

kubernetes集群构建 - new

笔记本： 新课程笔记
创建时间： 2018/9/5 星期三 上午 9:58 更新时间： 2018/10/12 星期五 下午 6:35
作者： 306798658@qq.com
URL： <https://github.com/gjmzj/kubeasz/blob/master/docs/00-%E9%9B%86%E7%BE%A4%E8%A7%84%E5%88%9...>

本文档参考 <https://github.com/gjmzj/kubeasz>

扩展：使用kubeadm部署集群 <https://blog.frognew.com/2018/08/kubeadm-install-kubernetes-1.11.html>

软硬件限制：

- 1) cpu和内存 master：至少1c2g，推荐2c4g；node：至少1c2g
- 2) linux系统 内核版本至少3.10，推荐CentOS7/RHEL7
- 3) docker 至少1.9版本，推荐1.12+
- 4) etcd 至少2.0版本，推荐3.0+

kubernetes官方github地址 <https://github.com/kubernetes/kubernetes/releases>

高可用集群所需节点规划：

部署节点-----x1：运行这份 ansible 脚本的节点
etcd节点-----x3：注意etcd集群必须是1,3,5,7...奇数个节点
master节点----x2：根据实际集群规模可以增加节点数，需要额外规划一个master VIP(虚地址)
lb节点-----x2：负载均衡节点两个，安装 haproxy+keepalived
node节点-----x3：真正应用负载的节点，根据需要提升机器配置和增加节点数

机器规划：

ip	主机名	角色
172.7.15.113(128)	aming-master	deploy、master1、lb1、etcd
172.7.15.114(129)	aming-node1	etcd、node
172.7.15.115(130)	aming-node2	etcd、node
172.7.15.116(131)	aming-master2	master2、lb2
172.7.15.118(150)		vip

准备工作

四台机器，全部执行：

```
yum install epel-release
yum update
yum install python
```

deploy节点安装和准备ansible

```
yum install -y python-pip git
pip install pip --upgrade -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com
pip install --no-cache-dir ansible -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com
```

deploy节点配置免密码登录

- 1) ssh-keygen 生产密钥
- 2) for ip in 113 114 115 116; do ssh-copy-id 172.7.15.\$ip; done

deploy上编排k8s

```
git clone https://github.com/gjmzj/kubeasz.git
mkdir -p /etc/ansible
mv kubeasz/* /etc/ansible/
```

从百度云网盘下载二进制文件 <https://pan.baidu.com/s/1c4RFaA#list/path=%2F>

可以根据自己所需版本，下载对应的tar包，这里我下载1.11

经过一番折腾，最终把k8s.1-11-2.tar.gz的tar包放到了depoly上

```
tar zxvf k8s.1-11-2.tar.gz
mv bin/* /etc/ansible/bin/
```

配置集群参数

```
cd /etc/ansible/
cp example/hosts.m-masters.example hosts //根据实际情况修改IP地址
[deploy]
172.7.15.113 NTP_ENABLED=no

[etcd]
172.7.15.113 NODE_NAME=etcd1
172.7.15.114 NODE_NAME=etcd2
172.7.15.115 NODE_NAME=etcd3

[kube-master]
172.7.15.113
172.7.15.116

[lb]
172.7.15.113 LB_IF="eno16777736" LB_ROLE=backup # 注意根据实际使用网卡设置 LB_IF变量
172.7.15.116 LB_IF="ens33" LB_ROLE=master

[kube-node]
172.7.15.114
172.7.15.115
```

修改完hosts，测试

```
ansible all -m ping
```

分步骤安装：

- 1) 创建证书和安装准备

```
ansible-playbook 01.prepare.yml
```

- 2) 安装etcd集群

```
ansible-playbook 02.etcd.yml
```

检查etcd节点健康状况：

```
for ip in 128 129 130 ; do ETCDCTL_API=3 etcdctl --endpoints=https://192.168.111.$ip:2379 --
cacert=/etc/kubernetes/ssl/ca.pem --cert=/etc/etcd/ssl/etcd.pem --key=/etc/etcd/ssl/etcd-key.pem endpoint health; done
```

- 3) 安装docker

```
ansible-playbook 03.docker.yml
```

- 4) 安装master节点

```
ansible-playbook 04.kube-master.yml
kubectl get componentstatus //查看集群状态
```

NAME	STATUS	MESSAGE	ERROR
scheduler	Healthy	ok	
controller-manager	Healthy	ok	
etcd-1	Healthy	{"health": "true"}	
etcd-2	Healthy	{"health": "true"}	
etcd-0	Healthy	{"health": "true"}	

5) 安装node节点

```
ansible-playbook 05.kube-node.yml
```

查看node节点

```
kubectl get nodes
```

6) 部署集群网络

```
ansible-playbook 06.network.yml
kubectl get pod -n kube-system //查看kube-system namespace上的pod，从中可以看到flannel相关的pod
```

7) 安装集群插件(dns, dashboard)

```
ansible-playbook 07.cluster-addon.yml
```

查看kube-system namespace下的服务

```
kubectl get svc -n kube-system
```

一步安装

```
ansible-playbook 90.setup.yml
```

查看集群信息：

```
kubectl cluster-info
```

查看node/pod使用资源情况：

```
kubectl top node
kubectl top pod --all-namespaces
```

测试DNS

a) 创建nginx service

```
kubectl run nginx --image=nginx --expose --port=80
```

b) 创建busybox 测试pod

```
kubectl run busybox --rm -it --image=busybox /bin/sh //进入到busybox内部
nslookup nginx.default.svc.cluster.local //结果如下
Server:      10.68.0.2
Address:     10.68.0.2:53

Name:   nginx.default.svc.cluster.local
Address: 10.68.9.156
```

增加node节点

1) deploy节点免密码登录node

ssh-copy-id 新node ip

2) 修改/etc/ansible/hosts

```
[new-node]
172.7.15.117
```

3) 执行安装脚本

```
ansible-playbook /etc/ansible/20.addnode.yml
```

4) 验证

```
kubectl get node
kubectl get pod -n kube-system -o wide
```

5) 后续工作

修改/etc/ansible/hosts，将new-node里面的所有ip全部移动到kube-node组里去

增加master节点（略）

<https://github.com/gjmzj/kubeasz/blob/master/docs/op/AddMaster.md>

升级集群

1) 备份etcd

```
ETCDCTL_API=3 etcdctl snapshot save backup.db
```

查看备份文件信息

```
ETCDCTL_API=3 etcdctl --write-out=table snapshot status backup.db
```

2) 到本项目的根目录kubeasz

```
cd /dir/to/kubeasz
拉取最新的代码
git pull origin master
```

3) 下载升级目标版本的kubernetes二进制包（百度网盘<https://pan.baidu.com/s/1c4RFaA#list/path=%2F>）
解压，并替换/etc/ansible/bin/下的二进制文件

4) docker升级（略），除非特别需要，否则不建议频繁升级docker

5) 如果接受业务中断，执行：

```
ansible-playbook -t upgrade_k8s,restart_dockerd 22.upgrade.yml
```

6) 不能接受短暂中断，需要这样做：

```
a) ansible-playbook -t upgrade_k8s 22.upgrade.yml
b) 到所有node上逐一：
    kubectl cordon和kubectl drain    //迁移业务pod
    systemctl restart docker
    kubectl uncordon    //恢复pod
```

备份和恢复

1) 备份恢复原理：

备份，从运行的etcd集群中备份数据到磁盘文件

恢复，把etcd的备份文件恢复到etcd集群中，然后据此重建整个集群

2) 如果使用kubeadm项目创建的集群，除了备份etcd数据外，还需要备份CA证书文件，以及ansible的hosts文件

3) 手动操作步骤：

```
mkdir -p /backup/k8s //创建备份目录
ETCDCTL_API=3 etcdctl snapshot save /backup/k8s/snapshot.db //备份etcd数据
cp /etc/kubernetes/ssl/ca* /backup/k8s/ //备份ca证书
deploy节点执行 ansible-playbook /etc/ansible/99.clean.yml //模拟集群崩溃
```

恢复步骤如下（在deploy节点）：

a) 恢复ca证书

```
mkdir -p /etc/kubernetes/ssl
cp /backup/k8s/ca* /etc/kubernetes/ssl/
```

b) 重建集群

```
cd /etc/ansible
ansible-playbook 01.prepare.yml
ansible-playbook 02.etcd.yml
ansible-playbook 03.docker.yml
ansible-playbook 04.kube-master.yml
ansible-playbook 05.kube-node.yml
```

c) 恢复etcd数据

停止服务

```
ansible etcd -m service -a 'name=etcd state=stopped'
```

清空文件

```
ansible etcd -m file -a 'name=/var/lib/etcd/member/ state=absent'
```

登录所有的etcd节点，参照本etcd节点/etc/systemd/system/etcd.service的服务文件，替换如下{}中变量后执行

```
cd /backup/k8s/
ETCDCTL_API=3 etcdctl snapshot restore snapshot.db \
  --name etcd1 \
  --initial-cluster etcd1=https://192.168.111.128:2380,etcd2=https://192.168.111.129:2380,etcd3=https://192.168.111.130:2380 \
  --initial-cluster-token etcd-cluster-0 \
  --initial-advertise-peer-urls https://192.168.111.128:2380
```

执行上面的步骤后，会生成{{ NODE_NAME }}.etcd目录

```
cp -r etcd1.etcd/member /var/lib/etcd/

systemctl restart etcd
```

d) 在deploy节点重建网络

```
ansible-playbook /etc/ansible/tools/change_k8s_network.yml
```

4) 不想手动恢复，可以用ansible自动恢复

需要一键备份

```
ansible-playbook /etc/ansible/23.backup.yml
```

检查/etc/ansible/roles/cluster-backup/files目录下是否有文件

```
tree /etc/ansible/roles/cluster-backup/files/ //如下
├── ca                                # 集群CA 相关备份
│   ├── ca-config.json
│   ├── ca.csr
│   ├── ca-csr.json
│   ├── ca-key.pem
│   └── ca.pem
├── hosts                            # ansible hosts备份
│   ├── hosts                        # 最近的备份
│   └── hosts-201807231642
├── readme.md
├── snapshot                         # etcd 数据备份
│   ├── snapshot-201807231642.db
│   └── snapshot.db                 # 最近的备份
```

模拟故障：

```
ansible-playbook /etc/ansible/99.clean.yml
```

修改文件/etc/ansible/roles/cluster-restore/defaults/main.yml，指定要恢复的etcd快照备份，如果不修改就是最新的一次

恢复操作：

```
ansible-playbook /etc/ansible/24.restore.yml
ansible-playbook /etc/ansible/tools/change_k8s_network.yml
```