链滴

记录使用 docker 容器实现 redis 主从同步和哨兵模式

作者：Linlx0628

原文链接：https://ld246.com/article/1592552945419

来源网站：链滴

事先准备：

主 master: 172.18.0.3

从 slaver1：172.18.0.4

从 slaver2：172.18.0.5

查看容器IP命令：docker inspect 容器ID

配置启动docker的redis容器

一般情况下，docker启动redis容器是没有redis.conf文件的，所以在这之前我们要先创建好redis.con文件。

我们可以挑选宿主机的任意位置创建

```
# master
[root@VM_0_10_centos ~]# mkdir -p /root/redis_1/{conf,data}
[root@VM_0_10_centos ~]# cd /root/redis/conf/
[root@VM_0_10_centos conf]# vim redis.conf

# slaver1
[root@VM_0_10_centos ~]# mkdir -p /root/redis_2/{conf,data}
[root@VM_0_10_centos ~]# cd /root/redis/conf/
[root@VM_0_10_centos conf]# vim redis.conf

# slaver2
[root@VM_0_10_centos ~]# mkdir -p /root/redis_3/{conf,data}
[root@VM_0_10_centos ~]# cd /root/redis/conf/
[root@VM_0_10_centos conf]# vim redis.conf
```

接下来我们要到**redis**的官网找**redis.conf**的内容

redis官网的redis.conf文件，我们可以找对应redis版本的**redis.conf**文件，点击打开之后把内容复制服务器要创建的**redis.conf**里 :wq 保存退出即可。

创建完redis.conf文件之后就可以启动redis容器了

```
# master

docker run -d --privileged=true -p 6379:6379 -p 26379:26379 -v /root/redis_1/conf/redis.conf /etc/redis/redis.conf -v /root/redis_1/data:/data --name redis32_master 87856cc39862 redis-server /etc/redis/redis.conf --appendonly yes

# slaver1
docker run -d --privileged=true -p 6378:6379 -p 26378:26379 -v /root/redis_2/conf/redis.conf /etc/redis/redis.conf -v /root/redis_2/data:/data --name redis32_slaver1 87856cc39862 redis-erver /etc/redis/redis.conf --appendonly yes

# slaver2
docker run -d --privileged=true -p 6377:6379 -p 26377:26379 -v /root/redis_3/conf/redis.conf /etc/redis/redis.conf -v /root/redis_3/data:/data --name redis32_slaver2 87856cc39862 redis-erver /etc/redis/redis.conf --appendonly yes
```

参数说明：

--privileged=true：容器内的root拥有真正root权限，否则容器内root只是外部普通用户权限

-p 6379:6379 ：映射容器6379端口到宿主机上

-p 26379:26379：同上，该端口后面实现哨兵模式要用到

-v /root/redis/conf/redis.conf:/etc/redis/redis.conf：映射配置文件

-v /root/redis/data:/data：映射数据目录

redis-server /etc/redis/redis.conf：指定配置文件启动redis-server进程

--name redis32：容器名

87856cc39862：image ID，可以用docker images查看，用于你要启动哪个容器

--appendonly yes：开启数据持久化

自此redis容器就启动了，要实现哨兵模式必须要三台机以上，所以你们可以在本地搭建三台容器，或有多余的服务器每个服务器搭建一个容器都可以（比较推荐）

实现主从关系

配置redis.conf文件

master的**redis.conf**文件(其余是默认设置不需要改动)

```
# 将127.0.0.1改为0.0.0.0，让其他服务器可访问
bind 0.0.0.0
```

```
daemonize yes
```

slaver1的redis.conf文件

```
# 将127.0.0.1改为0.0.0.0，让其他服务器可访问
bind 0.0.0.0
# 配置master服务器redis的IP地址和端口
slaveof 172.18.0.3 6379
```

slaver2的redis.conf文件

```
# 将127.0.0.1改为0.0.0.0，让其他服务器可访问
bind 0.0.0.0
# 配置master服务器redis的IP地址和端口
slaveof 172.18.0.3 6379
```

配置完之后，重启主从**redis**容器，随后可以测试一下

1. 主服务器写入，从服务器可以读取到
2. 从服务器不能写入

```
172.18.0.3:6379> set name lzl
OK
172.18.0.4:6379> get name
"lzl"
172.18.0.5:6379> get name
"lzl"
# 从服务器不能写入
172.18.0.4:6379> set name lzl
(error) READONLY You can't write against a read only slave.
172.18.0.5:6379> set nam fdk
(error) READONLY You can't write against a read only slave.
```

实现哨兵模式

sentinel是哨兵，用于监视主从服务器的运行状况，如果主服务器挂掉，会在从服务器中选举一个作主服务器。

配置文件如下，**文件要创建在容器里**，可以到任意目录下创建，这里我是到**docker**宿主机映射的 **/etc redis** 目录下创建的文件

容器默认没有vim命令，所以我们要自行安装，安装命令如下：

```
# 这个命令的作用是：同步 /etc/apt/sources.list 和 /etc/apt/sources.list.d 中列出的源的索引，这才能获取到最新的软件包。如果没有先执行这个命令是没办法安装vim的，包括下面的ps和netstat命

apt-get update

# 安装 vim命令
apt-get install vim
```

参数说明：

命令：**sentinel monitor mymaster 172.18.0.3 6379 1**

sentinel monitor：实现哨兵模式的固定句式

mymaster：mymaster是主节点的别名，可用于配置项目

172.18.0.3 6379：当前Sentinel节点监控 172.18.0.3:6379 这个主节点

1：代表判断主节点失败至少需要1个Sentinel节点节点同意

master的sentinel.conf

```
[root@VM_0_10_centos ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
ORTS                                      NAMES
e4a3df36f1c1        87856cc39862        "docker-entrypoint.s…"   18 hours ago        Up 17 hours
      0.0.0.0:6379->6379/tcp, 0.0.0.0:26379->26379/tcp   redis32_master
fa7c0e67a80b        87856cc39862        "docker-entrypoint.s…"   35 minutes ago      Up 32 minu
es       0.0.0.0:26378->26378/tcp, 0.0.0.0:6378->6379/tcp   redis32_slaver1
a77a085139a1        87856cc39862        "docker-entrypoint.s…"   19 hours ago        Up 57 minu
es       0.0.0.0:6377->6379/tcp, 0.0.0.0:26377->26377/tcp   redis32_slaver2

[root@VM_0_10_centos ~]# docker exec -it redis32 /bin/bash
root@e4a3df36f1c1:/data# cd /etc/redis
root@e4a3df36f1c1:/redis#  vim sentinel.conf

port 26379
# 初次配置时的状态，这个sentinel会自动更新
sentinel monitor mymaster 172.18.0.3 6379 1
daemonize yes
# 生成一个日志文件到当前目录下
logfile "./sentinel_log.log"
```

slaver1的sentinel.conf

```
#同上，参数上面即可，但是port参数不一样，对应docker容器启动命令上开放的端口，我这里maste
开放的是26379，slaver1是26378，slaver1是26377

[root@VM_0_10_centos ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
ORTS                                      NAMES
e4a3df36f1c1        87856cc39862        "docker-entrypoint.s…"   18 hours ago        Up 17 hours
      0.0.0.0:6379->6379/tcp, 0.0.0.0:26379->26379/tcp   redis32_master
fa7c0e67a80b        87856cc39862        "docker-entrypoint.s…"   35 minutes ago      Up 32 minu
es       0.0.0.0:26378->26378/tcp, 0.0.0.0:6378->6379/tcp   redis32_slaver1
a77a085139a1        87856cc39862        "docker-entrypoint.s…"   19 hours ago        Up 57 minu
es       0.0.0.0:6377->6379/tcp, 0.0.0.0:26377->26377/tcp   redis32_slaver2

[root@VM_0_10_centos ~]# docker exec -it redis32 /bin/bash
root@e4a3df36f1c1:/data# cd /etc/redis
root@e4a3df36f1c1:/redis#  vim sentinel.conf

port 26378
# 初次配置时的状态，这个sentinel会自动更新
sentinel monitor mymaster 172.18.0.3 6379 1
daemonize yes
logfile "./sentinel_log.log"
```

slaver2的sentinel.conf

#同上，参数上面即可，但是port参数不一样，对应docker容器启动命令上开放的端口，我这里maste开放的是26379，slaver1是26378，slaver1是26377

```
[root@VM_0_10_centos ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
ORTS                                      NAMES
e4a3df36f1c1        87856cc39862        "docker-entrypoint.s…"   18 hours ago        Up 17 hours
      0.0.0.0:6379->6379/tcp, 0.0.0.0:26379->26379/tcp   redis32_master
fa7c0e67a80b        87856cc39862        "docker-entrypoint.s…"   35 minutes ago      Up 32 minu
es      0.0.0.0:26378->26378/tcp, 0.0.0.0:6378->6379/tcp   redis32_slaver1
a77a085139a1        87856cc39862        "docker-entrypoint.s…"   19 hours ago        Up 57 minu
es      0.0.0.0:6377->6379/tcp, 0.0.0.0:26377->26377/tcp   redis32_slaver2

[root@VM_0_10_centos ~]# docker exec -it redis32_slaver2 /bin/bash
root@e4a3df36f1c1:/data# cd /etc/redis
root@e4a3df36f1c1:/redis#  vim sentinel.conf

port 26377
# 初次配置时的状态，这个sentinel会自动更新
sentinel monitor mymaster 172.18.0.3 6379 1
daemonize yes
logfile "./sentinel_log.log"
```

文件创建完成之后，启动sentinel，主从redis容器都要进行操作，操作方式一样

```
root@e4a3df36f1c1:/redis# redis-server sentinel.conf --sentinel
```

启动完成，进入reids Shell操作界面用 **info replication** 命令查看服务状态，也可以用ps命令，或者netstat命令查看是否启动成功

容器默认是没有ps和netstat命令的，所以我们要自己安装，安装命令如下：

```
# ps
apt-get install procps
# netstat
apt-get install net-tools
```

master

```
root@e4a3df36f1c1:/redis# reids-cli
127.0.0.1:6379> info replication
# Replication
role:master
connected_slaves:1
slave0:ip=122.51.148.103,port=6379,state=online,offset=197,lag=0
master_repl_offset:197
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:2
repl_backlog_histlen:196
```

slave1

```
root@e4a3df36f1c1:/redis# reids-cli
127.0.0.1:6379> info replication
# Replication
role:slave
master_host:49.234.200.117
master_port:6379
master_link_status:up
master_last_io_seconds_ago:7
master_sync_in_progress:0
slave_repl_offset:253
slave_priority:100
slave_read_only:1
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
```

slave2

```
root@e4a3df36f1c1:/redis# reids-cli
127.0.0.1:6379> info replication
# Replication
role:slave
master_host:49.234.200.117
master_port:6379
master_link_status:up
master_last_io_seconds_ago:1
master_sync_in_progress:0
slave_repl_offset:420761
slave_priority:100
slave_read_only:1
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
```

最后想要测试哨兵模式是否成功，可以将**master**主机容器**stop**，然后等待一会，在**slave1**或**slave2**用**nfo replication**命令查看**role**有没有由**slave**变成**master**，如果其中一台变成**master**则说明成功。