Redis Quick Start

This is a quick start document that targets people without prior experience with Redis. Reading this document will help you:

- Download and compile Redis to start hacking.
- Use **redis-cli** to access the server.
- Use Redis from your application.
- Understand how Redis persistence works.
- Install Redis more properly.
- Find out what to read next to understand more about Redis.

Installing Redis ✓

The suggested way of installing Redis is compiling it from sources as Redis has no dependencies other than a working GCC compiler and libc. Installing it using the package manager of your Linux distribution is somewhat discouraged as usually the available version is not the latest.

You can either download the latest Redis tar ball from the redis.io web site, or you can alternatively use this special URL that always points to the latest stable Redis version, that is, http://download.redis.io/redis-stable.tar.gz.

In order to compile Redis follow these simple steps:

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
```

At this point you can test if your build has worked correctly by typing **make test**, but this is an optional step. After compilation the **src** directory inside the Redis distribution is populated with the different executables that are part of Redis:

- ✓ **redis-server** is the Redis Server itself.

- **redis-sentinel** is the Redis Sentinel executable (monitoring and failover).
- **redis-cli** is the command line interface utility to talk with Redis.
- **redis-benchmark** is used to check Redis performances.
- **redis-check-aof** and **redis-check-rdb** (**redis-check-dump** in 3.0 and below) are useful in the rare event of corrupted data files.

It is a good idea to copy both the Redis server and the command line interface into the proper places, either manually using the following commands:

- sudo cp src/redis-server /usr/local/bin/
- sudo cp src/redis-cli /usr/local/bin/

Or just using `sudo make install`.

In the following documentation we assume that /usr/local/bin is in your PATH environment variable so that you can execute both the binaries without specifying the full path.

Starting Redis

The simplest way to start the Redis server is just executing the **redis-server** binary without any argument.

```
$ redis-server
[28550] 01 Aug 19:29:28 # Warning: no config file specified, using
[28550] 01 Aug 19:29:28 * Server started, Redis version 2.2.12
[28550] 01 Aug 19:29:28 * The server is now ready to accept connect
... more logs ...
```

In the above example Redis was started without any explicit configuration file, so all the parameters will use the internal default. This is perfectly fine if you are starting Redis just to play a bit with it or for development, but for production environments you should use a configuration file.

In order to start Redis with a configuration file use the full path of the configuration file as first argument, like in the following example: **redis-server /etc/redis.conf**. You should use the `redis.conf` file included in the root directory of the Redis source code distribution as a template to write your configuration file.

Check if Redis is working

External programs talk to Redis using a TCP socket and a Redis specific protocol. This protocol is implemented in the Redis client libraries for the different programming languages. However to make hacking with Redis simpler Redis provides a command line utility that can be used to send commands to Redis. This program is called **redis-cli**.

The first thing to do in order to check if Redis is working properly is sending a **PING** command using redis-cli:

```
$ redis-cli ping
PONG
```

Running **redis-cli** followed by a command name and its arguments will send this command to the Redis instance running on localhost at port 6379. You can change the host and port used by redis-cli, just try the --help option to check the usage information.

Another interesting way to run redis-cli is without arguments: the program will start in interactive mode, you can type different commands and see their replies.

```
$ redis-cli
redis 127.0.0.1:6379> ping
PONG
redis 127.0.0.1:6379> set mykey somevalue
OK
redis 127.0.0.1:6379> get mykey
"somevalue"
```

At this point you are able to talk with Redis. It is the right time to pause a bit with this tutorial and start the fifteen minutes introduction to Redis data types in order to learn a few Redis commands. Otherwise if you already know a few basic Redis commands you can keep reading.

Securing Redis

By default Redis binds to **all the interfaces** and has no authentication at all. If you use Redis in a very controlled environment, separated from the external internet and in general from attackers, that's fine. However if Redis without any hardening is exposed to the internet, it is a big security concern. If you are not 100% sure your environment is secured properly, please check the following steps in order to make Redis more secure, which are enlisted in order of increased security.

1. Make sure the port Redis uses to listen for connections (by default 6379 and additionally 16379 if you run Redis in cluster mode, plus 26379 for Sentinel) is firewalled, so that it is not possible to contact Redis from the outside world.
2. Use a configuration file where the `bind` directive is set in order to guarantee that Redis listens on only the network interfaces you are using. For example only the loopback interface (127.0.0.1) if you are accessing Redis just locally from the same computer, and so forth.
3. Use the `requirepass` option in order to add an additional layer of security so that clients will require to authenticate using the AUTH command.
4. Use spiped or another SSL tunneling software in order to encrypt traffic between Redis servers and Redis clients if your environment requires encryption.

Note that a Redis exposed to the internet without any security is very simple to exploit, so make sure you understand the above and apply **at least** a firewalling layer. After the firewalling is in place, try to connect with `redis-cli` from an external host in order to prove yourself the instance is actually not reachable.

Using Redis from your application

Of course using Redis just from the command line interface is not enough as the goal is to use it from your application. In order to do so you need to download and install a Redis client library for your programming language. You'll find a full list of clients for different languages in this page.

For instance if you happen to use the Ruby programming language our best advice is to use the Redis-rb client. You can install it using the command **gem install redis**.

These instructions are Ruby specific but actually many library clients for popular languages look quite similar: you create a Redis object and execute commands calling methods. A short interactive example using Ruby:

```
>> require 'rubygems'
=> false
>> require 'redis'
=> true
>> r = Redis.new
=> #<Redis client v2.2.1 connected to redis://127.0.0.1:6379/0 (Rec
>> r.ping
=> "PONG"
>> r.set('foo','bar')
=> "OK"
>> r.get('foo')
=> "bar"
```

Redis persistence

You can learn how Redis persistence works on this page, however what is important to understand for a quick start is that by default, if you start Redis with the default configuration, Redis will spontaneously save the dataset only from time to time (for instance after at least five minutes if you have at least 100 changes in your data), so if you want your database to persist and be reloaded after a restart make sure to call the **SAVE** command manually every time you want to force a data set snapshot. Otherwise make sure to shutdown the database using the **SHUTDOWN** command:

```
$ redis-cli shutdown
```

This way Redis will make sure to save the data on disk before quitting. Reading the persistence page is strongly suggested in order to better understand how Redis persistence works.

Installing Redis more properly

Running Redis from the command line is fine just to hack a bit with it or for development. However at some point you'll have some actual application to run on a real server. For this kind of usage you have two different choices:

- Run Redis using screen.
- Install Redis in your Linux box in a proper way using an init script, so that after a

restart everything will start again properly.

A proper install using an init script is strongly suggested. The following instructions can be used to perform a proper installation using the init script shipped with Redis 2.4 in a Debian or Ubuntu based distribution.

We assume you already copied **redis-server** and **redis-cli** executables under /usr/local/bin.

- Create a directory in which to store your Redis config files and your data:

  ```
  sudo mkdir /etc/redis
  sudo mkdir /var/redis
  ```

- Copy the init script that you'll find in the Redis distribution under the **utils** directory into /etc/init.d. We suggest calling it with the name of the port where you are running this instance of Redis. For example:

  ```
  sudo cp utils/redis_init_script /etc/init.d/redis_6379
  ```

- Edit the init script.

  ```
  sudo vi /etc/init.d/redis_6379
  ```

Make sure to modify **REDISPORT** accordingly to the port you are using. Both the pid file path and the configuration file name depend on the port number.

- Copy the template configuration file you'll find in the root directory of the Redis distribution into /etc/redis/ using the port number as name, for instance:

  ```
  sudo cp redis.conf /etc/redis/6379.conf
  ```

- Create a directory inside /var/redis that will work as data and working directory for this Redis instance:

```
sudo mkdir /var/redis/6379
```

- Edit the configuration file, making sure to perform the following changes:
  - Set **daemonize** to yes (by default it is set to no).
  - Set the **pidfile** to `/var/run/redis_6379.pid` (modify the port if needed).
  - Change the **port** accordingly. In our example it is not needed as the default port is already 6379.
  - Set your preferred **loglevel**.
  - Set the **logfile** to `/var/log/redis_6379.log`
  - Set the **dir** to /var/redis/6379 (very important step!)
- Finally add the new Redis init script to all the default runlevels using the following command:

```
sudo update-rc.d redis_6379 defaults
```

You are done! Now you can try running your instance with:

```
sudo /etc/init.d/redis_6379 start
```

Make sure that everything is working as expected:

- Try pinging your instance with redis-cli.
- Do a test save with **redis-cli save** and check that the dump file is correctly stored into /var/redis/6379/ (you should find a file called dump.rdb).
- Check that your Redis instance is correctly logging in the log file.
- If it's a new machine where you can try it without problems make sure that after a reboot everything is still working.

Note: In the above instructions we skipped many Redis configuration parameters that you would like to change, for instance in order to use AOF persistence instead of RDB persistence, or to setup replication, and so forth. Make sure to read the example `redis.conf` file (that is heavily commented) and the other documentation you can find in this web site for more information.