# Kubernetes-1.8.x-生产环境部署参考手册

## 环境准备：

部署全过程用 root 用户。

### 硬件软件

硬件：集群要求如下配置服务器至少需要三台作为 Master（master01，master02，master03），有多余的服务器可作为 Node（node01，node02）。

操作系统：CentOS Linux release 7.4.1708 (Core) 或以上。

硬件规格：CPU 8核以上，内存 16 GB 以上，磁盘 300 GB以上。

### 软件版本

|  |  |  |
| --- | --- | --- |
| 软件名称 | 版本 | 备注 |
| kubernetes | v1.8.x |  |
| docker | 17.03.2-ce | 社区版本 |
| docker-compose | 1.16.1 |  |
| harbor | v1.2.2 |  |
| Etcd | v3.2.9 |  |

### 系统配置

关闭所有服务器的 selinux 和 firewalld

|  |
| --- |
| # 关闭 selinux 和防火墙firewalld  setenforce 0 && \  systemctl disable firewalld && \  systemctl stop firewalld  # 编辑 /etc/selinux/config  vi /etc/selinux/config  # 修改，注意是 disabled，修改错了会导致服务器开机不了  SELINUX=disabled |

修改所有服务器的 /etc/sysctl.d/k8s.conf文件

|  |
| --- |
| cat << EOF > /etc/sysctl.d/k8s.conf  net.bridge.bridge-nf-call-ip6tables = 1  net.bridge.bridge-nf-call-iptables = 1  EOF  # 执行命令使其生效  modprobe br\_netfilter && \  sysctl -p /etc/sysctl.d/k8s.conf |

编辑所有服务器的 /etc/hosts文件，配置host

|  |
| --- |
| # vi /etc/hosts  # master  192.168.151.1 master01  192.168.151.2 master02  192.168.151.3 master03  # node  192.168.151.4 node01  192.168.151.5 node02 |

在所有服务器上创建 kube 用户

|  |
| --- |
| useradd kube -r -s /sbin/nologin |

所有 Master 服务器上创建用户 etcd

|  |
| --- |
| useradd etcd -r -s /sbin/nologin |

\*\* 必要的情况下设置服务器之间的免密码/密钥登陆

|  |
| --- |
| # 生成密钥对，ssh-keygen 一直回车默认即可  ssh-keygen  # 设置密钥登陆  ssh-copy-id root@master01  # 之后就可以直接密钥登陆  ssh root@master01 |

## 准备 Docker 环境

Master 和 Node 的每一台服务器都必须部署上 Docker 环境。

|  |
| --- |
| cd ~ && \  tar zxvf docker-ce-17.03.2.ce.tar.gz && \  cd docker-ce-17.03.2.ce && \  rpm -ivU \*.rpm && \  cd ~ && \  systemctl start docker && \  systemctl enable docker |

## 创建证书

在集群内某一台服务器上创建必要的证书，然后将生成的证书分发到对应的 Master 或者 Node 服务器上。

### 安装 cfssl

解压并安装上传的 cfssl.tar.gz

|  |
| --- |
| tar zxvf cfssl.tar.gz && \  /usr/bin/cp -r cfssl/\* /usr/bin/ |

### CA 证书

创建 CA 证书配置，生成 CA 证书和私钥

|  |
| --- |
| mkdir /opt/ssl  cd /opt/ssl  # 创建 config.json 文件  cat << EOF > config.json  {  "signing": {  "default": {  "expiry": "87600h"  },  "profiles": {  "kubernetes": {  "expiry": "87600h",  "usages": [  "signing",  "key encipherment",  "server auth",  "client auth"  ]  }  }  }  }  EOF  # 创建 csr.json 文件  cat << EOF > csr.json  {  "CN": "kubernetes",  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "CN",  "L": "ShenZhen",  "ST": "GuangDong",  "O": "k8s",  "OU": "System"  }  ]  }  EOF  # 生成 CA 证书和私钥  cfssl gencert -initca csr.json | cfssljson -bare ca  # 最后 CA 有关证书列表如下  [root@localhost ssl]# ll  total 20  -rw-r--r--. 1 root root 1005 Oct 24 17:26 ca.csr  -rw-------. 1 root root 1679 Oct 24 17:26 ca-key.pem  -rw-r--r--. 1 root root 1367 Oct 24 17:26 ca.pem  -rw-r--r--. 1 root root 386 Oct 24 17:23 config.json  -rw-r--r--. 1 root root 268 Oct 24 17:22 csr.json |

### Etcd 证书

创建 etcd 证书配置，生成 etcd证书和私钥

|  |
| --- |
| mkdir -p /opt/ssl/etcd  cd /opt/ssl/etcd/  # 创建 etcd-csr.json  cat << EOF > etcd-csr.json  {  "CN": "etcd",  "hosts": [  "127.0.0.1",  "192.168.151.1",  "192.168.151.2",  "192.168.151.3"  ],  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "CN",  "L": "ShenZhen",  "ST": "GuangDong",  "O": "k8s",  "OU": "System"  }  ]  }  EOF  # 生成 etcd 证书和私钥  cfssl gencert -ca=/opt/ssl/ca.pem \  -ca-key=/opt/ssl/ca-key.pem \  -config=/opt/ssl/config.json \  -profile=kubernetes etcd-csr.json | cfssljson -bare etcd  # etcd 有关证书证书列表如下  [root@localhost etcd]# ll  total 16  -rw-r--r--. 1 root root 1066 Oct 24 18:40 etcd.csr  -rw-r--r--. 1 root root 312 Oct 24 18:40 etcd-csr.json  -rw-------. 1 root root 1679 Oct 24 18:40 etcd-key.pem  -rw-r--r--. 1 root root 1444 Oct 24 18:40 etcd.pem |

### kube-apiserver 证书

创建 kube-apiserver 证书配置，生成 kube-apiserver 证书和私钥

|  |
| --- |
| mkdir -p /opt/ssl/kube-apiserver  cd /opt/ssl/kube-apiserver/  # 创建 kube-apiserver-csr.json  cat << EOF > kube-apiserver-csr.json  {  "CN": "kubernetes",  "hosts": [  "127.0.0.1",  "192.168.151.1",  "192.168.151.2",  "192.168.151.3",  "10.254.0.1",  "localhost",  "kubernetes",  "kubernetes.default",  "kubernetes.default.svc",  "kubernetes.default.svc.cluster",  "kubernetes.default.svc.cluster.local"  ],  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "CN",  "L": "ShenZhen",  "ST": "GuangDong",  "O": "k8s",  "OU": "System"  }  ]  }  EOF  # 生成 kube-apiserver 证书和私钥  cfssl gencert -ca=/opt/ssl/ca.pem \  -ca-key=/opt/ssl/ca-key.pem \  -config=/opt/ssl/config.json \  -profile=kubernetes kube-apiserver-csr.json | cfssljson -bare kube-apiserver  # kube-apiserver有关证书证书列表如下  [root@localhost kube-apiserver]# ll  total 16  -rw-r--r--. 1 root root 1277 Oct 24 19:51 kube-apiserver.csr  -rw-r--r--. 1 root root 499 Oct 24 19:51 kube-apiserver-csr.json  -rw-------. 1 root root 1679 Oct 24 19:51 kube-apiserver-key.pem  -rw-r--r--. 1 root root 1651 Oct 24 19:51 kube-apiserver.pem |

### kube-proxy 证书

创建 kube-proxy 证书配置，生成 kube-proxy 证书和私钥。

|  |
| --- |
| mkdir -p /opt/ssl/kube-proxy  cd /opt/ssl/kube-proxy  # 创建 kube-proxy-csr.json  cat << EOF > kube-proxy-csr.json  {  "CN": "system:kube-proxy",  "hosts": [  ],  "key": {  "algo": "rsa",  "size": 2048  },  "names": [  {  "C": "CN",  "ST": "Wuhan",  "L": "Hubei",  "O": "system:kube-proxy",  "OU": "System"  }  ]  }  EOF  # 生成kube-proxy证书和私钥  cfssl gencert -ca=/opt/ssl/ca.pem \  -ca-key=/opt/ssl/ca-key.pem \  -config=/opt/ssl/config.json \  -profile=kubernetes kube-proxy-csr.json | cfssljson -bare kube-proxy  # kube-proxy有关证书证书列表如下  [root@localhost kube-proxy]# ll  total 16  -rw-r--r--. 1 root root 1025 Oct 25 09:54 kube-proxy.csr  -rw-r--r--. 1 root root 243 Oct 25 09:52 kube-proxy-csr.json  -rw-------. 1 root root 1679 Oct 25 09:54 kube-proxy-key.pem  -rw-r--r--. 1 root root 1419 Oct 25 09:54 kube-proxy.pem |

### kubelet token.csv

生成的这个 token.csv 需要分发到所有的 master服务器的 /etc/kubernetes/known\_token/ 目录下，后面再说明。

|  |
| --- |
| mkdir -p /opt/ssl/kubelet  cd /opt/ssl/kubelet  # 生成 token.csv  TOKEN=$(head -c 16 /dev/urandom | od -An -t x | tr -d ' ') && \  echo $TOKEN,kubelet-bootstrap,10001,"system:kubelet-bootstrap" > token.csv |

## 证书分发

把生成好的证书分发到对应的 Master 或者 Node 服务器上

### Master

因为本文档的 Etcd 也选择部署再 Master 服务器上，所有证书一起拷贝过去，如果 Etcd 部署再单独的集群机器上，则另外拷贝，方式都是一样的。

拷贝 /opt/ssl/ 和 /opt/ssl/ kube-apiserver / 目录下的 pem 文件到每一个 Node 服务器的/etc/kubernetes/ssl 目录下。

|  |
| --- |
| for IP in master01 master02 master03;do  ssh root@$IP mkdir -p /etc/kubernetes/ssl  ssh root@$IP chown -R kube:kube /etc/kubernetes/ssl  scp /opt/ssl/ca\*.pem /opt/ssl/kube-apiserver/\*.pem root@$IP:/etc/kubernetes/ssl/    ssh root@$IP mkdir -p /etc/etcd/ssl  scp /opt/ssl/ca\*.pem /opt/ssl/etcd/\*.pem root@$IP:/etc/etcd/ssl  ssh root@$IP chown -R etcd:etcd /etc/etcd/ssl  ssh root@$IP chmod 644 /etc/etcd/ssl/{ca.pem,etcd-key.pem,etcd.pem}  ssh root@$IP mkdir -p /etc/kubernetes/known\_token  scp /opt/ssl/kubelet/token.csv root@$IP:/etc/kubernetes/known\_token/  ssh root@$IP chown -R kube:kube /etc/kubernetes/known\_token  done |

### Node

拷贝 /opt/ssl/ 和 /opt/ssl/kube-proxy/ 目录下的 pem 文件到每一个 Node 服务器的/etc/kubernetes/ssl 目录下。

|  |
| --- |
| for IP in node01 node02;do  ssh root@$IP mkdir -p /etc/kubernetes/ssl  scp /opt/ssl/ca\*.pem /opt/ssl/kube-proxy/\*.pem root@$IP:/etc/kubernetes/ssl/  ssh root@$IP chown -R kube:kube /etc/kubernetes/ssl  ssh root@$IP mkdir -p /etc/etcd/ssl  scp /opt/ssl/ca\*.pem /opt/ssl/etcd/\*.pem root@$IP:/etc/etcd/ssl  ssh root@$IP chown -R etcd:etcd /etc/etcd/ssl  ssh root@$IP chmod 644 /etc/etcd/ssl/{ca.pem,etcd-key.pem,etcd.pem}  done |

## Etcd 集群部署

### 部署集群

Etcd 选择部署在三台 Master 上面，如果有多余的机器可以单独出来部署，更加稳定。

在每台 Master 服务器上执行下面的部署操作。

|  |
| --- |
| cd ~  tar zxvf etcd-v3.2.9-linux-amd64.tar.gz  cd etcd-v3.2.9-linux-amd64  /usr/bin/cp -r etcd etcdctl /usr/bin/  mkdir –p /var/lib/etcd/  chown -R etcd:etcd /var/lib/etcd/  cd ~ |

创建启动 service，注意不同的 Master 的配置要修改标红色的文字。

|  |
| --- |
| cat << EOF > /usr/lib/systemd/system/etcd.service  [Unit]  Description=Etcd Server  After=network.target  After=network-online.target  Wants=network-online.target  [Service]  Type=notify  WorkingDirectory=/var/lib/etcd/  EnvironmentFile=-/etc/etcd/etcd.conf  User=etcd  # set GOMAXPROCS to number of processors  ExecStart=/bin/bash -c \\  "GOMAXPROCS=$(nproc) /usr/bin/etcd \\  --name etcd1 \\  --cert-file=/etc/etcd/ssl/etcd.pem \\  --key-file=/etc/etcd/ssl/etcd-key.pem \\  --peer-cert-file=/etc/etcd/ssl/etcd.pem \\  --peer-key-file=/etc/etcd/ssl/etcd-key.pem \\  --trusted-ca-file=/etc/etcd/ssl/ca.pem \\\  --peer-trusted-ca-file=/etc/etcd/ssl/ca.pem \\  --initial-advertise-peer-urls https://192.168.151.1:2380 \\  --listen-peer-urls https://192.168.151.1:2380 \\  --listen-client-urls https://192.168.151.1:2379,https://127.0.0.1:2379 \\  --advertise-client-urls https://192.168.151.1:2379 \\  --initial-cluster-token k8s\_etcd \\  --initial-cluster etcd1=https://192.168.151.1:2380 \\  --initial-cluster-state new \\  --data-dir=/var/lib/etcd/etcd1.etcd"  Restart=on-failure  LimitNOFILE=65536  RestartSec=5  [Install]  WantedBy=multi-user.target  EOF  # 启动和自启动 etcd  systemctl daemon-reload && systemctl enable etcd && systemctl start etcd && systemctl status etcd |

### 验证集群

|  |
| --- |
| etcdctl \  --endpoints=https://127.0.0.1:2379 \  --cert-file=/etc/etcd/ssl/etcd.pem \  --ca-file=/etc/etcd/ssl/ca.pem \  --key-file=/etc/etcd/ssl/etcd-key.pem \  cluster-health |

### 设置集群 flannel网段

注意：整个 Etcd 集群只需要操作一次即可

|  |
| --- |
| etcdctl \  --endpoints=https://127.0.0.1:2379 \  --cert-file=/etc/etcd/ssl/etcd.pem \  --ca-file=/etc/etcd/ssl/ca.pem \  --key-file=/etc/etcd/ssl/etcd-key.pem \  mkdir /flannel/network && \  etcdctl \  --endpoints=https://127.0.0.1:2379 \  --cert-file=/etc/etcd/ssl/etcd.pem \  --ca-file=/etc/etcd/ssl/ca.pem \  --key-file=/etc/etcd/ssl/etcd-key.pem \  mk /flannel/network/config "{ \"Network\": \"172.30.0.0/16\", \"SubnetLen\": 24, \"Backend\": { \"Type\": \"vxlan\" } }" |

## Flannel 部署

前面部署好了 Docker 和 etcd 之后就可以接着部署 flannel。

### 部署

|  |
| --- |
| cd ~  tar zxvf flannel-v0.9.0-linux-amd64.tar.gz  cd flannel  mkdir –p /usr/libexec/flannel/  /usr/bin/cp -r flanneld /usr/bin/ && /usr/bin/cp -r mk-docker-opts.sh /usr/libexec/flannel/ |

### 配置

flannel 的相关配置

|  |
| --- |
| ## 配置参数文件  cat << EOF > /etc/sysconfig/flanneld  # Flanneld configuration options  # etcd url location. Point this to the server where etcd runs  FLANNEL\_ETCD\_ENDPOINTS="https://192.168.151.1:2379, https://192.168.151.2:2379, https://192.168.151.3:2379"  # etcd config key. This is the configuration key that flannel queries  # For address range assignment  FLANNEL\_ETCD\_PREFIX="/flannel/network"  # Any additional options that you want to pass  #FLANNEL\_OPTIONS=""    #FLANNEL\_ETCD\_KEYFILE=/etc/etcd/ssl/etcd-key.pem  #FLANNEL\_ETCD\_CERTFILE=/etc/etcd/ssl/etcd.pem  #FLANNEL\_ETCD\_CAFILE=/etc/etcd/ssl/ca.pem  EOF  # 创建 /usr/bin/flanneld-start  cat << EOF > /usr/bin/flanneld-start  #!/bin/sh  exec /usr/bin/flanneld --etcd-endpoints="https://192.168.204.28:2379" \\  --etcd-prefix="/flannel/network" \\  --etcd-keyfile="/etc/etcd/ssl/etcd-key.pem" \\  --etcd-certfile="/etc/etcd/ssl/etcd.pem" \\  --etcd-cafile="/etc/etcd/ssl/ca.pem"  EOF  # 添加可执行属性  chmod +x /usr/bin/flanneld-start  # 配置 flanneld 的 service  cat << EOF > /usr/lib/systemd/system/flanneld.service  [Unit]  Description=Flanneld overlay address etcd agent  After=network.target  After=network-online.target  Wants=network-online.target  After=etcd.service  Before=docker.service  [Service]  Type=notify  EnvironmentFile=/etc/sysconfig/flanneld  EnvironmentFile=-/etc/sysconfig/docker-network  ExecStart=/usr/bin/flanneld-start  ExecStartPost=/usr/libexec/flannel/mk-docker-opts.sh -k DOCKER\_NETWORK\_OPTIONS -d /run/flannel/docker  Restart=on-failure  [Install]  WantedBy=multi-user.target  RequiredBy=docker.service  EOF |

docker 的相关配置

|  |
| --- |
| mkdir -p /usr/lib/systemd/system/docker.service.d  # 配置 flannel.conf 文件  cat << EOF > /usr/lib/systemd/system/docker.service.d/flannel.conf  [Service]  EnvironmentFile=-/run/flannel/docker  EOF  # 编辑 /usr/lib/systemd/system/docker.service  vi /usr/lib/systemd/system/docker.service  ExecStart=/usr/bin/dockerd  # 后面追加$DOCKER\_NETWORK\_OPTIONS 修改成如下：  ExecStart=/usr/bin/dockerd $DOCKER\_NETWORK\_OPTIONS |

设置 flannel 开机启动

|  |
| --- |
| systemctl enable flanneld && systemctl restart flanneld && systemctl status flanneld |

再重启 docker

|  |
| --- |
| systemctl restart docker && systemctl status docker |

## kubernetes 二进制分发

|  |
| --- |
| tar zxvf kubernetes-server-linux-amd64-1.8.x.tar.gz  cd kubernetes |

kubectl、kube-apiserver、kube-manager、kube-scheduler 分发到三台 master 服务器上。

|  |
| --- |
| for IP in master01 master02 master03;do  scp server/bin/{kube-apiserver,kube-controller-manager,kubectl,kube-scheduler} root@$IP: /usr/bin/  done |

kubectl、kubelet、kube-proxy 分发到两台 node 服务器上

|  |
| --- |
| for IP in node01 node02;do  scp server/bin/{kubectl,kubelet,kube-proxy} root@$IP: /usr/bin/  done |

## Master部署 kube-apiserver

|  |
| --- |
| # 配置 service  cat << EOF > /usr/lib/systemd/system/kube-apiserver.service  [Unit]  Description=Kubernetes API Server  Documentation=https://github.com/GoogleCloudPlatform/kubernetes  After=network.target  After=etcd.service  [Service]  EnvironmentFile=-/etc/kubernetes/config  EnvironmentFile=-/etc/kubernetes/apiserver  User=kube  ExecStart=/usr/bin/kube-apiserver \\  --admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,PersistentVolumeLabel,DefaultStorageClass,ResourceQuota,DefaultTolerationSeconds \\  --advertise-address=192.168.204.28 \\  --bind-address=192.168.204.28 \\  --insecure-bind-address=127.0.0.1 \\  --service-cluster-ip-range=10.254.0.0/16 \\  --service-node-port-range=30000-32000 \\  --allow-privileged=true \\  --apiserver-count=3 \\  --logtostderr=true \\  --v=0 \\  --audit-log-maxage=30 \\  --audit-log-maxbackup=3 \\  --audit-log-maxsize=100 \\  --authorization-mode=RBAC \\  --enable-swagger-ui=true \\  --event-ttl=1h \\  --secure-port=6443 \\  --insecure-port=8080 \\  --etcd-servers=https://192.168.204.28:2379 \\  --etcd-cafile=/etc/etcd/ssl/ca.pem \\  --etcd-certfile=/etc/etcd/ssl/etcd.pem \\  --etcd-keyfile=/etc/etcd/ssl/etcd-key.pem \\  --storage-backend=etcd3 \\  --tls-cert-file=/etc/kubernetes/ssl/kube-apiserver.pem \\  --tls-private-key-file=/etc/kubernetes/ssl/kube-apiserver-key.pem \\  --client-ca-file=/etc/kubernetes/ssl/ca.pem \\  --service-account-key-file=/etc/kubernetes/ssl/kube-apiserver-key.pem \\  --token-auth-file=/etc/kubernetes/known\_token/token.csv \\  --enable-bootstrap-token-auth=true \\  --kubelet-https=true \\  --anonymous-auth=False  Restart=on-failure  Type=notify  LimitNOFILE=65536  [Install]  WantedBy=multi-user.target  EOF  # 设置开机启动  systemctl daemon-reload && systemctl enable kube-apiserver && systemctl restart kube-apiserver && systemctl status kube-apiserver |

## Master部署 kube-controller-manager

|  |
| --- |
| # 配置 service  cat << EOF > /usr/lib/systemd/system/kube-controller-manager.service  [Unit]  Description=Kubernetes Controller Manager  Documentation=https://github.com/GoogleCloudPlatform/kubernetes  [Service]  EnvironmentFile=-/etc/kubernetes/config  EnvironmentFile=-/etc/kubernetes/controller-manager  User=kube  ExecStart=/usr/bin/kube-controller-manager \\  --address=127.0.0.1 \\  --master=http://127.0.0.1:8080 \\  --service-cluster-ip-range=10.254.0.0/16 \\  --cluster-name=kubernetes \\  --cluster-signing-cert-file=/etc/kubernetes/ssl/ca.pem \\  --cluster-signing-key-file=/etc/kubernetes/ssl/ca-key.pem \\  --service-account-private-key-file=/etc/kubernetes/ssl/kube-apiserver-key.pem \\  --root-ca-file=/etc/kubernetes/ssl/ca.pem \\  --leader-elect=true \\  --node-monitor-grace-period=40s \\  --node-monitor-period=5s \\  --pod-eviction-timeout=5m0s \\  --v=2  Restart=on-failure  LimitNOFILE=65536  [Install]  WantedBy=multi-user.target  EOF  # 设置开机启动  systemctl daemon-reload && systemctl enable kube-controller-manager && systemctl restart kube-controller-manager && systemctl status kube-controller-manager |

## Master部署 kube-scheduler

|  |
| --- |
| # 配置 service  cat << EOF > /usr/lib/systemd/system/kube-scheduler.service  [Unit]  Description=Kubernetes Scheduler Plugin  Documentation=https://github.com/GoogleCloudPlatform/kubernetes  [Service]  EnvironmentFile=-/etc/kubernetes/config  EnvironmentFile=-/etc/kubernetes/scheduler  User=kube  ExecStart=/usr/bin/kube-scheduler \\  --address=127.0.0.1 \\  --logtostderr=true \\  --v=2 \\  --master=127.0.0.1:8080 \\  --leader-elect=true  Restart=on-failure  LimitNOFILE=65536  [Install]  WantedBy=multi-user.target  EOF  # 设置开机启动  systemctl daemon-reload && systemctl enable kube-scheduler && systemctl restart kube-scheduler && systemctl status kube-scheduler |

## Node 部署代理到 kube-apiserver 的 Nginx

|  |
| --- |
| mkdir -p /etc/nginx  # 创建代理配置文件  cat << EOF > /etc/nginx/nginx.conf  error\_log stderr notice;  worker\_processes auto;  events {  multi\_accept on;  use epoll;  worker\_connections 1024;  }  stream {  upstream kube\_apiserver {  least\_conn;  server 192.168.204.28:6443;  }  server {  listen 0.0.0.0:6443;  proxy\_pass kube\_apiserver;  proxy\_timeout 10m;  proxy\_connect\_timeout 1s;  }  }  EOF  # 创建 system service  cat << EOF > /etc/systemd/system/nginx-proxy.service  [Unit]  Description=kubernetes apiserver docker wrapper  Wants=docker.socket  After=docker.service  [Service]  User=root  PermissionsStartOnly=true  ExecStart=/usr/bin/docker run -p 127.0.0.1:6443:6443 \\  -v /etc/nginx:/etc/nginx \\  --name nginx-proxy \\  --net=host \\  --restart=on-failure:5 \\  --memory=512M \\  nginx:1.13.3-alpine  ExecStartPre=-/usr/bin/docker rm -f nginx-proxy  ExecStop=/usr/bin/docker stop nginx-proxy  Restart=always  RestartSec=15s  TimeoutStartSec=30s  [Install]  WantedBy=multi-user.target  EOF  # 设置开机启动  systemctl daemon-reload && systemctl enable nginx-proxy && systemctl start nginx-proxy && systemctl status nginx-proxy |