**Redis Data Integrity and Availability Preservation Schema v2.0**

***Sentinels:***

To preserve high data integrity and availability of data in Redis DB, we now explore the use of Redis sentinels to create a data network to provide data duplica and also slave servers (back-up server), so that the DB system will not fail as a whole when one server fails.

**Network set-up:**

We here use 3x3 set-up as our basic set-up. 3x3 means that we use 3 sentinels and 3 servers (1 master + 2 slaves) to build up our data networks. This is also the minimal size of the network to provide stability of the system. Following are the requirements for a stable sentinel network configuration:

1, there must be at least 3 servers to build the network, because if there are only 2 servers, there is a chance where both server sees each other as master and there will be no server with writing permission. This phenomenon has been tested in previous demo.

2, there must be at least sentinels. Since the master server is decided by voting among sentinels, only 2 sentinels can result in 1-1 voting and can reach a dead repeat of voting.

3, number of sentinels are recommended to be odd so that a majority vote can be resulted every time voting happens.

4, quorum number (number of vote required to vote up a master, it is [# of sentinels/2+1] by default) is recommended to be configured close to half number of the sentinels so no conflict vote will happen.

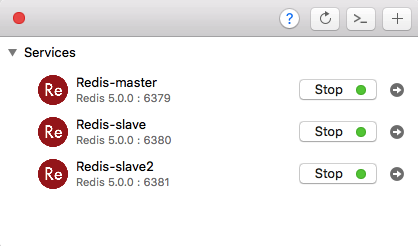
This also explains why here 3x3 is minimal size of sentinel network that will be able to provide a stable network.

**Server set-up:**

Here we use DBngin (<https://dbngin.com> ) for setting up servers. This allows us to keep the original Redis configuration while setting up multiple servers on multiple ports.



While setting up, we will use port 6379 for master server, 6380 and 6381 for slave servers. Use default configuration will be enough. Be noted that this set-up is just for initialization, the role of master and slaves will be later re-assigned by sentinels.



To set 6380 and 6381 as slaves for 6379. Run:

*>redis-cli –p 6380*

*>slaveof localhost 6379*

*[terminate or open new terminal]*

*>redis-cli –p 6381*

*>slaveof localhost 6379*

Now 6380 and 6381 will have replica data from 6379 and will be synchronized all the time.

**Sentinel set-up**:

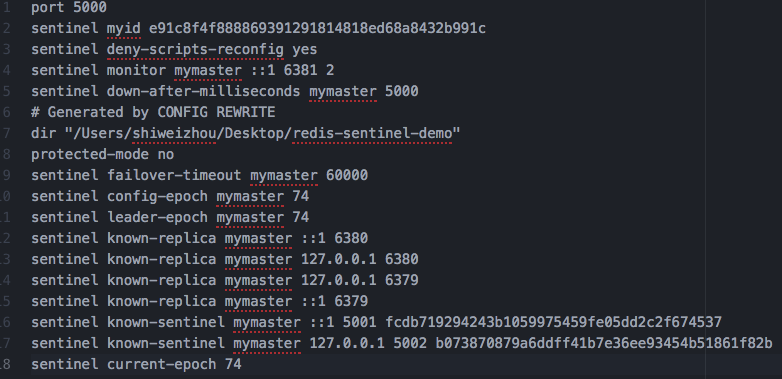
We will use ports 5000,5001 and 5002 for 3 sentinels. To start sentinels, we have to run Redis in sentinel mode with a configuration file:

*redis-sentinel sentinel.conf*

The confirmation file for port 5000 will look like:



Change the first line of port to 5001 and 5002 for other 2 sentinels. Run all 3 sentinels and they will detect servers automatically. Be noted that configuration file will be changed dynamically as sentinels are running, so making sure they have separate files and also the right permissions. The configuration file can also be used to see the current status of sentinel. Here is an example of configuration file when sentinel is running:



You will be able to see all slaves from “*sentinel known-replica*” and also current master from “*sentinel monitor my master*”. Same info can be retrieved by typing corresponding command to sentinel server.

**Running in real time:**

Refer to the demo for the real time example.

**Current Issues:**

Sentinels will re-assign roles even when there is no server fails. During re-assigning, all requests will be blocked and cause delay or even fail to current requests.

***Backup.rdp***

Other than sentinels, .rpd back up is designed as 2nd methods, which is more light weighted and faster. Even though it is more static, but it can be more responsive with manual operation. This schema is designed to generate backup.rdb with current directory with selective keys.

**Set up**:

This schema requires 2 Redis database: the main running Redis database defined as [*dbMain*] in the script and a temporary Redis databse defined as [*db2nd*] in the script. By default, [*dbMain*] runs in port 6379 and [*db2nd*] runs in 6380. The configuration can be changed accordingly. [*db2nd*] must be running during the time of script execution but can be closed before and after.

**Configuration**:

Search [configuration] to find the comment line and configuration parameters below it. There are following configurations:

* Copy\_size: this defines max limit for backup number for 1 key. E.g: if copy\_size=2, there will always be 2 copies of every backup keys based on most recent 2 timestamp.
* dbMain: defines the port where the main db is running on
* db2nd: defines the port where the temp db is running on
* keyList: defines the array of keys you want to backup to .rdb file. You can create your own I/O or for loop to create this list. By default, it contains key [test] in the array.

**Execution**:

Run [node backup.js]. You should see:

data copied

OK

backup.rdb generated in current directory

Those 3 messages shows following result. [data copied] shows keys backup to temp db; [OK] shows dump.rdb file generated from temp db; [backup.rdb generated in current directory] shows backup.rdb generated in current directory.

**Temp db schema**:

To see how temp db is designed to backup multiple keys’ copy based on time-stamp. Refer to document [Redis Selective Backup and Restore.doc].