How To Install and Use Redis

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=255*

Redis, developed in 2009, is a flexible, open-source, key value data store. Following in the footsteps of other NoSQL databases, such as Cassandra, CouchDB, and MongoDB, Redis allows the user to store vast amounts of data without the limits of a relational database. Additionally, it has also been compared to memcache and can be used, with its basic elements as a cache with persistence.

Setup

Before you install redis, there are a couple of prerequisites that need to be downloaded to make the installation as easy as possible. Start off by updating all of the apt-get packages:

```
sudo apt-get update
```

Once the process finishes, download a compiler with build essential which will help us install Redis from source:

```
sudo apt-get install build-essential
```

Finally, we need to download tcl:

```
sudo apt-get install tcl8.5
```

Installing Redis

With all of the prerequisites and dependancies downloaded to the server, we can go ahead and begin to install redis from source:

Download the tarball from google code. The latest stable version is 2.4.16.

```
wget http://redis.googlecode.com/files/redis-2.4.16.tar.gz
```

Untar it and switch into that directory:

```
tar xzf redis-2.4.16.tar.gz
```

```
cd redis-2.4.16
```

Proceed to with the make command:

```
make
```

Run the recommended make test:

```
make test
```

Finish up by running make install, which installs the program system-wide.

```
sudo make install
```

Once the program has been installed, Redis comes with a built in script that sets up Redis to run as a background daemon. To access the script move into the utils directory:

```
cd utils
```

From there, run the Ubuntu/Debian install script:

```
sudo ./install_server.sh
```

As the script runs, you can choose the default options by pressing enter. Once the script completes, the redis-server will be running in the background. You can start and stop redis with these commands (the number depends on the port you set during the installation. 6379 is the default port setting):

```
sudo service redis_6379 start
sudo service redis_6379 stop
```

You can then access the redis database by typing the following command:

```
redis-cli
```

You now have Redis installed and running. The prompt will look like this:

```
redis 127.0.0.1:6379>
```

To set Redis to automatically start at boot, run:

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

Redis Operations

A simple command to add information to a string (the most basic redis datatype) could look like this:

```
> SET users:GeorgeWashington "job: President, born:1732, dislikes: cherry trees"
OK
```

In this case the command SET is followed by the key (users:GeorgeWashington), and then the value (the string itself)

Colons in Redis have no bearing on its operations. However, they can be useful in describing the key to be filled.

We can retrieve the details of the new string with the command "GETâ€•

```
GET users:GeorgeWashington
"job: President, born:1732, dislikes: cherry trees"
```

Ranges:
When retrieving data you can define the range with 2 parameters: the first and the last element (the first element is considered 0). If your end parameter is -1, all the elements through the end of the list will be included. For example, if a list contains the 6 colors of the rainbow (arranged with the classic ROYGBV), you"ll be able to see the following results:

```
> LRANGE ROYGBV 0 3
1) "red"
2) "orange"
3) "yellow"
4) "green"
> LRANGE ROYGBV 0 -1
1) "red"
2) "orange"
3) "yellow"
4) "green"
5) "blue"
6) "violet"
> LRANGE ROYGBV 3 -1
1) "green"
2) "blue"
3) "violet"
```

Expiration:
While Redis is very helpful in storing information, it can be also used to systematically expire data.

The time that a key should exist can be designated either in seconds or with a Unix Time stamp (seconds since 1/1/1970). Two helpful commands that can control expiration are EXPIRE, which sets the length of time that a key should exist, and TTL, which displays the time remaining before the key expires.

```
> SET classified:information "Secret Stuff"
OK
> EXPIRE classified:information 45
(integer) 1
> TTL classified:information
(integer) 31
```

Attempting to retrieve the information after it has expired results in "nilâ€•

```
> GET classified:information
(nil)
```

Incrementing:
Redis also has the capability to increment strings in its database in an atomic operation. If a process is occurring to increment a value, no other command can do it at the same time and the numbers will remain consistent across the database.

```
> SET population 6
OK
> INCRBY population 10
(integer) 16
> INCR population
(integer) 17
```

Transactions:
Redis also has the capability to perform transactions, which must abide by two principals:

1) The commands must be performed in order. They will not be interrupted during the process by other requests.

2) The transactions must be processed in their entirety.

Transactions are begun with the command MULTI and subsequently run with the command EXEC .

If, for some reason, there is a server issue that disrupts the process, the transaction will be exited, and Redis will experience an error blocking it from restarting until the command,`edis-check-aof`is run and the partial transaction is undone and removed.

After that, the server will be able to restart.

```
> MULTI
OK
> SET population 6
QUEUED
> INCRBY population 10
QUEUED
> INCR population
QUEUED
redis 127.0.0.1:6379> EXEC
1) OK
2) (integer) 16
3) (integer) 1
4) (integer) 17
```

Redis Data Types

Redis has five data types: Strings, Sets, Sorted Sets, Lists, Hashes

Strings
Strings are Redis" most basic data type.

Some common commands associated with strings are:
SET: sets a value to a key
GET: gets a value from a key
DEL: deletes a key and its value
INCR: atomically increments a key
INCRBY: increments a key by a designated values
EXPIRE: the length of time that a key should exist (denoted in seconds)

Strings can be used to store objects, arranged by key. For example:

```
> SET newkey "the redis string begins"
OK
> GET newkey
"the redis string begins"
```

Sets
If you want to combine strings, you can do that with REDIS sets, a collection of unordered strings.

Some common commands for Sets are:
SADD: Add one or members to a set
SMEMBERS: Get all set members
SINTER: Find the intersection of multiple sets
SISMEMBER: check if a value is in a set
SRANDMEMBER: Get a random set member

Sets can be helpful in a variety of situations. Because each member of a set is unique, adding members to a set does not require a "check then addâ€• operation. Instead the set will check whether the item is a duplicate whenever a SADD command is performed.

```
> SADD colors red
(integer) 1
redis 127.0.0.1:6379> SADD colors orange
(integer) 1
redis 127.0.0.1:6379> SADD colors yellow
(integer) 1
redis 127.0.0.1:6379> SADD colors orange
(integer) 0
redis 127.0.0.1:6379> SMEMBERS colors
1) "red"
2) "yellow"
3) "orange"
```

Sets can be especially useful, for example, in checking for unique IP addresses visiting a page, or extracting elements at random with the SRANDMEMBER command.

Sorted Sets

Sorted sets have an intuitive name: they are a collection of strings associated with a number and are arranged by default in order of least to greatest.

This datatype works well with ranges, and, because they are ordered from the outset, adding, remove, or updating values can be done quickly.

Some common commands for Sorted Sets are:
ZADD: Adds members to a sorted set
ZRANGE: Displays the members of a sorted set arranged by index (with the default low to high)
ZREVRANGE: Displays the members of a sorted set arranged by index (from high to low)
ZREM: Removes members from a sorted set

We can create a sample sorted set with the sizes (in square miles) of the smallest countries in the world.

```
> zadd countries 9 Tuvalu
(integer) 1
> zadd countries 62 Liechtenstein
(integer) 1
> zadd countries .7 Monaco
(integer) 1
> zadd countries .2 VaticanCity
(integer) 1
> zadd countries 107 Seychelles
(integer) 1
redis 127.0.0.1:6379> zrange countries 0 -1
1) "VaticanCity"
2) "Monaco"
3) "Tuvalu"
4) "Liechtenstein"
5) "Seychelles"
```

Lists

Lists in Redis are a collection of ordered values. This is in contrast to Sets which are unordered. You can add elements to the beginning or end of a list (even when there are over ten million elements in the list) with great speed.

Some common commands associated with Lists are:
LPUSH: Add a value to the begining of a list
RPUSH: Add a value to the end of a list
LPOP: Get and remove the first element in a list
RPOP: Get and remove the last element in a list
LREM: Remove elements from a list
LRANGE: Get a range of elements from a list LTRIM: Modifies a list so leave only a specified range

We can create a list of people assigned to bring lunch each week:

```
> rpush lunch.provider alice
(integer) 1
> rpush lunch.provider bob
(integer) 2
> rpush lunch.provider carol
(integer) 3
> rpush lunch.provider don
(integer) 4
> rpush lunch.provider emily
(integer) 5
```

If we wanted to push someone to the front of the queue, we could use the LPUSH command:

```
lpush lunch.provider zoe
(integer) 6
```

The LRANGE command would then display our whole list:

```
 lrange lunch.provider 0 -1
1) "zoe"
2) "alice"
3) "bob"
4) "carol"
5) "don"
6) "emily"
```

Lists can often be used to create a timeline of events or maintain a collection of a limited number of elements.

Hashes

Hashes in Redis are a useful tool to represent objects with many fields. They are set up to store vast amount of fields in a small amount of space. A hash can store more than 4 billion field-value pairs.

Some common Hash commands are:
HMSET: Sets up multiple hash values
HSET: Sets the hash field with a string value
HGET: Retrieves the value of a hash field
HMGET: Retrieves all of the values for given hash fields
HGETALL: Retrieves all of the values for in a hash

We can use a hash to describe a sample site user.

```
> hmset user:1 username jsmith password 4bAc0s email jsmith@gmail.com
OK
> hgetall user:1
1) "username"
2) "jsmith"
3) "password"
4) "4bAc0s"
```

```
5) "email"
6) "jsmith@gmail.com"
```

If you need to look up specific information, HMGET displays the values for only the requested fields.

```
> hmget user:1 username email
1) "jsmith"
"jsmith@gmail.com"
```

Conclusion

Since its release Redis has quickly gained a lot of popularity, and is harnessed by the likes of github, flickr, Disqus, and Craigslist. Additionally, Redis can be used with most programming languages.