlan",
        "VNI": 1
     }
}
```

6.2 – Configuração dos Minions (Nodes)

Configuração dos hosts Minions.
Cada passo descrito abaixo deve ser feito em todos os hosts Minions da rede.

MINIONS – Kubernetes Config
Altere apenas a linha **KUBE_MASTER** e informe o servidor do Kubernetes.
#vi /etc/kubernetes/config

Ex:

```
# logging to stderr means we get it in the systemd journal
KUBE_LOGTOSTDERR="--logtostderr=true"
```

```
# journal message level, 0 is debug
KUBE_LOG_LEVEL="--v=0"

# Should this cluster be allowed to run privileged docker containers
KUBE_ALLOW_PRIV="--allow-privileged=false"

# How the controller-manager, scheduler, and proxy find the apiserver
KUBE_MASTER="--master=http://kubernetes-master1.devopslab.com.br:8080"
```

MINIONS – KUBERLET

Aqui você vai informar qual é o servidor da Api, setar o hostname e bind do kuberlet.

#vi /etc/kubernetes/kubelet

Ex:

```
###
# kubernetes kubelet (minion) config

# The address for the info server to serve on (set to 0.0.0.0 or "" for all
interfaces)
KUBELET_ADDRESS="--address=0.0.0.0"

# The port for the info server to serve on
# KUBELET_PORT="--port=10250"

# You may leave this blank to use the actual hostname
KUBELET_HOSTNAME="--hostname-override=kubernetes-
minion1.devopslab.com.br"

# location of the api-server
KUBELET_API_SERVER="--api-servers=http://kubernetes-
master1.devopslab.com.br:8080"

# pod infrastructure container
KUBELET_POD_INFRA_CONTAINER="--pod-infra-container-
image=registry.access.redhat.com/rhel7/pod-infrastructure:latest"

# Add your own!
KUBELET_ARGS=""
```

MINIONS – rede FLANNEL

Será alterado apenas a linha que informa o ip do ETCD rodando no servidor MASTER.

#vi /etc/sysconfig/flanneld

Ex:

```
# etcd url location.  Point this to the server where etcd runs
FLANNEL_ETCD="http://kubernetes-master1.devopslab.com.br:2379"

# etcd config key.  This is the configuration key that flannel queries
# For address range assignment
```

FLANNEL_ETCD_KEY="/atomic.io/network"

Any additional options that you want to pass
#FLANNEL_OPTIONS=""
Observe a linha: '**FLANNEL_ETCD_KEY="/atomic.io/network"**'
Esta key "**/atomic.io/network**" foi criada anteriormente no ETCD do Kubernetes Master.
MINIONS – Restart dos serviços e habilitação do start no boot

Ex:

[root@kubernetes-minion1 ~]# for SERVICES in kube-proxy kubelet docker flanneld; do
    systemctl restart $SERVICES
    systemctl enable $SERVICES
    systemctl status $SERVICES
done

6.3 – Validação do MASTER e MINIONS

MINIONS
Verificação da rede.
Verifique se foram criadas as redes para o docker e flannel.

Ex:
[root@kubernetes-minion1 ~]# ip -4 a|grep inet
    inet 127.0.0.1/8 scope host lo
    inet 10.0.2.31/24 brd 10.0.2.255 scope global enp0s3
    inet 10.0.10.1/24 scope global docker0
    inet 10.0.10.0/16 scope global flannel.1

Teste de consulta ao Etcd
#curl -s http://kubernetes-master1.devopslab.com.br:2379/v2/keys/atomic.io/network/subnets | python -mjson.tool

Ex:
[root@kubernetes-minion1 ~]#curl -s http://kubernetes-master1.devopslab.com.br:2379/v2/keys/atomic.io/network/subnets | python -mjson.tool

{
    "action": "get",
    "node": {
        "createdIndex": 32,
        "dir": true,
        "key": "/atomic.io/network/subnets",
        "modifiedIndex": 32,
        "nodes": [
            {
                "createdIndex": 133,

```
            "expiration": "2016-04-03T01:08:09.060073648Z",
            "key": "/atomic.io/network/subnets/10.0.10.0-24",
            "modifiedIndex": 133,
            "ttl": 84138,
            "value": "{\"PublicIP\":\"10.0.2.31\",\"BackendType\":\"vxlan\",\"BackendData\":{
\"VtepMAC\":\"8e:28:15:ca:cd:3b\"}}"
         },
         {
            "createdIndex": 374,
            "expiration": "2016-04-03T01:42:01.861701761Z",
            "key": "/atomic.io/network/subnets/10.0.14.0-24",
            "modifiedIndex": 374,
            "ttl": 86171,
            "value": "{\"PublicIP\":\"10.0.2.32\",\"BackendType\":\"vxlan\",\"BackendData\":{
\"VtepMAC\":\"b6:1b:ee:eb:ce:ef\"}}"
         },
         {
            "createdIndex": 434,
            "expiration": "2016-04-03T01:45:36.592848197Z",
            "key": "/atomic.io/network/subnets/10.0.91.0-24",
            "modifiedIndex": 434,
            "ttl": 86385,
            "value": "{\"PublicIP\":\"10.0.2.33\",\"BackendType\":\"vxlan\",\"BackendData\":{
\"VtepMAC\":\"4e:84:7d:78:21:00\"}}"
         }
      ]
   }
}
```


MASTER
No servidor Master faça um “kubectl get nodes” para verificar todos os nós do cluster.
#kubectl get nodes

7. Criação de Pods com o Kubernets

Vamos criar 3 containeres com a imagem nginx e com a porta 80 exposta e um replication controller nomeado como webserver-nginx.

#kubectl run webserver-nginx –image=nginx –replicas=3 –port=80

Ex:
[root@kubernetes-master1 ~]# kubectl run webserver-nginx --image=nginx --replicas=3 --port=80
replicationcontroller "webserver-nginx" created

Listando os Pods do cluster
#kubectl get pods

Ex:
[root@kubernetes-master1 ~]# kubectl get pods

```
NAME                    READY     STATUS    RESTARTS   AGE
webserver-nginx-2th95   0/1       Pending   0          33m
webserver-nginx-v4404   0/1       Pending   0          33m
webserver-nginx-za52c   0/1       Pending   0          33m
```

O status pending é normal, pois o Kubernetes está provisionando os Pods, pode demorar alguns minutos dependendo do tamanho da imagem.

No final você terá todos os Pods rodando.
#kubectl get pods

Ex:

```
[root@kubernetes-master1 ~]#  kubectl get pods
NAME                    READY     STATUS    RESTARTS   AGE
webserver-nginx-2th95   1/1       Running   0          54m
webserver-nginx-v4404   1/1       Running   0          54m
webserver-nginx-za52c   1/1       Running   1          54m
```

Verificando o Replication controller
#kubectl get rc

Ex:

```
[root@kubernetes-master1 ~]# kubectl get rc
CONTROLLER        CONTAINER(S)     IMAGE(S)   SELECTOR            REPL
ICAS   AGE
webserver-nginx   webserver-nginx   nginx      run=webserver-
nginx   3         59m
```

Verificando informações sobre um Pod.
Você criou um Pod pelo Kubernetes mas você não sabe muita coisa sobre ele, como os Ips.
#kubectl describe pod webserver-nginx-2th95

Consultando os Ips de todos os Pods de um específico Replication Controller.
#kubectl get pods -l run=webserver-nginx -o yaml | grep podIP

Ex:

```
[root@kubernetes-master1 ~]# kubectl get pods -l run=webserver-nginx -o yaml
| grep podIP
    podIP: 10.0.14.2
    podIP: 10.0.10.2
    podIP: 10.0.91.2
```

8. Criando um serviço e acessando os serviços dos Pods

Até o momento nós temos o Kubernetes rodando, Pods criados, porta 80 exposta nos containeres, e Nginx instalado.

Mas como você vai acessar os serviço http do Nginx? Qual é a Url? Qual é o Ip?.
Veja quando nós temos um cluster auto gerenciável como nós criamos aqui, a ideia é que todos os Pods estejam rodando e quando algum deles cair o próprio cluster se encarregará de subir um novo container, e este container com um novo IP.

Então não adianta eu saber os Ips do containers, se um deles morrer, será criado um outro, com um novo Ip, Id e etc.

Aí que entra a ideia de Serviço, vamos criar um serviço que atuará como Loadbalancer do cluster, este serviço vai disponibilizar um IP do cluster, e então acessar os serviços por este Ip, tanto faz quais são os containeres, Ips, onde estão e etc, queremos apenas o ip do cluster, e que este responda pelos containers.

Vamos ver como isto funciona.
Consultando o RC.
#kubectl get rc
Criando um serviço.
#kubectl expose rc webserver-nginx –port=80

Consultando o IP do cluster.
#kubectl get service webserver-nginx

Consultando os Ips dos Pods do Cluster.
kubectl describe service webserver-nginx

Nosso **CLUSTER_IP é 10.254.174.71** e este IP é acessível apenas de dentro da nossa Infraestrutura, ou seja apenas os hosts Minions tem acesso a este ip. Então a partir de qualquer host Minion você pode fazer um curl para validar se os serviços de Nginx estão respondendo.
#curl 10.254.174.71

Ex:
[root@kubernetes-minion1 ~]# curl 10.254.174.71

<!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>
[root@kubernetes-minion1 ~]#
```

Agora se você quer acessar o cluster externamente, torná-lo público ou disponível para outras redes vamos utilizar algumas das opções como **Loadbalancer, EXTERNAL_IP ou NodePort**. **NodePort** é o conhecido expose do Docker, você vai criar um Pod e expor a porta do host Docker/Minion, podendo acessar o serviço do container através do Ip do host Docker/Minion.
Lembrando **CLUSTER_IP** é acessível apenas dentro da rede dos hosts Minions.