https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-redis-on-ubuntu-16-04

How To Install and Configure Redis on Ubuntu 16.04

PostedMay 11, 2016 304.9kviews [NOSQL](https://www.digitalocean.com/community/tags/nosql?type=tutorials) [REDIS](https://www.digitalocean.com/community/tags/redis?type=tutorials) [UBUNTU](https://www.digitalocean.com/community/tags/ubuntu?type=tutorials) [UBUNTU 16.04](https://www.digitalocean.com/community/tags/ubuntu-16-04?type=tutorials)

**Introduction**

Redis is an in-memory key-value store known for its flexibility, performance, and wide language support. In this guide, we will demonstrate how to install and configure Redis on an Ubuntu 16.04 server.

Prerequisites

To complete this guide, you will need access to an Ubuntu 16.04 server. You will need a non-root user with sudo privileges to perform the administrative functions required for this process. You can learn how to set up an account with these privileges by following our [Ubuntu 16.04 initial server setup guide](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-16-04).

When you are ready to begin, log in to your Ubuntu 16.04 server with your sudo user and continue below.

Install the Build and Test Dependencies

In order to get the latest version of Redis, we will be compiling and installing the software from source. Before we download the code, we need to satisfy the build dependencies so that we can compile the software.

To do this, we can install the build-essential meta-package from the Ubuntu repositories. We will also be downloading the tcl package, which we can use to test our binaries.

We can update our local apt package cache and install the dependencies by typing:

* sudo apt-get update
* sudo apt-get install build-essential tcl

Download, Compile, and Install Redis

Next, we can begin to build Redis.

**Download and Extract the Source Code**

Since we won't need to keep the source code that we'll compile long term (we can always re-download it), we will build in the /tmp directory. Let's move there now:

* cd /tmp

Now, download the latest stable version of Redis. This is always available at [a stable download URL](http://download.redis.io/redis-stable.tar.gz):

* curl -O http://download.redis.io/redis-stable.tar.gz

Unpack the tarball by typing:

* tar xzvf redis-stable.tar.gz

Move into the Redis source directory structure that was just extracted:

* cd redis-stable

**Build and Install Redis**

Now, we can compile the Redis binaries by typing:

* make

After the binaries are compiled, run the test suite to make sure everything was built correctly. You can do this by typing:

* make test

This will typically take a few minutes to run. Once it is complete, you can install the binaries onto the system by typing:

* sudo make install

Configure Redis

Now that Redis is installed, we can begin to configure it.

To start off, we need to create a configuration directory. We will use the conventional /etc/redisdirectory, which can be created by typing:

* sudo mkdir /etc/redis

Now, copy over the sample Redis configuration file included in the Redis source archive:

* sudo cp /tmp/redis-stable/redis.conf /etc/redis

Next, we can open the file to adjust a few items in the configuration:

* sudo nano /etc/redis/redis.conf

In the file, find the supervised directive. Currently, this is set to no. Since we are running an operating system that uses the systemd init system, we can change this to systemd:

/etc/redis/redis.conf

. . .

# If you run Redis from upstart or systemd, Redis can interact with your

# supervision tree. Options:

# supervised no - no supervision interaction

# supervised upstart - signal upstart by putting Redis into SIGSTOP mode

# supervised systemd - signal systemd by writing READY=1 to $NOTIFY\_SOCKET

# supervised auto - detect upstart or systemd method based on

# UPSTART\_JOB or NOTIFY\_SOCKET environment variables

# Note: these supervision methods only signal "process is ready."

# They do not enable continuous liveness pings back to your supervisor.

supervised systemd

. . .

Next, find the dir directory. This option specifies the directory that Redis will use to dump persistent data. We need to pick a location that Redis will have write permission and that isn't viewable by normal users.

We will use the /var/lib/redis directory for this, which we will create in a moment:

/etc/redis/redis.conf

. . .

# The working directory.

#

# The DB will be written inside this directory, with the filename specified

# above using the 'dbfilename' configuration directive.

#

# The Append Only File will also be created inside this directory.

#

# Note that you must specify a directory here, not a file name.

dir /var/lib/redis

. . .

Save and close the file when you are finished.

Create a Redis systemd Unit File

Next, we can create a systemd unit file so that the init system can manage the Redis process.

Create and open the /etc/systemd/system/redis.service file to get started:

* sudo nano /etc/systemd/system/redis.service

Inside, we can begin the [Unit] section by adding a description and defining a requirement that networking be available before starting this service:

/etc/systemd/system/redis.service

[Unit]

Description=Redis In-Memory Data Store

After=network.target

In the [Service] section, we need to specify the service's behavior. For security purposes, we should not run our service as root. We should use a dedicated user and group, which we will call redis for simplicity. We will create these momentarily.

To start the service, we just need to call the redis-server binary, pointed at our configuration. To stop it, we can use the Redis shutdown command, which can be executed with the redis-cli binary. Also, since we want Redis to recover from failures when possible, we will set the Restart directive to "always":

/etc/systemd/system/redis.service

[Unit]

Description=Redis In-Memory Data Store

After=network.target

[Service]

User=redis

Group=redis

ExecStart=/usr/local/bin/redis-server /etc/redis/redis.conf

ExecStop=/usr/local/bin/redis-cli shutdown

Restart=always

Finally, in the [Install] section, we can define the systemd target that the service should attach to if enabled (configured to start at boot):

/etc/systemd/system/redis.service

[Unit]

Description=Redis In-Memory Data Store

After=network.target

[Service]

User=redis

Group=redis

ExecStart=/usr/local/bin/redis-server /etc/redis/redis.conf

ExecStop=/usr/local/bin/redis-cli shutdown

Restart=always

[Install]

WantedBy=multi-user.target

Save and close the file when you are finished.

Create the Redis User, Group and Directories

Now, we just have to create the user, group, and directory that we referenced in the previous two files.

Begin by creating the redis user and group. This can be done in a single command by typing:

* sudo adduser --system --group --no-create-home redis

Now, we can create the /var/lib/redis directory by typing:

* sudo mkdir /var/lib/redis

We should give the redis user and group ownership over this directory:

* sudo chown redis:redis /var/lib/redis

Adjust the permissions so that regular users cannot access this location:

* sudo chmod 770 /var/lib/redis

Start and Test Redis

Now, we are ready to start the Redis server.

**Start the Redis Service**

Start up the systemd service by typing:

* sudo systemctl start redis

Check that the service had no errors by running:

* sudo systemctl status redis

You should see something that looks like this:

Output

● redis.service - Redis Server

Loaded: loaded (/etc/systemd/system/redis.service; enabled; vendor preset: enabled)

Active: active (running) since Wed 2016-05-11 14:38:08 EDT; 1min 43s ago

Process: 3115 ExecStop=/usr/local/bin/redis-cli shutdown (code=exited, status=0/SUCCESS)

Main PID: 3124 (redis-server)

Tasks: 3 (limit: 512)

Memory: 864.0K

CPU: 179ms

CGroup: /system.slice/redis.service

└─3124 /usr/local/bin/redis-server 127.0.0.1:6379

. . .

**Test the Redis Instance Functionality**

To test that your service is functioning correctly, connect to the Redis server with the command-line client:

* redis-cli

In the prompt that follows, test connectivity by typing:

* ping

You should see:

Output

PONG

Check that you can set keys by typing:

* set test "It's working!"

Output

OK

Now, retrieve the value by typing:

* get test

You should be able to retrieve the value we stored:

Output

"It's working!"

Exit the Redis prompt to get back to the shell:

* exit

As a final test, let's restart the Redis instance:

* sudo systemctl restart redis

Now, connect with the client again and confirm that your test value is still available:

* redis-cli
* get test

The value of your key should still be accessible:

Output

"It's working!"

Back out into the shell again when you are finished:

* exit

**Enable Redis to Start at Boot**

If all of your tests worked, and you would like to start Redis automatically when your server boots, you can enable the systemd service.

To do so, type:

* sudo systemctl enable redis

Output

Created symlink from /etc/systemd/system/multi-user.target.wants/redis.service to /etc/systemd/system/redis.service.

Conclusion

You should now have a Redis instance installed and configured on your Ubuntu 16.04 server. To learn more about how to secure your Redis installation, take a look at our [How To Secure Your Redis Installation on Ubuntu 14.04](https://www.digitalocean.com/community/tutorials/how-to-secure-your-redis-installation-on-ubuntu-14-04) (from step 3 onward). Although it was written with Ubuntu 14.04 in mind, it should mostly work for 16.04 as well.