**Redis集群**

**1. 集群配置**

3主+3从

由于仅用于测试，故我这里只用1台服务器进行模拟

redis列表



**2.编写redis.conf**

在server上创建一个目录用于存放redis集群部署文件。这里我放的路径为/data/test/redis/

在/data/test/redis/目录下创建redis-1,redis-2,redis-3,redis-4,redis-5,redis-6文件夹:

mkdir -p /opt/docker/redis-cluster/{redis-1,redis-2,redis-3,redis-4,redis-5,redis-6}

#创建持久化目录

mkdir -p /opt/docker/redis-cluster/redis-1/data

mkdir -p /opt/docker/redis-cluster/redis-2/data

mkdir -p /opt/docker/redis-cluster/redis-3/data

mkdir -p /opt/docker/redis-cluster/redis-4/data

mkdir -p /opt/docker/redis-cluster/redis-5/data

mkdir -p /opt/docker/redis-cluster/redis-6/data

**3. 在每个redis-\*文件夹下创建redis.conf文件，并写入如下内容:**

cluster-enabled yes # 开启集群

cluster-config-file nodes.conf # 集群配置文件

cluster-node-timeout 5000 # 集群节点多少时间未响应视为该节点丢失

appendonly yes

port 6379 # redis监听端口

masterauth passwd123 #设置master节点密码

requirepass passwd123 #设置密码

注意：port值不能都为6379，根据上面redis列表设置的端口号，依次给redis-1 ~ redis-6设置6379~6384端口号

**4.编写docker-compose.yml文件**

这里直接使用daocloud提供的redis镜像地址daocloud.io/library/redis:6.0.4

在/root/redis-cluster文件夹下创建docker-compose.yml文件。

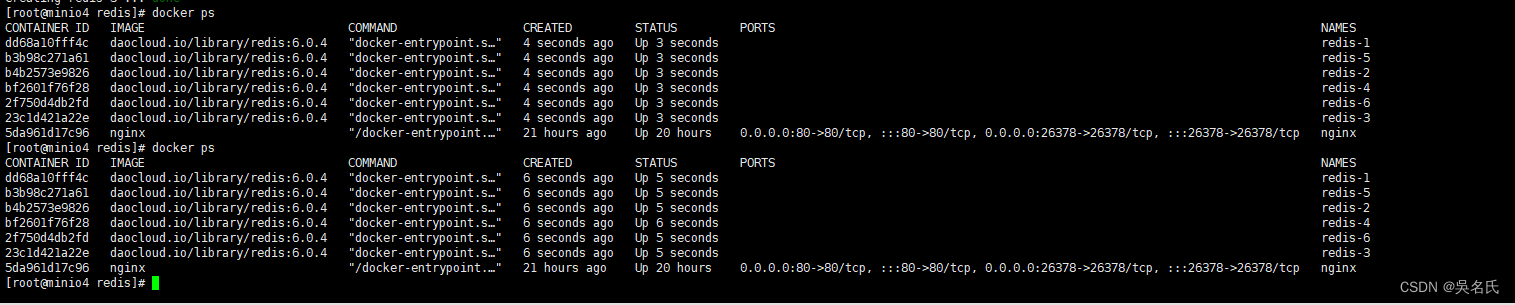
docker-compose.yml文件内容如下：

version: '3.1'  
services:  
 # redis1配置  
redis1:  
image: daocloud.io/library/redis:6.0.4  
container\_name: redis-1  
restart: always  
network\_mode: "host"  
volumes:  
 - /data/test/redis/redis-1/data:/data  
 - /data/test/redis/redis-1/redis.conf:/usr/local/etc/redis/redis.conf  
command: ["redis-server", "/usr/local/etc/redis/redis.conf"]  
 # redis2配置  
redis2:  
image: daocloud.io/library/redis:6.0.4  
container\_name: redis-2  
restart: always  
network\_mode: "host"  
volumes:  
 - /data/test/redis/redis-2/data:/data  
 - /data/test/redis/redis-2/redis.conf:/usr/local/etc/redis/redis.conf  
command: ["redis-server", "/usr/local/etc/redis/redis.conf"]  
 # redis3配置  
redis3:  
image: daocloud.io/library/redis:6.0.4  
container\_name: redis-3  
restart: always  
network\_mode: "host"  
volumes:  
 - /data/test/redis/redis-3/data:/data  
 - /data/test/redis/redis-3/redis.conf:/usr/local/etc/redis/redis.conf  
command: ["redis-server", "/usr/local/etc/redis/redis.conf"]  
 # redis4配置  
redis4:  
image: daocloud.io/library/redis:6.0.4  
container\_name: redis-4  
restart: always  
network\_mode: "host"  
volumes:  
 - /data/test/redis/redis-4/data:/data  
 - /data/test/redis/redis-4/redis.conf:/usr/local/etc/redis/redis.conf  
command: ["redis-server", "/usr/local/etc/redis/redis.conf"]  
 # redis5配置  
redis5:  
image: daocloud.io/library/redis:6.0.4  
container\_name: redis-5  
restart: always  
network\_mode: "host"  
volumes:  
 - /data/test/redis/redis-5/data:/data  
 - /data/test/redis/redis-5/redis.conf:/usr/local/etc/redis/redis.conf  
command: ["redis-server", "/usr/local/etc/redis/redis.conf"]  
 # redis6配置  
redis6:  
image: daocloud.io/library/redis:6.0.4  
container\_name: redis-6  
restart: always  
network\_mode: "host"  
volumes:  
 - /data/test/redis/redis-6/data:/data  
 - /data/test/redis/redis-6/redis.conf:/usr/local/etc/redis/redis.conf  
command: ["redis-server", "/usr/local/etc/redis/redis.conf"]

启动容器,执行命令:

#启动容器

docker-compose -f xxx.yaml up -d



**5.开启集群**

随便找一个容器进入，这里我选择redis-1进入。 在进入容器后，输入如下命令开启集群:

**redis-cli --cluster create 192.168.66.180:6379 \**

**192.168.66.180:6380 \**

**192.168.66.180:6381 \**

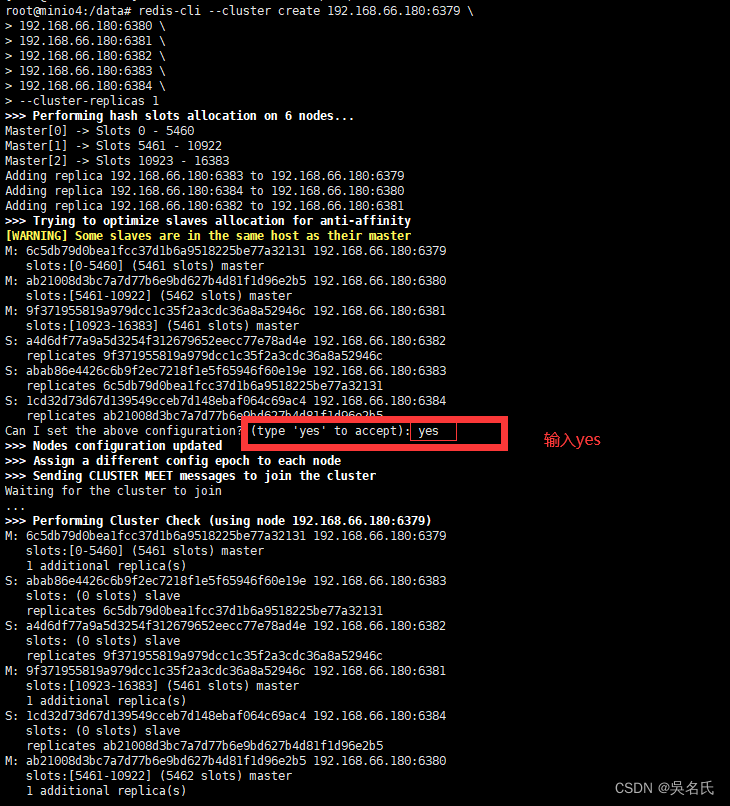
**192.168.66.180:6382 \**

**192.168.66.180:6383 \**

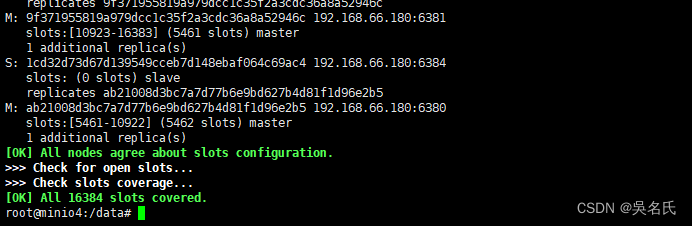
**192.168.66.180:6384 \**

**--cluster-replicas 1**

**注意：自己服务器的ip**

****

**出现下图即为成功。**

****

**Kafka集群**

创建文件：docker-compose-zookeepers-kafkas.yml

注意：为了能够Java连接到容器中的kafka，在每个kafka服务中我没有设置其他协议别名，就是单独使用了一个PLAINTEXT，并且每个服务都各自映射自己端口，而不是之前的8083:8082，8084:8082。

我自己的话也试过定义多个协议名方案不过都不行，只能最终方案选择这个。

准备操作：查看你的公网ip地址或者你的虚拟机ip地址。

拿到这个ip地址后去修改下面配置中对应的ip地址，只有这样你的java程序才能够发送消息成功并接收！

version: '3.1'

networks:

zk-net: # 网络名

name: zk-net

driver: bridge

services:

# zookeeper集群

zoo1:

image: zookeeper:3.8.0

container\_name: zoo1 # 容器名称

restart: always # 开机自启

hostname: zoo1 # 主机名

ports:

- 2181:2181 # 端口号

environment:

ZOO\_MY\_ID: 1

ZOO\_SERVERS: server.1=zoo1:2888:3888;2181 server.2=zoo2:2888:3888;2181 server.3=zoo3:2888:3888;2181

networks:

- zk-net

zoo2:

image: zookeeper:3.8.0

container\_name: zoo2

restart: always

hostname: zoo2

ports:

- 2182:2181

environment:

ZOO\_MY\_ID: 2

ZOO\_SERVERS: server.1=zoo1:2888:3888;2181 server.2=zoo2:2888:3888;2181 server.3=zoo3:2888:3888;2181

networks:

- zk-net

zoo3:

image: zookeeper:3.8.0

container\_name: zoo3

restart: always

hostname: zoo3

ports:

- 2183:2181

environment:

ZOO\_MY\_ID: 3

ZOO\_SERVERS: server.1=zoo1:2888:3888;2181 server.2=zoo2:2888:3888;2181 server.3=zoo3:2888:3888;2181

networks:

- zk-net

# kafka集群

kafka1:

image: 'bitnami/kafka:3.0.0'

container\_name: kafka1

hostname: kafka1

networks:

- zk-net

ports:

- '9092:9092'

environment:

- KAFKA\_CFG\_ZOOKEEPER\_CONNECT=zoo1:2181,zoo2:2182,zoo3:2183/kafka

- KAFKA\_BROKER\_ID=1

- KAFKA\_CFG\_LISTENERS=PLAINTEXT://:9092

- KAFKA\_CFG\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.10.105:9092

- ALLOW\_PLAINTEXT\_LISTENER=yes

depends\_on:

- zoo1

- zoo2

- zoo3

kafka2:

image: 'bitnami/kafka:3.0.0'

container\_name: kafka2

hostname: kafka2

networks:

- zk-net

ports:

- '9093:9093'

environment:

- KAFKA\_CFG\_ZOOKEEPER\_CONNECT=zoo1:2181,zoo2:2182,zoo3:2183/kafka

- KAFKA\_BROKER\_ID=2

- KAFKA\_CFG\_LISTENERS=PLAINTEXT://:9093

- KAFKA\_CFG\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.10.105:9093

- ALLOW\_PLAINTEXT\_LISTENER=yes

depends\_on:

- zoo1

- zoo2

- zoo3

kafka3:

image: 'bitnami/kafka:3.0.0'

container\_name: kafka3

hostname: kafka3

networks:

- zk-net

ports:

- '9094:9094'

environment:

- KAFKA\_CFG\_ZOOKEEPER\_CONNECT=zoo1:2181,zoo2:2182,zoo3:2183/kafka

- KAFKA\_BROKER\_ID=3

- KAFKA\_CFG\_LISTENERS=PLAINTEXT://:9094

- KAFKA\_CFG\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.10.105:9094

- ALLOW\_PLAINTEXT\_LISTENER=yes

depends\_on:

- zoo1

- zoo2

- zoo3

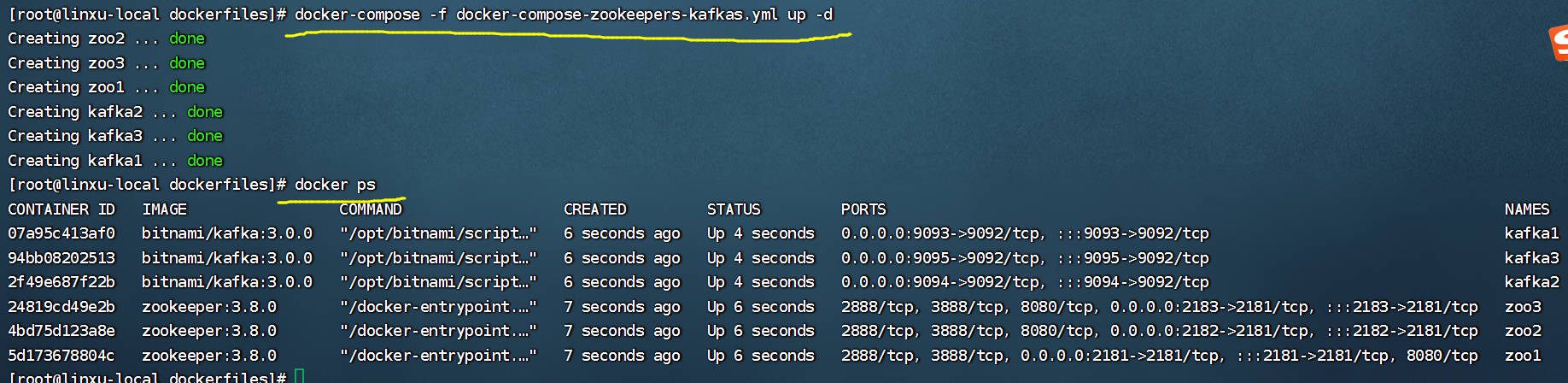
启动：

# -f表示指定某个配置文件名 -d：表示后台启动

docker-compose -f docker-compose-zookeepers-kafkas.yml up -d

# 查看当前服务

docker ps



**部署Prometheus, node-exporter, Grafana**

编写docker-compose文件

version: '2'  
networks:  
monitor:  
driver: bridge  
  
services:  
prometheus:  
image: prom/prometheus  
container\_name: prometheus  
hostname: prometheus  
restart: always  
volumes:  
 - /data/test/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml  
 - /data/test/prometheus/node\_down.yml:/etc/prometheus/node\_down.yml  
ports:  
 - "9090:9090"  
networks:  
 - monitor  
  
alertmanager:  
image: prom/alertmanager  
container\_name: alertmanager  
hostname: alertmanager  
restart: always  
volumes:  
 - /data/test/alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.yml  
ports:  
 - "9093:9093"  
networks:  
 - monitor  
  
grafana:  
image: grafana/grafana  
container\_name: grafana  
hostname: grafana  
restart: always  
ports:  
 - "3000:3000"  
networks:  
 - monitor  
  
node-exporter:  
image: quay.io/prometheus/node-exporter  
container\_name: node-exporter  
hostname: node-exporter  
restart: always  
ports:  
 - "9100:9100"  
networks:  
 - monitor  
  
cadvisor:  
image: google/cadvisor:latest  
container\_name: cadvisor  
hostname: cadvisor  
restart: always  
volumes:  
 - /:/rootfs:ro  
 - /var/run:/var/run:rw  
 - /sys:/sys:ro  
 - /var/lib/docker/:/var/lib/docker:ro  
ports:  
 - "8080:8080"  
networks:  
 - monitor

#启动容器： docker-compose -f xxxx up -d

nginx+keepalived实现高可用 Web 负载均衡

安装nginx

下载nginx-1.12.2.tar.gz

进入你上传的目录下解压

tar -zxvf nginx-1.12.2.tar.gz

解压完了编译安装

在你解压的nginx目录下执行

./configure --prefix=/usr/local/nginx

make && make install

配置 Nginx

user root;

worker\_processes 1;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include 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 logs/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

server {

listen 88;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root html;

index index.html index.htm;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

}

验证nginx是否安装成功

# /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

启动 Nginx

# /usr/local/nginx/sbin/nginx

设置开机自启动

设置 Nginx 开机启动

# vi /etc/rc.local

加入：

/usr/local/nginx/sbin/nginx

安装keepalived

上传keepalived-2.2.8.tar.gz文件

解压：

tar -zxvf keepalived-2.2.8.tar.gz

解压完了进入目录，并完成配置安装

/usr/local/keepalived/

./configure --prefix=/usr/local/keepalived

等待配置进行安装完后，运行真正的安装

make && make install

复制文件到/etc/keepalived/

(keepalived默认是读取/etc/keepalived/下的keepalived.conf)先创建目录

mkdir /etc/keepalived/

#复制你的Keepalived目录下的文件，对应的路径下的文件，不同版本文件路径不一样

cp /usr/local/keepalived/keepalived-2.2.8/keepalived/etc/init.d/keepalived /etc/init.d/

cp /usr/local/keepalived/keepalived-2.2.8/keepalived/etc/keepalived/keepalived.conf.sample /etc/keepalived/

cp /usr/local/keepalived/keepalived-2.2.8/keepalived/etc/sysconfig//keepalived /etc/sysconfig/

cp /usr/local/sbin/keepalived /usr/sbin/

就三点不同，一点必须相同，

1. router\_id 不同，

2. state BACKUP不同（主节点为 MASTER， 对应的备份节点为 BACKUP） ，

3. priority不同（节点优先级， 值范围 0-254， MASTER 要比 BACKUP 高）。 4.virtual\_router\_id 必相同。 配置如下：

MASTER keepalived.conf文件：

! Configuration File for keepalived

global\_defs {

## keepalived 自带的邮件提醒需要开启 sendmail 服务。 建议用独立的监控或第三方 SMTP

router\_id 192.168.1.160 ## 标识本节点的字条串，通常为 hostname

}

## keepalived 会定时执行脚本并对脚本执行的结果进行分析，动态调整 vrrp\_instance 的优先级。如果脚本执行结果为 0，并且 weight 配置的值大于 0，则优先级相应的增加。如果脚本执行结果非 0，并且 weight配置的值小于 0，则优先级相应的减少。其他情况，维持原本配置的优先级，即配置文件中 priority 对应的值。

vrrp\_script chk\_nginx {

script "/etc/keepalived/nginx\_check.sh" ## 检测 nginx 状态的脚本路径

interval 2 ## 检测时间间隔

weight -20 ## 如果条件成立，权重-20

}

## 定义虚拟路由， VI\_1 为虚拟路由的标示符，自己定义名称

vrrp\_instance VI\_1 {

state MASTER ## 主节点为 MASTER， 对应的备份节点为 BACKUP

interface ens33 ## 绑定虚拟 IP 的网络接口，与本机 IP 地址所在的网络接口相同， 我的是 eth0

virtual\_router\_id 160 ## 虚拟路由的 ID 号， 两个节点设置必须一样， 可选 IP 最后一段使用, 相同的 VRID 为一个组，他将决定多播的 MAC 地址

mcast\_src\_ip 192.168.1.160 ## 本机 IP 地址

priority 100 ## 节点优先级， 值范围 0-254， MASTER 要比 BACKUP 高

nopreempt ## 优先级高的设置 nopreempt 解决异常恢复后再次抢占的问题

advert\_int 1 ## 组播信息发送间隔，两个节点设置必须一样， 默认 1s

## 设置验证信息，两个节点必须一致

authentication {

auth\_type PASS

auth\_pass 1111 ## 真实生产，按需求对应该过来

}

## 将 track\_script 块加入 instance 配置块

track\_script {

chk\_nginx ## 执行 Nginx 监控的服务

} #

# 虚拟 IP 池, 两个节点设置必须一样

virtual\_ipaddress {

192.168.1.161 ## 虚拟 ip，可以定义多个

}

}

BACKUP keepalived.conf文件：

! Configuration File for keepalived

! Configuration File for keepalived

global\_defs {

## keepalived 自带的邮件提醒需要开启 sendmail 服务。 建议用独立的监控或第三方 SMTP

router\_id 192.168.1.131 ## 标识本节点的字条串，通常为 hostname

}

## keepalived 会定时执行脚本并对脚本执行的结果进行分析，动态调整 vrrp\_instance 的优先级。如果脚本执行结果为 0，并且 weight 配置的值大于 0，则优先级相应的增加。如果脚本执行结果非 0，并且 weight配置的值小于 0，则优先级相应的减少。其他情况，维持原本配置的优先级，即配置文件中 priority 对应的值。

vrrp\_script chk\_nginx {

script "/etc/keepalived/nginx\_check.sh" ## 检测 nginx 状态的脚本路径

interval 2 ## 检测时间间隔

weight -20 ## 如果条件成立，权重-20

}

## 定义虚拟路由， VI\_1 为虚拟路由的标示符，自己定义名称

vrrp\_instance VI\_1 {

state BACKUP ## 主节点为 MASTER， 对应的备份节点为 BACKUP

interface eno1 ## 绑定虚拟 IP 的网络接口，与本机 IP 地址所在的网络接口相同， 我的是 eth0

virtual\_router\_id 160## 虚拟路由的 ID 号， 两个节点设置必须一样， 可选 IP 最后一段使用, 相同的 VRID 为一个组，他将决定多播的 MAC 地址

mcast\_src\_ip 192.168.1.160 ## 本机 IP 地址

priority 90 ## 节点优先级， 值范围 0-254， MASTER 要比 BACKUP 高

nopreempt ## 优先级高的设置 nopreempt 解决异常恢复后再次抢占的问题

advert\_int 1 ## 组播信息发送间隔，两个节点设置必须一样， 默认 1s

## 设置验证信息，两个节点必须一致

authentication {

auth\_type PASS

auth\_pass 1111 ## 真实生产，按需求对应该过来

}

## 将 track\_script 块加入 instance 配置块

track\_script {

chk\_nginx ## 执行 Nginx 监控的服务

} #

# 虚拟 IP 池, 两个节点设置必须一样

virtual\_ipaddress {

192.168.1.161 ## 虚拟 ip，可以定义多个

}

}

编写 Nginx 状态检测脚本 nginx\_check.sh（俩个都要）

# vi /etc/keepalived/nginx\_check.sh

#!/bin/bash

A=`ps -C nginx –no-header |wc -l`

if [ $A -eq 0 ];then

/usr/local/nginx/sbin/nginx

sleep 2

if [ `ps -C nginx --no-header |wc -l` -eq 0 ];then

killall keepalived

fi

fi

保存后，给脚本赋执行权限：

# chmod +x /etc/keepalived/nginx\_check.sh

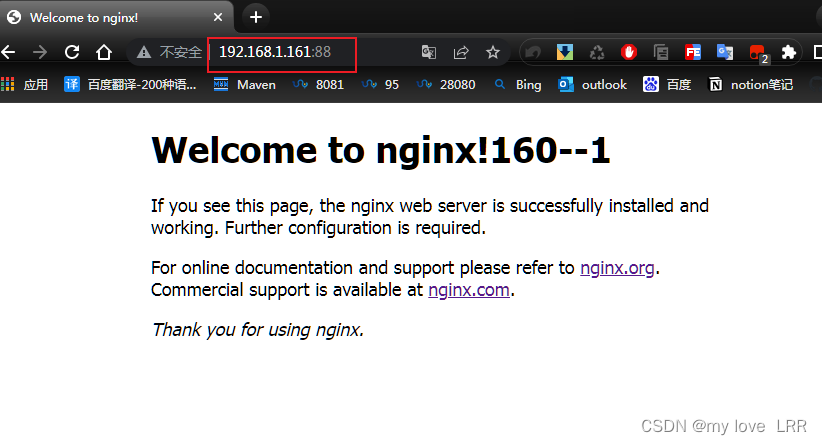
启动keepalived

service keepalived start

查看状态：

service keepalived status

测试keepalived



可以看到地址是虚拟vip，88是nginx监听的。把160的keepalived、nginx挂掉看下会发生什么。

停止keepalived：service keepalived stop。

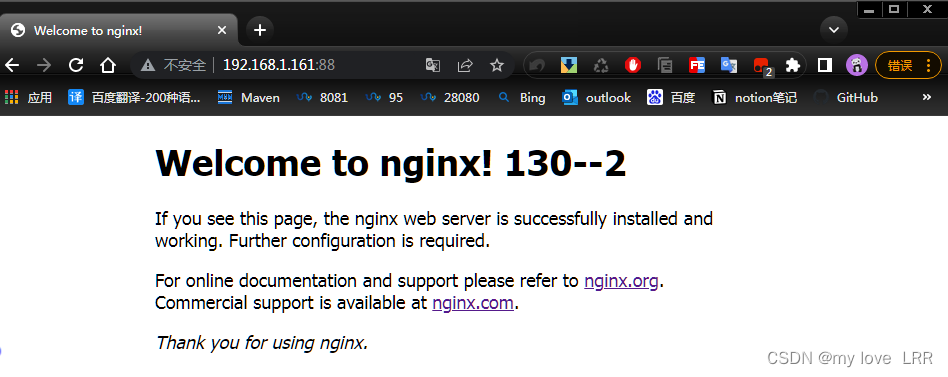
进入nginx目录：cd /usr/local/nginx/sbin。执行

停止keepalived：

service keepalived stop。

停止nginx：

/usr/local/nginx/sbin/nginx -s stop

ip端口还是192.168.1.161:88  


**Linux安装ES集群**

整体步骤

步骤如下:

1、拷贝opt目录下的elasticsearch-7.6.2安装包3个，分别命名:

elasticsearch-7.6.2-icoding1

elasticsearch-7.6.2-icoding2

elasticsearch-7.6.2-icoding3

2、然后修改elasticsearch.yml文件。

3、然后启动启动icoding1、icoding2、icoding3三个节点。

4、打开浏览器输入: http://192.168.189.129:9201/\_cat/health?v如果返回的node.total是3，代表集群搭建成功

在此，需要我们特别注意的是，像本文这样单服务器多节点（3个节点)的情况，仅供测试使用，集群环境如下:

1、拷贝副本

拷贝opt目录下的elasticsearch-7.6.2安装包3个，打开虚拟机到opt目录

执行拷贝三份

cd /opt

cp -r elasticsearch-7.6.2 elasticsearch-7.6.2-icoding1

cp -r elasticsearch-7.6.2 elasticsearch-7.6.2-icoding2

cp -r elasticsearch-7.6.2 elasticsearch-7.6.2-icoding3

2、修改elasticsearch.yml配置文件

1)、创建日志目录

cd /opt

mkdir logs

mkdir data

#授权给icoding用户

chown -R icoding:icoding ./logs

chown -R icoding:icoding ./data

打开elasticsearch.yml配置，分别配置下面三个节点的配置文件

vi /opt/elasticsearch-7.6.2-icoding1/config/elasticsearch.yml

vi /opt/elasticsearch-7.6.2-icoding2/config/elasticsearch.yml

vi /opt/elasticsearch-7.6.2-icoding3/config/elasticsearch.yml

2)、下面是elasticsearch-7.6.2-icoding1配置文件

cluster.name: icoding-course

node.name: icoding-1

node.master: true

node.data: true

node.max\_local\_storage\_nodes: 3

network.host: 0.0.0.0

http.port: 9201

transport.tcp.port: 9700

discovery.seed\_hosts: ["localhost:9700","localhost:9800","localhost:9900"]

cluster.initial\_master\_nodes: ["icoding-1","icoding-2","icoding-3"]

path.data: /opt/data

path.logs: /opt/logs

#如果是CentOS 6版本 加入以下两行

bootstrap.memory\_lock: false

bootstrap.system\_call\_filter: false

下面是elasticsearch-7.6.2-icoding2配置文件

cluster.name: icoding-course

node.name: icoding-2

node.master: true

node.data: true

node.max\_local\_storage\_nodes: 3

network.host: 0.0.0.0

http.port: 9202

transport.tcp.port: 9800

discovery.seed\_hosts: ["localhost:9700","localhost:9800","localhost:9900"]

cluster.initial\_master\_nodes: ["icoding-1","icoding-2","icoding-3"]

path.data: /opt/data

path.logs: /opt/logs

#如果是CentOS 6版本 加入以下两行

bootstrap.memory\_lock: false

bootstrap.system\_call\_filter: false

下面是elasticsearch-7.6.2-icoding3配置文件

cluster.name: icoding-course

node.name: icoding-3

node.master: true

node.data: true

node.max\_local\_storage\_nodes: 3

network.host: 0.0.0.0

http.port: 9203

transport.tcp.port: 9900

discovery.seed\_hosts: ["localhost:9700","localhost:9800","localhost:9900"]

cluster.initial\_master\_nodes: ["icoding-1","icoding-2","icoding-3"]

path.data: /opt/data

path.logs: /opt/logs

#如果是CentOS 6版本 加入以下两行

bootstrap.memory\_lock: false

bootstrap.system\_call\_filter: false

执行授权

在root用户下执行

chown -R icoding:icoding /opt/elasticsearch-7.6.2-icoding1

chown -R icoding:icoding /opt/elasticsearch-7.6.2-icoding2

chown -R icoding:icoding /opt/e1asticsearch-7.6.2-icoding3

如果有的日志文件授权失败，可使用(也是在root下执行)

cd /opt/elasticsearch-7.6.2-icoding1

chown -R icoding:icoding logs

cd /opt/elasticsearch-7.6.2-icoding2

chown -R icoding:icoding logs

cd /opt/elasticsearch-7.6.2-icoding3

chown -R icoding:icoding logs

访问节点

依次启动我们搭建的es（切换为普通用户再启动）

cd /opt/elasticsearch-7.6.2-icoding1/bin/

./elasticsearch

cd /opt/elasticsearch-7.6.2-icoding2/bin/

./elasticsearch

cd /opt/elasticsearch-7.6.2-icoding3/bin/

./elasticsearch

查看集群状态  
打开浏览器输入:<http://192.168.189.129:9201/_cat/health?v>,如果返回的node.total是3，代表集群搭建成功  
