《Docker环境下的前后端分离部署与运维》课程脚本

Docker虚拟机常用命令

1. 先更新软件包

```
yum -y update
```

2. 安装Docker虚拟机

```
yum install -y docker
```

3. 运行、重启、关闭Docker虚拟机

```
service docker start
service docker start
service docker stop
```

4. 搜索镜像

```
docker search 镜像名称
```

5. 下载镜像

```
docker pull 镜像名称
```

6. 查看镜像

```
docker images
```

7. 删除镜像

```
docker rmi 镜像名称
```

8. 运行容器

```
docker run 启动参数  镜像名称
```

9. 查看容器列表

```
docker ps -a
```

10. 停止、挂起、恢复容器

```
docker stop 容器ID
docker pause 容器ID
docker unpase 容器ID
```

11. 查看容器信息

```
docker inspect 容器ID
```

12. 删除容器

```
docker rm 容器ID
```

13. 数据卷管理

```
docker volume create 数据卷名称  #创建数据卷
docker volume rm 数据卷名称  #删除数据卷
docker volume inspect 数据卷名称  #查看数据卷
```

14. 网络管理

```
docker network ls 查看网络信息
docker network create --subnet=网段 网络名称
docker network rm 网络名称
```

15. 避免VM虚拟机挂起恢复之后，Docker虚拟机断网

```
vi /etc/sysctl.conf
```

文件中添加`net.ipv4.ip_forward=1`这个配置

```shell
#重启网络服务
systemctl  restart network
```

安装PXC集群，负载均衡，双机热备

1. 安装PXC镜像

   ```
   docker pull percona/percona-xtradb-cluster
   ```

2. 为PXC镜像改名

   ```
   docker tag percona/percona-xtradb-cluster pxc
   ```

3. 创建net1网段

   ```
   docker network create --subnet=172.18.0.0/16 net1
   ```

4. 创建5个数据卷

   ```
   docker volume create --name v1
   docker volume create --name v2
   docker volume create --name v3
   docker volume create --name v4
   docker volume create --name v5
   ```

5. 创建备份数据卷（用于热备份数据）

   ```
   docker volume create --name backup
   ```

6. 创建5节点的PXC集群

   注意，每个MySQL容器创建之后，因为要执行PXC的初始化和加入集群等工作，耐心等待1分钟左右再用客户端连接MySQL。另外，必须第1个MySQL节点启动成功，用MySQL客户端能连接上之后，再去创建其他MySQL节点。

```
#创建第1个MySQL节点
docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=abc123456 -e CLUSTER_NAME=PXC -e
XTRABACKUP_PASSWORD=abc123456 -v v1:/var/lib/mysql -v backup:/data --privileged --
name=node1 --net=net1 --ip 172.18.0.2 pxc
#创建第2个MySQL节点
docker run -d -p 3307:3306 -e MYSQL_ROOT_PASSWORD=abc123456 -e CLUSTER_NAME=PXC -e
XTRABACKUP_PASSWORD=abc123456 -e CLUSTER_JOIN=node1 -v v2:/var/lib/mysql -v
backup:/data --privileged --name=node2 --net=net1 --ip 172.18.0.3 pxc
#创建第3个MySQL节点
docker run -d -p 3308:3306 -e MYSQL_ROOT_PASSWORD=abc123456 -e CLUSTER_NAME=PXC -e
XTRABACKUP_PASSWORD=abc123456 -e CLUSTER_JOIN=node1 -v v3:/var/lib/mysql --
privileged --name=node3 --net=net1 --ip 172.18.0.4 pxc
#创建第4个MySQL节点
docker run -d -p 3309:3306 -e MYSQL_ROOT_PASSWORD=abc123456 -e CLUSTER_NAME=PXC -e
XTRABACKUP_PASSWORD=abc123456 -e CLUSTER_JOIN=node1 -v v4:/var/lib/mysql --
privileged --name=node4 --net=net1 --ip 172.18.0.5 pxc
#创建第5个MySQL节点
docker run -d -p 3310:3306 -e MYSQL_ROOT_PASSWORD=abc123456 -e CLUSTER_NAME=PXC -e
XTRABACKUP_PASSWORD=abc123456 -e CLUSTER_JOIN=node1 -v v5:/var/lib/mysql -v
backup:/data --privileged --name=node5 --net=net1 --ip 172.18.0.6 pxc
```

7. 安装Haproxy镜像

```
docker pull haproxy
```

8. 宿主机上编写Haproxy配置文件

```
vi /home/soft/haproxy.cfg
```

配置文件如下：

```
global
    #工作目录
    chroot /usr/local/etc/haproxy
    #日志文件，使用rsyslog服务中local5日志设备（/var/log/local5），等级info
    log 127.0.0.1 local5 info
    #守护进程运行
    daemon

defaults

    log global
```

```
    mode    http
    #日志格式
    option  httplog
    #日志中不记录负载均衡的心跳检测记录
    option  dontlognull
    #连接超时（毫秒）
    timeout connect 5000
    #客户端超时（毫秒）
    timeout client  50000
    #服务器超时（毫秒）
    timeout server  50000

#监控界面
listen  admin_stats
    #监控界面的访问的IP和端口
    bind  0.0.0.0:8888
    #访问协议
    mode        http
    #URI相对地址
    stats uri   /dbs
    #统计报告格式
    stats realm     Global\ statistics
    #登陆帐户信息
    stats auth  admin:abc123456
#数据库负载均衡
listen  proxy-mysql
    #访问的IP和端口
    bind  0.0.0.0:3306
    #网络协议
    mode  tcp
    #负载均衡算法（轮询算法）
    #轮询算法：roundrobin
    #权重算法：static-rr
    #最少连接算法：leastconn
    #请求源IP算法：source
    balance  roundrobin
    #日志格式
    option  tcplog
    #在MySQL中创建一个没有权限的haproxy用户，密码为空。Haproxy使用这个账户对MySQL数据库心跳检测
    option  mysql-check user haproxy
    server  MySQL_1 172.18.0.2:3306 check weight 1 maxconn 2000
```

```
    server  MySQL_2 172.18.0.3:3306 check weight 1 maxconn 2000
    server  MySQL_3 172.18.0.4:3306 check weight 1 maxconn 2000
    server  MySQL_4 172.18.0.5:3306 check weight 1 maxconn 2000
    server  MySQL_5 172.18.0.6:3306 check weight 1 maxconn 2000
    #使用keepalive检测死链
    option  tcpka
```

9. 创建两个Haproxy容器

```
#创建第1个Haproxy负载均衡服务器
docker run -it -d -p 4001:8888 -p 4002:3306 -v
/home/soft/haproxy:/usr/local/etc/haproxy --name h1 --privileged --net=net1 --ip
172.18.0.7 haproxy
#进入h1容器，启动Haproxy
docker exec -it h1 bash
haproxy -f /usr/local/etc/haproxy/haproxy.cfg
#创建第2个Haproxy负载均衡服务器
docker run -it -d -p 4003:8888 -p 4004:3306 -v
/home/soft/haproxy:/usr/local/etc/haproxy --name h2 --privileged --net=net1 --ip
172.18.0.8 haproxy
#进入h2容器，启动Haproxy
docker exec -it h2 bash
haproxy -f /usr/local/etc/haproxy/haproxy.cfg
```

10. Haproxy容器内安装Keepalived，设置虚拟IP

```
#进入h1容器
docker exec -it h1 bash
#更新软件包
apt-get update
#安装VIM
apt-get install vim
#安装Keepalived
apt-get install keepalived
#编辑Keepalived配置文件（参考下方配置文件）
vim /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
#宿主机执行ping命令
ping 172.18.0.201
```

配置文件内容如下：

```
vrrp_instance  VI_1 {
    state  MASTER
    interface  eth0
    virtual_router_id  51
    priority  100
    advert_int  1
    authentication {
        auth_type  PASS
        auth_pass  123456
    }
    virtual_ipaddress {
        172.18.0.201
    }
}
```

```
#进入h2容器
docker exec -it h2 bash
#更新软件包
apt-get update
#安装VIM
apt-get install vim
#安装Keepalived
apt-get install keepalived
#编辑Keepalived配置文件
vim /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
#宿主机执行ping命令
ping 172.18.0.201
```

配置文件内容如下：

```
vrrp_instance  VI_1 {
    state  MASTER
    interface  eth0
    virtual_router_id  51
    priority  100
    advert_int  1

    authentication {
```

```
        auth_type  PASS
        auth_pass  123456
    }
    virtual_ipaddress {
        172.18.0.201
    }
}
```

11. 宿主机安装Keepalived，实现双击热备

```
#宿主机执行安装Keepalived
yum -y install keepalived
#修改Keepalived配置文件
vi /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
```

Keepalived配置文件如下：

```
vrrp_instance VI_1 {
    state MASTER
    interface ens33
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.99.150
    }
}

virtual_server 192.168.99.150 8888 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP

    real_server 172.18.0.201 8888 {
```

```
        weight 1
    }
}

virtual_server 192.168.99.150 3306 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP

    real_server 172.18.0.201 3306 {
        weight 1
    }
}
```

12. 热备份数据

```
#进入node1容器
docker exec -it node1 bash
#更新软件包
apt-get update
#安装热备工具
apt-get install percona-xtrabackup-24
#全量热备
innobackupex --user=root --password=abc123456 /data/backup/full
```

13. 冷还原数据 停止其余4个节点，并删除节点

```
docker stop node2
docker stop node3
docker stop node4
docker stop node5
docker rm node2
docker rm node3
docker rm node4
docker rm node5
```

node1容器中删除MySQL的数据

```
#删除数据
rm -rf /var/lib/mysql/*
#清空事务
innobackupex --user=root --password=abc123456 --apply-back /data/backup/full/2018-04-15_05-09-07/
#还原数据
innobackupex --user=root --password=abc123456 --copy-back  /data/backup/full/2018-04-15_05-09-07/
```

重新创建其余4个节点，组件PXC集群

安装Redis，配置RedisCluster集群

1. 安装Redis镜像

```
docker pull yyyyttttwwww/redis
```

2. 创建net2网段

```
docker network create --subnet=172.19.0.0/16 net2
```

3. 创建6节点Redis容器

```
docker run -it -d --name r1 -p 5001:6379 --net=net2 --ip 172.19.0.2 redis bash
docker run -it -d --name r2 -p 5002:6379 --net=net2 --ip 172.19.0.3 redis bash
docker run -it -d --name r3 -p 5003:6379 --net=net2 --ip 172.19.0.4 redis bash
docker run -it -d --name r4 -p 5004:6379 --net=net2 --ip 172.19.0.5 redis bash
docker run -it -d --name r5 -p 5005:6379 --net=net2 --ip 172.19.0.6 redis bash
```

4. 启动6节点Redis服务器

```
#进入r1节点
docker exec -it r1 bash
cp /home/redis/redis.conf /usr/redis/redis.conf
cd /usr/redis/src
./redis-server ../redis.conf
#进入r2节点
docker exec -it r2 bash
cp /home/redis/redis.conf /usr/redis/redis.conf

cd /usr/redis/src
```

```
./redis-server ../redis.conf
#进入r3节点
docker exec -it r3 bash
cp /home/redis/redis.conf /usr/redis/redis.conf
cd /usr/redis/src
./redis-server ../redis.conf
#进入r4节点
docker exec -it r4 bash
cp /home/redis/redis.conf /usr/redis/redis.conf
cd /usr/redis/src
./redis-server ../redis.conf
#进入r5节点
docker exec -it r5 bash
cp /home/redis/redis.conf /usr/redis/redis.conf
cd /usr/redis/src
./redis-server ../redis.conf
#进入r6节点
docker exec -it r6 bash
cp /home/redis/redis.conf /usr/redis/redis.conf
cd /usr/redis/src
./redis-server ../redis.conf
```

5. 创建Cluster集群

```
#在r1节点上执行下面的指令
cd /usr/redis/src
mkdir -p ../cluster
cp redis-trib.rb ../cluster/
cd ../cluster
#创建Cluster集群
./redis-trib.rb create --replicas 1 172.19.0.2:6379 172.19.0.3:6379
172.19.0.4:6379 172.19.0.5:6379 172.19.0.6:6379 172.19.0.7:6379
```

打包部署后端项目

1. 进入人人开源后端项目，执行打包（修改配置文件，更改端口，打包三次生成三个JAR文件）

```
mvn clean install -Dmaven.test.skip=true
```

2. 安装Java镜像

```
docker pull java
```

3. 创建3节点Java容器

```
#创建数据卷，上传JAR文件
docker volume create j1
#启动容器
docker run -it -d --name j1 -v j1:/home/soft --net=host java
#进入j1容器
docker exec -it j1 bash
#启动Java项目
nohup java -jar /home/soft/renren-fast.jar

#创建数据卷，上传JAR文件
docker volume create j2
#启动容器
docker run -it -d --name j2 -v j2:/home/soft --net=host java
#进入j1容器
docker exec -it j2 bash
#启动Java项目
nohup java -jar /home/soft/renren-fast.jar

#创建数据卷，上传JAR文件
docker volume create j3
#启动容器
docker run -it -d --name j3 -v j3:/home/soft --net=host java
#进入j1容器
docker exec -it j3 bash
#启动Java项目
nohup java -jar /home/soft/renren-fast.jar
```

4. 安装Nginx镜像

```
docker pull nginx
```

5. 创建Nginx容器，配置负载均衡

宿主机上/home/n1/nginx.conf配置文件内容如下：

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
```

```
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    proxy_redirect          off;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    client_max_body_size    10m;
    client_body_buffer_size   128k;
    proxy_connect_timeout   5s;
    proxy_send_timeout      5s;
    proxy_read_timeout      5s;
    proxy_buffer_size        4k;
    proxy_buffers           4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

    upstream tomcat {
        server 192.168.99.104:6001;
        server 192.168.99.104:6002;
        server 192.168.99.104:6003;
    }
```

```
    server {
        listen       6101;
        server_name  192.168.99.104;
        location / {
            proxy_pass   http://tomcat;
            index  index.html index.htm;
        }
    }
}
```

创建第1个Nginx节点

```
docker run -it -d --name n1 -v /home/n1/nginx.conf:/etc/nginx/nginx.conf --
net=host --privileged nginx
```

宿主机上/home/n2/nginx.conf配置文件内容如下：

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;
```

```
    #gzip  on;

    proxy_redirect          off;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    client_max_body_size    10m;
    client_body_buffer_size   128k;
    proxy_connect_timeout   5s;
    proxy_send_timeout      5s;
    proxy_read_timeout      5s;
    proxy_buffer_size        4k;
    proxy_buffers           4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

    upstream tomcat {
        server 192.168.99.104:6001;
        server 192.168.99.104:6002;
        server 192.168.99.104:6003;
    }
    server {
        listen       6102;
        server_name  192.168.99.104;
        location / {
            proxy_pass   http://tomcat;
            index  index.html index.htm;
        }
    }
}
```

创建第2个Nginx节点

```
docker run -it -d --name n2 -v /home/n2/nginx.conf:/etc/nginx/nginx.conf --
net=host --privileged nginx
```

6. 在Nginx容器安装Keepalived

```
#进入n1节点
docker exec -it n1 bash
#更新软件包
apt-get update
#安装VIM
apt-get install vim
#安装Keepalived
apt-get install keepalived
#编辑Keepalived配置文件(如下)
vim /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
```

```
vrrp_instance VI_1 {
    state MASTER
    interface ens33
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.99.151
    }
}
virtual_server 192.168.99.151 6201 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP
    real_server 192.168.99.104 6101 {
        weight 1
    }
}
```

```
#进入n1节点
docker exec -it n2 bash
#更新软件包
apt-get update
#安装VIM
apt-get install vim
#安装Keepalived
apt-get install keepalived
#编辑Keepalived配置文件(如下)
vim /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
```

```
vrrp_instance VI_1 {
    state MASTER
    interface ens33
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.99.151
    }
}
virtual_server 192.168.99.151 6201 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP
    real_server 192.168.99.104 6102 {
        weight 1
    }
}
```

打包部署后端项目

1. 在前端项目路径下执行打包指令

```
npm run build
```

2. build目录的文件拷贝到宿主机的/home/fn1/renren-vue、/home/fn2/renren-vue、/home/fn3/renren-vue的目录下面
3. 创建3节点的Nginx，部署前端项目

   宿主机/home/fn1/nginx.conf的配置文件

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

        proxy_redirect          off;
        proxy_set_header        Host $host;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size    10m;
        client_body_buffer_size   128k;
```

```
        proxy_connect_timeout   5s;
        proxy_send_timeout      5s;
        proxy_read_timeout      5s;
        proxy_buffer_size        4k;
        proxy_buffers           4 32k;
        proxy_busy_buffers_size  64k;
        proxy_temp_file_write_size 64k;

        server {
                listen 6501;
                server_name  192.168.99.104;
                location  /  {
                        root  /home/fn1/renren-vue;
                        index  index.html;
                }
        }
}
```

```
#启动第fn1节点
docker run -it -d --name fn1 -v /home/fn1/nginx.conf:/etc/nginx/nginx.conf -v
/home/fn1/renren-vue:/home/fn1/renren-vue --privileged --net=host nginx
```

宿主机/home/fn2/nginx.conf的配置文件

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
```

```
    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    proxy_redirect          off;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    client_max_body_size    10m;
    client_body_buffer_size   128k;
    proxy_connect_timeout   5s;
    proxy_send_timeout      5s;
    proxy_read_timeout      5s;
    proxy_buffer_size        4k;
    proxy_buffers           4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

    server {
        listen 6502;
        server_name  192.168.99.104;
        location  /  {
            root  /home/fn2/renren-vue;
            index  index.html;
        }
    }
}
```

```
#启动第fn2节点
docker run -it -d --name fn2 -v /home/fn2/nginx.conf:/etc/nginx/nginx.conf -v
/home/fn2/renren-vue:/home/fn2/renren-vue --privileged --net=host nginx
```

宿主机/home/fn3/nginx.conf的配置文件

```
user  nginx;

worker_processes  1;
```

```
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    proxy_redirect          off;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    client_max_body_size    10m;
    client_body_buffer_size   128k;
    proxy_connect_timeout   5s;
    proxy_send_timeout      5s;
    proxy_read_timeout      5s;
    proxy_buffer_size        4k;
    proxy_buffers           4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

    server {
        listen 6503;
        server_name  192.168.99.104;
        location  /  {
```

```
            root   /home/fn3/renren-vue;
            index  index.html;
        }
    }
}
```

启动fn3节点

```
#启动第fn3节点
docker run -it -d --name fn3 -v /home/fn3/nginx.conf:/etc/nginx/nginx.conf -v
/home/fn3/renren-vue:/home/fn3/renren-vue --privileged --net=host nginx
```

4. 配置负载均衡

宿主机/home/ff1/nginx.conf配置文件

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;
```

```
    proxy_redirect          off;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    client_max_body_size    10m;
    client_body_buffer_size   128k;
    proxy_connect_timeout   5s;
    proxy_send_timeout      5s;
    proxy_read_timeout      5s;
    proxy_buffer_size        4k;
    proxy_buffers           4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

    upstream fn {
        server 192.168.99.104:6501;
        server 192.168.99.104:6502;
        server 192.168.99.104:6503;
    }
    server {
        listen       6601;
        server_name  192.168.99.104;
        location / {
            proxy_pass   http://fn;
            index  index.html index.htm;
        }
    }
}
```

```
#启动ff1节点
docker run -it -d --name ff1 -v /home/ff1/nginx.conf:/etc/nginx/nginx.conf --
net=host --privileged nginx
```

宿主机/home/ff2/nginx.conf配置文件

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
```

```
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    proxy_redirect          off;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    client_max_body_size    10m;
    client_body_buffer_size   128k;
    proxy_connect_timeout   5s;
    proxy_send_timeout      5s;
    proxy_read_timeout      5s;
    proxy_buffer_size        4k;
    proxy_buffers           4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

    upstream fn {
        server 192.168.99.104:6501;
        server 192.168.99.104:6502;
        server 192.168.99.104:6503;
    }
    server {
        listen       6602;
        server_name  192.168.99.104;
```

```
        location / {
            proxy_pass   http://fn;
            index  index.html index.htm;
        }
    }
}
```

```
#启动ff2节点
docker run -it -d --name ff2 -v /home/ff2/nginx.conf:/etc/nginx/nginx.conf --
net=host --privileged nginx
```

5. 配置双机热备

```
#进入ff1节点
docker exec -it ff1 bash
#更新软件包
apt-get update
#安装VIM
apt-get install vim
#安装Keepalived
apt-get install keepalived
#编辑Keepalived配置文件(如下)
vim /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
```

```
vrrp_instance VI_1 {
    state MASTER
    interface ens33
    virtual_router_id 52
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.99.152
    }
}
virtual_server 192.168.99.151 6701 {
```

```
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP
    real_server 192.168.99.104 6601 {
        weight 1
    }
}
```

```
#进入ff1节点
docker exec -it ff2 bash
#更新软件包
apt-get update
#安装VIM
apt-get install vim
#安装Keepalived
apt-get install keepalived
#编辑Keepalived配置文件(如下)
vim /etc/keepalived/keepalived.conf
#启动Keepalived
service keepalived start
```

```
vrrp_instance VI_1 {
    state MASTER
    interface ens33
    virtual_router_id 52
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.99.152
    }
}
virtual_server 192.168.99.151 6701 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
```

```
    persistence_timeout 50
    protocol TCP
    real_server 192.168.99.104 6602 {
        weight 1
    }
}
```