1.主从

核心

从库执行

```
127.0.0.1:6379> slaveof <masterIp> <masterPort>
```

```
docker pull redis
```

docker启动两个redis

端口映射

master 6379:6379

slave1 6380:6379

安装ifconfig

```
apt update & apt install net-tools
```

主库ip

```
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 7448  bytes 9701060 (9.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4803  bytes 265707 (259.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

从库，配置成从节点

```
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 7087  bytes 9093191 (8.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5061  bytes 274127 (267.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# redis-cli
127.0.0.1:6379> slaveof 172.17.0.2 6379
OK
127.0.0.1:6379>
```

使用info命令，查看信息

```
# Replication
role:slave
master_host:172.17.0.2
master_port:6379
master_link_status:up
master_last_io_seconds_ago:0
master_sync_in_progress:0
slave_repl_offset:98
slave_priority:100
slave_read_only:1
replica_announced:1
connected_slaves:0
master_failover_state:no-failover
master_replid:9e3204d548e62a81d86982d2cb5b6904296b9e2f
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:98
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:98
```

在从库上，获取到主库数据(这条数据是之前插入主库的，从库新建，还没数据)

```
127.0.0.1:6379> get name
"master"
127.0.0.1:6379>
```

至此，简单的主从搭建完成

2.sentinel

主从自动切换，主挂了，从自动变为主，原主重启后自动变为从

sential 节点
sential节点也是一个服务，提供对节点的监控，并且每个redis节点都需要启动它自己的sential节点
也可以直接连接sential服务

2.1 编辑 sentinel的配置文件

master 172.17.0.2 6379

```
port 26379
# 2代表有两个sentinel认为master down了就可以进行选举
sentinel monitor mymaster 127.0.0.1 6379 2
# master down掉 30s后进行切换
sentinel down-after-milliseconds mymaster 30000
sentinel failover-timeout mymaster 120000
sentinel parallel-syncs mymaster 1

```

slave1 172.17.0.3 6379

```
port 26380
# 2代表有两个sentinel认为master down了就可以进行选举
sentinel monitor mymaster 172.17.0.2 6379 2
# master down掉 30s后进行切换
sentinel down-after-milliseconds mymaster 10000
sentinel failover-timeout mymaster 60000
sentinel parallel-syncs mymaster 1
```

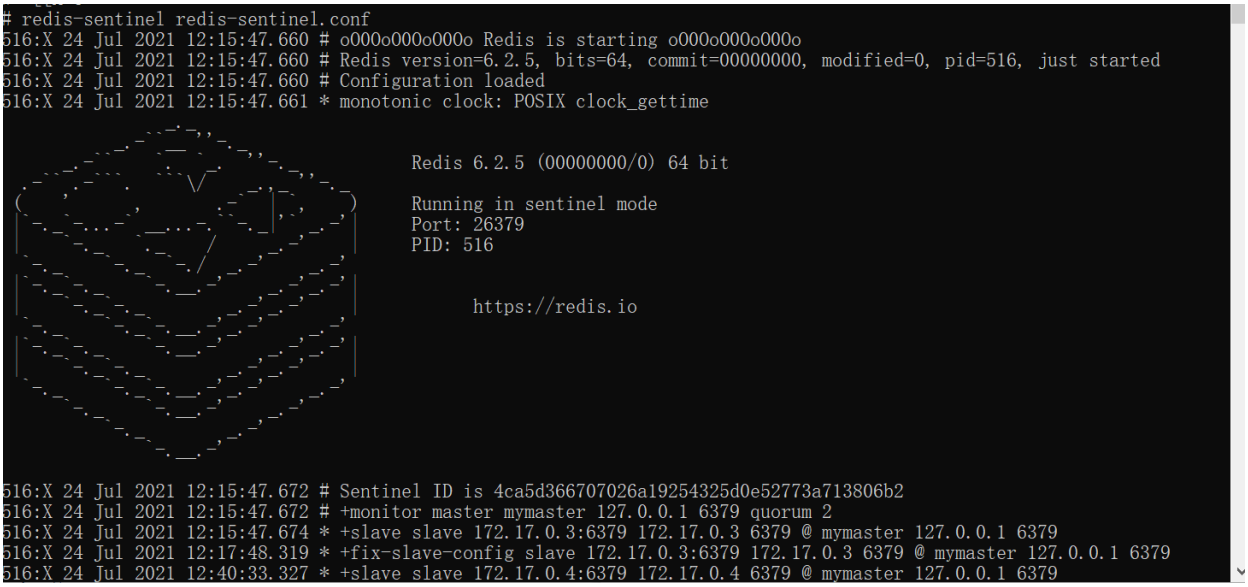
slave2 172.17.0.4 6379

```
port 26381
# 2代表有两个sentinel认为master down了就可以进行选举
sentinel monitor mymaster 172.17.0.21 6379 2
# master down掉 30s后进行切换
sentinel down-after-milliseconds mymaster 10000
sentinel failover-timeout mymaster 60000
sentinel parallel-syncs mymaster 1
```

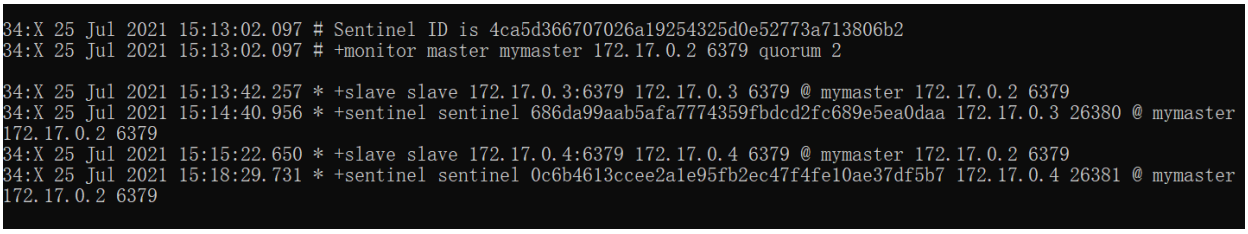
2.2 启动sentinel服务

三个docker分别执行

```
# rendis-sentinel ./redis-sentinel.conf
```



1主2从+sentinel



2.3 master down

master切换到slave 172.17.0.4

```
40:X 25 Jul 2021 15:18:28.430 * +sentinel sentinel 686da99aab5afa7774359fbdcd2fc689e5ea0daa 172.17.0.3 26380 @ mymaster
172.17.0.2 6379
40:X 25 Jul 2021 15:20:18.591 # +sdown master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:18.591 # +sdown sentinel 4ca5d366707026a19254325d0e52773a713806b2 172.17.0.2 26379 @ mymaster 172
.17.0.2 6379
40:X 25 Jul 2021 15:20:18.644 # +odown master mymaster 172.17.0.2 6379 #quorum 2/2
40:X 25 Jul 2021 15:20:18.644 # +new-epoch 1
40:X 25 Jul 2021 15:20:18.644 # +try-failover master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:18.653 # +vote-for-leader 0c6b4613ccee2a1e95fb2ec47f4fe10ae37df5b7 1
40:X 25 Jul 2021 15:20:18.660 # 686da99aab5afa7774359fbdcd2fc689e5ea0daa voted for 0c6b4613ccee2a1e95fb2ec47f4fe10ae37df
5b7 1
40:X 25 Jul 2021 15:20:18.709 # +elected-leader master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:18.710 # +failover-state-select-slave master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:18.776 # +selected-slave slave 172.17.0.4:6379 172.17.0.4 6379 @ mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:18.776 * +failover-state-send-slaveof-noone slave 172.17.0.4:6379 172.17.0.4 6379 @ mymaster 172.
17.0.2 6379
40:X 25 Jul 2021 15:20:18.860 * +failover-state-wait-promotion slave 172.17.0.4:6379 172.17.0.4 6379 @ mymaster 172.17.0
.2 6379
40:X 25 Jul 2021 15:20:19.194 # +promoted-slave slave 172.17.0.4:6379 172.17.0.4 6379 @ mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:19.194 # +failover-state-reconf-slaves master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:19.241 * +slave-reconf-sent slave 172.17.0.3:6379 172.17.0.3 6379 @ mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:19.741 # -odown master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:20.240 * +slave-reconf-inprog slave 172.17.0.3:6379 172.17.0.3 6379 @ mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:20.240 * +slave-reconf-done slave 172.17.0.3:6379 172.17.0.3 6379 @ mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:20.291 # +failover-end master mymaster 172.17.0.2 6379
40:X 25 Jul 2021 15:20:20.292 # +switch-master mymaster 172.17.0.2 6379 172.17.0.4 6379
40:X 25 Jul 2021 15:20:20.292 * +slave slave 172.17.0.3:6379 172.17.0.3 6379 @ mymaster 172.17.0.4 6379
40:X 25 Jul 2021 15:20:20.292 * +slave slave 172.17.0.2:6379 172.17.0.2 6379 @ mymaster 172.17.0.4 6379
40:X 25 Jul 2021 15:20:30.317 # +sdown slave 172.17.0.2:6379 172.17.0.2 6379 @ mymaster 172.17.0.4 6379
```

slave变成 master可以进行写入

```
# redis-cli
127.0.0.1:6379> slaveof 172.17.0.2 6379
OK
127.0.0.1:6379> get a
"1"
127.0.0.1:6379> set b 2
(error) READONLY You can't write against a read only replica.
127.0.0.1:6379> set b 2
OK
127.0.0.1:6379>
```

原主节点重启后变成从节点

```
33:X 25 Jul 2021 15:26:50.334 # Sentinel ID is 4ca5d366707026a19254325d0e52773a713806b2
33:X 25 Jul 2021 15:26:50.334 # +monitor master mymaster 172.17.0.2 6379 quorum 2
33:X 25 Jul 2021 15:26:50.373 # +new-epoch 1
33:X 25 Jul 2021 15:26:50.373 # +config-update-from sentinel 686da99aab5afa7774359fbdcd2fc689e5ea0daa 172.17.0.3 26380 @
 mymaster 172.17.0.2 6379
33:X 25 Jul 2021 15:26:50.373 # +switch-master mymaster 172.17.0.2 6379 172.17.0.4 6379
33:X 25 Jul 2021 15:26:50.373 * +slave slave 172.17.0.3:6379 172.17.0.3 6379 @ mymaster 172.17.0.4 6379
33:X 25 Jul 2021 15:26:50.373 * +slave slave 172.17.0.2:6379 172.17.0.2 6379 @ mymaster 172.17.0.4 6379
```

3.cluster

参考: https://juejin.cn/post/6844904057044205582#heading-3

3.1 redis.conf

```
port 6379
```

```
2
3 bind 0.0.0.0
4 dir ./
5 # 开启集群模式
6 cluster-enabled yes
7 cluster-config-file nodes-6379.conf
8 cluster-node-timeout 15000
9
```

3.2 指定配置文件启动redis

```
1 docker run -p 6379:6379 --name redis -v
/usr/local/docker/redis.conf:/etc/redis/redis.conf -v /usr/local/docker/dat
a:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes
```

3.3 启动redis服务(3主6从)

master1

```
1 docker run -p 16379:6379 --name redis-master1 -v C:\\Users\\xiong\\Downlo
ads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /et
c/redis/redis.conf --appendonly yes
```

master2

```
1 docker run -p 26379:6379 --name redis-master2 -v C:\\Users\\xiong\\Downlo
ads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /et
c/redis/redis.conf --appendonly yes
```

master3

```
1 docker run -p 36379:6379 --name redis-master3 -v C:\\Users\\xiong\\Downlo
ads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /et
c/redis/redis.conf --appendonly yes
```

slave11

```
1 docker run -p 16380:6379 --name redis-slave11 -v C:\\Users\\xiong\\Downlo
ads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /et
c/redis/redis.conf --appendonly yes
```

slave12

```
1 docker run -p 16381:6379 --name redis-slave12 -v C:\\Users\\xiong\\Downlo
ads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /et
c/redis/redis.conf --appendonly yes
```

slave21

```
docker run -p 26380:6379 --name redis-slave21 -v C:\\Users\\xiong\\Downloads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /etc/redis/redis.conf --appendonly yes
```

slave22

```
docker run -p 26381:6379 --name redis-slave22 -v C:\\Users\\xiong\\Downloads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /etc/redis/redis.conf --appendonly yes
```

slave31

```
docker run -p 36380:6379 --name redis-slave31 -v C:\\Users\\xiong\\Downloads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /etc/redis/redis.conf --appendonly yes
```

slave32

```
docker run -p 36381:6379 --name redis-slave32 -v C:\\Users\\xiong\\Downloads\\redis\\redis.conf:/etc/redis/redis.conf -d redis:v0.2 redis-server /etc/redis/redis.conf --appendonly yes
```

3.4 手动配置 方式

3.4.1 将所有节点加入集群

```
> cluster meet ip port
```

```
# redis-cli
127.0.0.1:6379> cluster meet 172.17.0.3 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.4 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.5 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.6 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.7 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.8 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.9 6379
OK
127.0.0.1:6379> cluster meet 172.17.0.10 6379
OK
127.0.0.1:6379> cluster nodes
64719a12de21696d83c671ab26cb20a4d74855c1 172.17.0.6:6379@16379 master - 0 1627281521462 4 connected
0ea12d0b9ac1564a9298de9d8d04950862752545 172.17.0.4:6379@16379 master - 0 1627281520457 2 connected
08f5f2000b29f7153f241d837fa8ca7c62301086 172.17.0.2:6379@16379 myself,master - 0 1627281518000 1 connected
04d3a02f5634cdeb0ee5d2a880a71f9209ce373b 172.17.0.7:6379@16379 master - 0 1627281519000 5 connected
c0ee23be6c1bf8380ffdc23bc87555a04ee58a3b 172.17.0.3:6379@16379 master - 0 1627281519000 6 connected
c607bd2333d5d44da2d9558c37506889d4a56a47 172.17.0.10:6379@16379 master - 0 1627281522467 8 connected
e741ce272133fa5b642899f802c58bc794ed13bc 172.17.0.9:6379@16379 master - 0 1627281518000 0 connected
c2aa05534c022a5dadf284fc103cc92269465f5d 172.17.0.8:6379@16379 master - 0 1627281521000 7 connected
704cd4ebe7e13c0461bec62ea69d0938fe5a1fff 172.17.0.5:6379@16379 master - 0 1627281520000 3 connected
127.0.0.1:6379>
```

3.4.2 将redis槽位均匀分配给三个master

```
# redis-cli -h 172.17.0.2 -p 6379 cluster addslots $(seq 5461)
OK
# redis-cli -h 172.17.0.3 -p 6379 cluster addslots $(seq 5462 10922)
OK
# redis-cli -h 172.17.0.4 -p 6379 cluster addslots $(seq 10923 16383)
OK
#
```

注意: 这里需要在第一个master 单独 cluster addslots 0 （添加第0个槽位）
或者使用 $(seq 0 5461)

这里有个错误

```
# redis-cli -h 172.17.0.2 -p 6379 cluster addslots {0..5461}
(error) ERR Invalid or out of range slot
```

原因 : 在我拉下来的docker容器里 {0..5461} 表达式失效
解决方案: 使用 $(seq 5461) 替换

$(seq 5461) 1-5461
$(seq 3 5461) 3-5461

3.4.3 进行主从配置

```
// id信息 可以从cluster nodes中看到
# redis-cli -h 172.17.0.8 -p 6379 cluster replicate xxxx
```

```
3  # redis-cli -h <slave_ip> -p <slave_port> cluster replicate <master_id>
```

```
OK
127.0.0.1:6379> cluster nodes
704cd4ebe7e13c0461bec62ea69d0938fe5a1fff 172.17.0.5:6379@16379 myself,slave 08f5f2000b29f7153f241d837fa8ca7c62301086 0 1
627283457000 1 connected
04d3a02f5634cdeb0ee5d2a880a71f9209ce373b 172.17.0.7:6379@16379 master - 0 1627283460695 5 connected
64719a12de21696d83c671ab26cb20a4d74855c1 172.17.0.6:6379@16379 master - 0 1627283457678 4 connected
08f5f2000b29f7153f241d837fa8ca7c62301086 172.17.0.2:6379@16379 master - 0 1627283458684 1 connected 1-5461
0ea12d0b9ac1564a9298de9d8d04950862752545 172.17.0.4:6379@16379 master - 0 1627283456000 2 connected 10923-16383
e741ce272133fa5b642899f802c58bc794ed13bc 172.17.0.9:6379@16379 master - 0 1627283459690 0 connected
c2aa05534c022a5dadf284fc103cc92269465f5d 172.17.0.8:6379@16379 master - 0 1627283455665 7 connected
c607bd2333d5d44da2d9558c37506889d4a56a47 172.17.0.10:6379@16379 master - 0 1627283457000 8 connected
c0ee23be6c1bf8380ffdc23bc87555a04ee58a3b 172.17.0.3:6379@16379 master - 0 1627283457000 6 connected 5462-10922
127.0.0.1:6379>
# redis-cli -h 172.17.0.6 -p 6379 cluster replicate 08f5f2000b29f7153f241d837fa8ca7c62301086
OK
# redis-cli -h 172.17.0.7 -p 6379  cluster repliacte c0ee23be6c1bf8380ffdc23bc87555a04ee58a3b
(error) ERR Unknown subcommand or wrong number of arguments for 'repliacte'. Try CLUSTER HELP.
# redis-cli -h 172.17.0.7 -p 6379 cluster replicate c0ee23be6c1bf8380ffdc23bc87555a04ee58a3b
OK
# redis-cli -h 172.17.0.8 -p 6379 cluster replicate c0ee23be6c1bf8380ffdc23bc87555a04ee58a3b
OK
# redis-cli -h 172.17.0.9 -p 6379 cluster replicate 0ea12d0b9ac1564a9298de9d8d04950862752545
OK
# redis-cli -h 172.17.0.10 -p 6379 cluster replicate 0ea12d0b9ac1564a9298de9d8d04950862752545
OK
#
```

3.4.4 查看集群信息

配置完成后，可以看到集群状态为(ok).

注意，需要将16384个槽位全部分配完毕才行。

```
127.0.0.1:6379> cluster addslots 0
OK
127.0.0.1:6379> cluster info
cluster_state:ok
cluster_slots_assigned:16384
cluster_slots_ok:16384
cluster_slots_pfail:0
cluster_slots_fail:0
cluster_known_nodes:9
cluster_size:3
cluster_current_epoch:8
cluster_my_epoch:1
cluster_stats_messages_ping_sent:2647
cluster_stats_messages_pong_sent:2714
cluster_stats_messages_meet_sent:8
cluster_stats_messages_sent:5369
cluster_stats_messages_ping_received:2714
cluster_stats_messages_pong_received:2655
cluster_stats_messages_received:5369
127.0.0.1:6379>
```

3.5 自动配置方式

自动配置方式

```
redis-cli --cluster create 192.168.163.132:6379 192.168.163.132:6380 192.168.163.132:6381 192.168.163.132:6382 192.168.163.132:6383 192.168.163.132:6384 --cluster-replicas 1
```

说明：--cluster-replicas 参数为数字，1表示每个主节点需要1个从节点。

3.6 配置sentinel

.....