### Configuring REDIS CLUSTER (master slave)

### Introduction

Redis (REmote DIctionary Server) is an in-memory, key-value database, commonly referred to as a data structure server. One of the key differences between Redis and other key-value databases is Redis’s ability to store and manipulate high-level data types. These data types are fundamental data structures (lists, maps, sets, and sorted sets) that most developers are familiar with. Redis’s exceptional performance, simplicity, and atomic manipulation of data structures lends itself to solving problems that are difficult or perform poorly when implemented with traditional relational databases.

We will need two servers, one of the server will act as master & other one as slave.

#### Master server

IP address – 192.168.72.91 Redis port number – 6379

#### Slave server

IP address – 192.168.72.92 Redis port number – 6379

We will now start with the configuration on the redis server.

## Step 1: Install Redis

To get started, we will install Redis on both the **master** and **slave** servers.

We will install an up-to-date Redis Server package using [Chris Lea's Redis PPA](https://launchpad.net/%7Echris-lea/+archive/ubuntu/redis-server).

First, add the PPA to both of your servers:

$ apt-add-repository ppa:chris-lea/redis-server

Next, update the server's local package index and install the Redis server package by typing:

$ sudo apt-get update

$ sudo apt-get install redis-server -y

Check that Redis is up and running by typing:

$ redis-cli ping

You should receive back the following response:

Output

PONG

## Step 2: Secure Traffic Between the Two Servers

Open the Redis configuration file on each server:

$ vi /etc/redis/redis.conf

find the bind line and append the server's own isolated network IP address:

bind 127.0.0.1 192.168.72.91

Save and close the file. Restart the service by typing:

$ systemctl restart redis-server.service

Open up access to the Redis port:

$ ufw allow 6379

You should now be able to access one server from the other by provide the alternate server's IP address

$ redis-cli -h 192.168.72.91 ping

Output

PONG

## Step 3: Configure Redis Master

Now that Redis is up and running on each server, we have to edit their configuration files. Let's start with the **master server**.

$ vi /etc/redis/redis.conf

tcp-keepalive 60

requirepass temp123 (your\_redis\_master\_password)

there are a few optional settings you may skip this

maxmemory-policy noeviction

appendonly yes

appendfilename "redis-staging-ao.aof"

When you are finished, save and close the file.

Restart the Redis service to reload our configuration changes:

$ systemctl restart redis-server.service

## Step 4: Test the Redis Master

Check that you can authenticate using the password

$ redis-cli

you will get this type of promt

127.0.0.1:6379>

use the auth command to authenticate:

127.0.0.1:6379> auth temp123

Redis master output

OK

127.0.0.1:6379> info replication

Redis master output

# Replication

role:master

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

set a test key so that we can check after slave server configuration

127.0.0.1:3679> set test 'this key was defined on the master server'

127.0.0.1:3679> exit

Now that we have the master server ready, let's move on to our slave server.

## Step 5:Configure the Redis Slave

we need to make some changes on **slave server** to connect to our master server.

$ vi /etc/redis/redis.conf

slaveof 192.168.72.91 6379 (master server ip)

masterauth temp123 (password of master)

requirepass temp456 (password of slave)

Save and close the file when you are finished.

## Step 6: Test the Redis Slave and Apply Changes

Restart the Redis service on the slave to implement these changes:

$ systemctl restart redis-server.service

$ redis-cli

127.0.0.1:6379> auth temp456

127.0.0.1:6379> get test

Redis slave output

"this key was defined on the master server"

Once we restarted our Redis service on the slave, replication began immediately.

127.0.0.1:6379> info replication

Redis slave output

# Replication

role:slave

master\_host:192.168.72.91

master\_port:6379

master\_link\_status:up

master\_last\_io\_seconds\_ago:5

master\_sync\_in\_progress:0

slave\_repl\_offset:1387

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

If you look at the same information on the Redis master server, you would see something like this:

127.0.0.1:6379> info replication

Redis master output

# Replication

role:master

connected\_slaves:1

slave0:ip=192.168.72.92,port=6379,state=online,offset=1737,lag=1

master\_repl\_offset:1737

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:2

repl\_backlog\_histlen:1736

As you can see, the master and slave servers correctly identify.