

How to Install Redis® on Debian 12

Learn how to install Redis on Debian 12 with our step-by-step guide. Includes installation methods, configuration tips, and troubleshooting for optimal performance.

Contents

01	Introduction	3
02	Prerequisites	3
03	Install Redis®	3
04	Configure Redis®	4
05	Secure Redis®	6
06	Access Redis®	7
07	Conclusion	10

Introduction

Redis® is a fast in-memory database popularly used for caching, processing queues in message-broker applications, displaying analytics data in leaderboards, and more. Unlike relational databases that use structured tables and SQL queries, Redis® is a key-value store that supports a variety of data structures with high-performance features for real-time applications.

This article explains how to install Redis® on Debian 12.

Prerequisites

Before you begin:

- [Deploy a Debian 12](#) instance on Vultr.
- Access the instance [using SSH](#).
- Create a [non-root user with sudo privileges and switch to the user](#).
- [Update the instance](#).

Install Redis®

Redis® is available in the default Debian 12 package repositories. You can also install a specific version using the official [PPA repository](#) or by compiling the version directly from source code. Follow the steps below to install Redis® using the APT package manager.

1. Update the server's package information index.

CONSOLE

```
$ sudo apt update
```

2. Install Redis®.

```
CONSOLE
```

```
$ sudo apt install -y redis
```

3. View the installed Redis® version.

```
CONSOLE
```

```
$ redis-server -v
```

Your output should be similar to the one below.

```
Redis server v=7.0.15 sha=00000000:0 malloc=jemalloc-5.3.0 bits=64  
build=8fef3e995a542118
```

Configure Redis®

After installing Redis®, the package starts automatically and listens for connections on the localhost port `6379` by default. Follow the steps below to configure the listening port, address, and system service preferences using the main Redis® configuration file.

1. Open the default Redis® configuration file using a text editor such as `nano`.

```
CONSOLE
```

```
$ sudo nano /etc/redis/redis.conf
```

- Find the `bind` directive and verify that the value is set to `127.0.0.1 -::1`. This instructs Redis® to only accept internal server connections.

```
INI
```

```
bind 127.0.0.1 -::1
```

- Find the `port` directive value and verify that it's set to the default port `6379` or modify it based on your needs.

```
INI
```

```
port 6379
```

- Find the `daemonize` directive and verify that it's set to `yes` to enable Redis® to run as a system service.

```
INI
```

```
daemonize yes
```

Save and close the file.

In addition, configure the following directives to set Redis® memory requirements and database limits.

- `maxmemory`: Defines the maximum amount of memory Redis® can use. When Redis® reaches this limit, it evicts keys based on the eviction policy.
- `appendonly`: Instructs Redis® to log every write operation received in an append-only file (AOF) when set to `yes` to enable data durability.
- `appendfilename`: Defines the name of the AOF file.
- `logfile`: Sets the path to the Redis® log file.
- `databases`: Sets the number of Redis® databases to support on the server. By default, `16` databases.
- `requirepass`: Sets a password to connect to the Redis® server. Clients need to authenticate using this password when enabled.
- `include`: Enables additional configuration files.

2. Restart the Redis® service to apply changes.

```
CONSOLE
```

```
$ sudo systemctl restart redis
```

Secure Redis®

Redis® does not require authentication by default, but this setting gives elevated privileges to any connected client. Follow the steps below to enable authentication, create standard users, and disable dangerous Redis® commands for specific user categories on your server.

1. Open the Redis® configuration file.

CONSOLE

```
$ sudo nano /etc/redis/redis.conf
```

- Find the `requirepass` directive.

INI

```
# requirepass foobared
```

- Uncomment the directive and replace `foobared` with a strong password for the default user.

INI

```
requirepass strong_password
```

- Add a `user` directive to create a new `db_client` user and disable dangerous Redis® commands. Replace `strong_password` with your desired password.

INI

```
user db_client on +@all -@dangerous ~* >strong_password
```

The above directive defines a new user with specific permissions, pattern-based key access, and a password with the following details:

- `user db_client`: Specifies a username of the account that can access the `redis-cli` console.
- `on`: Enables the user.
- `+@all`: Grants the user access to all commands.
- `-@dangerous`: Revokes the user's access to all dangerous commands that can modify the data set, configurations, or the server state in ways that can be harmful. For example, `FLUSHALL`, `FLUSHDB`, and `SHUTDOWN`.
- `~*`: Grants the user access to all keys on the server.
- `>strong_password`: Modifies the user's password.

The new Redis® configuration section should look like the one below.

```
INI
```

```
requirepass strong_password  
user db_client on +@all -@dangerous ~* >strong_password
```

Save and close the file.

2. Restart the Redis® service to apply the configuration changes.

```
CONSOLE
```

```
$ sudo systemctl restart redis
```

Access Redis®

Redis® includes the `redis-cli` utility, a command-line interface (CLI) that allows you to send commands to the Redis® server and read replies. Follow the steps below to use the `redis-cli` utility.

1. Access the Redis® CLI.

```
CONSOLE
```

```
$ redis-cli
```

2. Ping the database without authentication.

```
CONSOLE
```

```
127.0.0.1:6379> ping
```

The command should fail with the following message:

```
(error) NOAUTH Authentication required.
```

3. Authenticate to the Redis® database server as the `default` user using the password you set earlier. Replace `strong_password` with the actual password you set in the configuration.

```
CONSOLE
```

```
127.0.0.1:6379> auth strong_password
```

Your output should look like the one below when successful.

```
OK
```

4. Switch to the `db_client` standard user account you created earlier. Replace `strong_password` with the actual user password you set in the configuration.

```
CONSOLE
```

```
127.0.0.1:6379> auth db_client strong_password
```

Output:

```
OK
```

5. Ping the database again.

```
CONSOLE
```

```
127.0.0.1:6379> ping
```

Your output should look like the one below when successful.

```
PONG
```

6. Create a new `hello` key and set the value to `Greetings from Vultr!`.

```
CONSOLE
```

```
127.0.0.1:6379> set hello "Greetings from Vultr!"
```

7. Retrieve the `hello` key value from the database.

```
CONSOLE
```

```
127.0.0.1:6379> get hello
```

Output:

```
"Greetings from Vultr!"
```

8. Test a dangerous command with the `db_client` user such as `FLUSHALL` that deletes all keys in existing databases.

```
CONSOLE
```

```
127.0.0.1:6379> flushall
```

The dangerous command should fail with the following output:

```
(error) NOPERM this user has no permissions to run the 'flushall' command
```

9. Exit from Redis® CLI.

CONSOLE

```
127.0.0.1:6379> exit
```

Conclusion

You have installed and configured the Redis® on Debian 12 and enabled secure authentication for all database users. In addition, you've created a standard user and limited user privileges to secure the database server. For more information and configuration options, visit the [Redis® documentation](#).



VULTR

