Setup a highly-available Redis cluster with Sentinel and HAProxy -

Author:- Jagjit, GNU IT Solutions & Services.(GiTSS)

The purpose of this tutorial is to show You how to quickly setup a Redis cluster with Sentinel (comes with Redis) and HAProxy on Ubuntu 18.04

The purpose of this is to show You how to quickly setup a Redis cluster with Sentinel (comes with Redis) and HAProxy on Ubuntu 18.04

A quick explanation of the technologies used in this tutorial

The big picture

Prerequisites

Virtual machines

VM Instance	Name	Apps / Service	IP Address
Instance 01	haproxy-01	HAProxy	192.168.1.10
Instance 02	redis-01	Redis, Sentinel (Master)	192.168.1.11
Instance 03	redis-02	Redis, Sentinel (Slave)	192.168.1.12
Instance 04	redis-03	Redis, Sentinel (Slave)	192.168.1.13

NOTE: Remember that when Sentinel is in charge of deciding which server is Master and Slave. The above stated Master / Slave constellation will change. The table above is a just a snapshot of our initial setup.

1. Install redis-server

```
sudo apt update
sudo apt install redis-server
```

Run command below to check that every is working

```
root@redis-example-com-01:~# redis-cli ping
PONG
```

2. Install and enable UFW on the machines that will host Redis (Optional, if firewall is disabled)

```
sudo apt-get update
sudo apt install ufw
sudo ufw enable
sudo ufw allow 22
sudo ufw allow in on eth1 to any port 6379
sudo ufw allow in on eth1 to any port 26379
```

In the example above we have allowed traffic on port 22 from everywhere on the internet and we only allow traffic from our private network via our `eth1` interface (the interface that has access to our private network) to access our Redis and Sentinel services.

NOTE: Remember to harden Your SSH config in order increase security on machines that are open to traffic from the internet. I would recommend using a jumper machine.

The status of UFW should look something like

```
root@redis-example-com-01:~# ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
6379 on eth1               ALLOW       Anywhere
26379 on eth1              ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
6379 (v6) on eth1          ALLOW       Anywhere (v6)
26379 (v6) on eth1         ALLOW       Anywhere (v6)
```

The above means that redis and sentinel will only be accessible from the VMs on Your private network.

3. Understand how Master/Slave VMs will talk to each other

Our setup will have three VMs. Below we assume that our private network has the IP format of

`192.168.1.XYZ`

```
192.168.1.11 ==> Master Node (Initially)
192.168.1.12 ==> Slave Node (Initially)
192.168.1.13 ==> Slave Node (Initially)
```

The only difference between our `/etc/redis/redis.conf` files in our three VMs is that all our slave VMs must have the following 192.168.1.11 as their `Master IP` address.

Make sure that we are pointing to our Initial Master IP in the `/etc/redis/redis.conf`of our slave VMs

Add/change config files for Master VM and Slave VMs

There are two config files we need to have per virtual machine

1. /etc/redis/redis.conf
2. /etc/redis/sentinel.conf

Change a few lines in `/etc/redis/redis.conf`

Only on Slave VMs

Add the line below to replicate data from our VM that has IP `192.168.1.11` in `/etc/redis/redis.conf`

```
slaveof 192.168.1.11 6379
```

And set `protected-mode` to `no` in `/etc/redis/redis.conf`

```
protected-mode no
```

NOTE: Replace the IP in the example above with a proper IP from Your private network.

```
slaveof <YOUR_MASTER_IP_HERE> <YOUR_MASTER_PORT_HERE>
```

Create a new file in `/etc/redis/sentinel.conf` and add the lines below

Master (VM: 192.168.1.11)

```
sentinel myid 9862e14b6d9fb11c035c4a28d48573455a7876a2
sentinel monitor redis-primary 192.168.1.11 6379 2
sentinel down-after-milliseconds redis-primary 2000
sentinel failover-timeout redis-primary 5000
protected-mode no
```

NOTE: You will find the myid number in Your `/etc/redis/sentinel.conf` after You have started Sentinel the first time.

The format above is `sentinel monitor <NAME> <IP> <PORT> <QUORUM>`. Which basically means monitor this Redis node with `<IP>` and `<PORT>` , use `<NAME>` hereafter in the config and at least `<QUORUM>` (in this case two) sentinels has to agree that this Redis node is down for Sentinel to trigger the promotion of another machine to master in this cluster.

Slave (VM: 192.168.1.12)

```
sentinel myid 9862e14b6d9fb11c035c4a28d48573455a7876a2
sentinel monitor redis-primary 192.168.1.11 6379 2
sentinel down-after-milliseconds redis-primary 2000
sentinel failover-timeout redis-primary 5000
protected-mode no
```

Slave (VM: 192.168.1.13)

```
sentinel myid 9862e14b6d9fb11c035c4a28d48573455a7876a2
sentinel monitor redis-primary 192.168.1.11 6379 2
sentinel down-after-milliseconds redis-primary 2000
sentinel failover-timeout redis-primary 5000
protected-mode no
```

Ready to fire things up!

You can start/stop/restart the Redis service by running

```
systemctl start|stop|restart|status redis
```

1. Keep an eye on the logs to see what is going on under the hood

   ```
   tail -f /var/log/redis/redis-server.log
   ```

2. Start a new terminal and run the command below on all three VMs

```
systemctl restart redis
```

3. Check that Redis is responding properly

```
redis-cli ping
```

4. Check that the replication is working properly, fire up redis-cli on Your Master VM and set a key in the database named hello and its content to world

```
root@redis-example-com-01:# redis-cli SET hello "world"
OK
```

5. Fire up new terminals for the Slave VMs and run

```
redis-cli GET hello
```

The response should look something like below

```
root@redis-example-com-02:# redis-cli GET hello
"world"
```

6. Fire up Sentinel

```
redis-server /etc/redis/sentinel.conf --sentinel
```

It should give You a detailed log of what is going on what connections are setup between Master and Slave VMs.

Now go to your terminal on Your Master VM and type the command below and see the magic happen.

```
redis-cli -p 6379 DEBUG sleep 30
```

If everything is alright, one of the Slave VMs will be promoted to the "New Master". You can query the role of the redis server with the command below

```
redis-cli info replication | grep role
```

NOTE: You should run Sentinel as a daemon in production.

Add the lines below to `/etc/redis/sentinel.conf` if You want to daemonize the Sentinel service on Your production environment.

```
logfile "/var/log/redis/sentinel.log"
daemonize yes
```

You can now access the Sentinel logs on a specific VM by running

```
tail -f /var/log/redis/sentinel.log
```

Run Sentinel as a daemon via systemd

`/etc/redis/sentinel.conf`

```
sentinel myid 9862e14b6d9fb11c035c4a28d48573455a7876a2
sentinel monitor redis-primary 192.168.1.11 6379 2
sentinel down-after-milliseconds redis-primary 5000
protected-mode no
# Run in production with systemd
logfile "/var/log/redis/sentinel.log"
pidfile "/var/run/redis/sentinel.pid"
daemonize yes
```

`/etc/systemd/system/sentinel.service`

```
[Unit]
Description=Sentinel for Redis
After=network.target

[Service]
Type=forking
User=redis
```

```
Group=redis
PIDFile=/var/run/redis/sentinel.pid
ExecStart=/usr/bin/redis-server /etc/redis/sentinel.conf --sentinel
ExecStop=/bin/kill -s TERM $MAINPID
Restart=always

[Install]
WantedBy=multi-user.target
```

Change the access permissions of related files

```
sudo chown redis:redis /etc/redis/sentinel.conf
sudo chown redis:redis /var/log/redis/sentinel.log
sudo chmod 640 /etc/redis/sentinel.conf
sudo chmod 660 /var/log/redis/sentinel.log
```

Reload and allow Sentinel daemon to autostart

```
sudo systemctl daemon-reload
sudo systemctl enable sentinel.service
```

How to start, restart, stop and check status

```
sudo  systemctl start|restart|stop|status sentinel
```

Setup HAProxy for our cluster

```
sudo apt update
sudo apt install haproxy
```

You can check which version You have by running the command below

```
root@haproxy-01:# haproxy -v
HA-Proxy version 1.8.8-1ubuntu0.4 2019/01/24
Copyright 2000-2018 Willy Tarreau <willy@haproxy.org>
```

1. Go to /etc/haproxy/haproxy.cfg and replace its content with the snippet below

```
global
        daemon
        maxconn 256
```

```
defaults
        mode tcp
        timeout connect 5000ms
        timeout client 50000ms
        timeout server 50000ms


frontend http
        bind :8080
        default_backend stats


backend stats
        mode http
        stats enable

        stats enable
        stats uri /
        stats refresh 1s
        stats show-legends
        stats admin if TRUE


frontend redis-read
    bind *:6379
    default_backend redis-online


frontend redis-write
        bind *:6380
        default_backend redis-primary


backend redis-primary
        mode tcp
        balance first
        option tcp-check

        tcp-check send info\ replication\r\n
        tcp-check expect string role:master

        server redis-01:192.168.1.11:6379 192.168.1.11:6379 maxconn 1024 check
inter 1s
        server redis-02:192.168.1.12:6379 192.168.1.12:6379 maxconn 1024 check
inter 1s
        server redis-03:192.168.1.13:6379 192.168.1.13:6379 maxconn 1024 check
inter 1s


backend redis-online
        mode tcp
        balance roundrobin
        option tcp-check

        tcp-check send PING\r\n
        tcp-check expect string +PONG

        server redis-01:192.168.1.11:6379 192.168.1.11:6379 maxconn 1024 check
inter 1s
        server redis-02:192.168.1.12:6379 192.168.1.12:6379 maxconn 1024 check
inter 1s
        server redis-03:192.168.1.13:6379 192.168.1.13:6379 maxconn 1024 check
inter 1s
```

2. Restart the HAProxy service

```
sudo systemctl restart haproxy
```

3. You can browse the Stats UI that comes out of the box with HAProxy by visiting http://<YOUR_HAPROXY_IP>:8080 on Your browser

> TIP: You can access the interface without exposing port 8080 on the VM by tunneling via SSH with `ssh -N -L 8080:localhost:8080 <YOUR_USER>@<YOUR_IP_OR_DOMAIN>` and then visit `localhost:8080` from a browser on the machine You are sitting on.

4. Check that Your HAProxy server is routing traffic to Your primary Redis node by executing the command below on Your HAProxy VM.

```
apt install redis-tools -y
redis-cli -h 192.168.1.10 -p 6379 info replication | grep role
role:master
redis-cli -h 192.168.1.10 -p 6379 info replication | grep role
role:slave
redis-cli -h 192.168.1.10 -p 6379 info replication | grep role
role:slave
redis-cli -h 192.168.1.10 -p 6379 info replication | grep role
role:master
redis-cli -h 192.168.1.10 -p 6379 info replication | grep role
role:slave
redis-cli -h 192.168.1.10 -p 6379 info replication | grep role
role:slave
```

As You can see above the roundrobin balance algorithm sends each request to all the machines listed under the `backend redis-online` section in our `haproxy.cfg`

Now check the 6380 port which is designated for write requests.

It should return something similar to the response below. The important part is `role:master` as Your HAProxy should always route traffic to the master VM at any given time.

```
redis-cli -h <YOUR_HAPROXY_PRIVATE_IP> -p 6380 info replication
```

```
# Replication
role:master
connected_slaves:2
slave0:ip=192.168.1.12,port=6379,state=online,offset=300869,lag=0
slave1:ip=192.168.1.13,port=6379,state=online,offset=300869,lag=1
master_replid:5727c2e7a8807e43cde4171209059c497965626f
master_replid2:4b9ba6abc844fd849ee067ccf4bc1d1aefe3cade
master_repl_offset:301301
second_repl_offset:65678
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:454
```

```
repl_backlog_histlen:30084`
```

That's it, You should now have everything You need to setup a well functioning Redis cluster.