Git • Ubuntu

Configure Git Server with HTTP on Ubuntu

7 months ago • by Shahriar Shovon

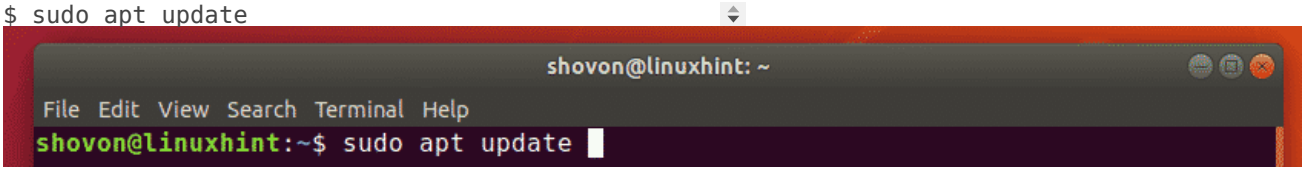
If you want to setup a Git HTTP Server for working with Git repositories privately, then this article is for you. In this article, I am going to show you how to configure a Git Smart HTTP server on Ubuntu with Apache HTTP server. So, let's get started.

Installing Git and Apache HTTP Server:

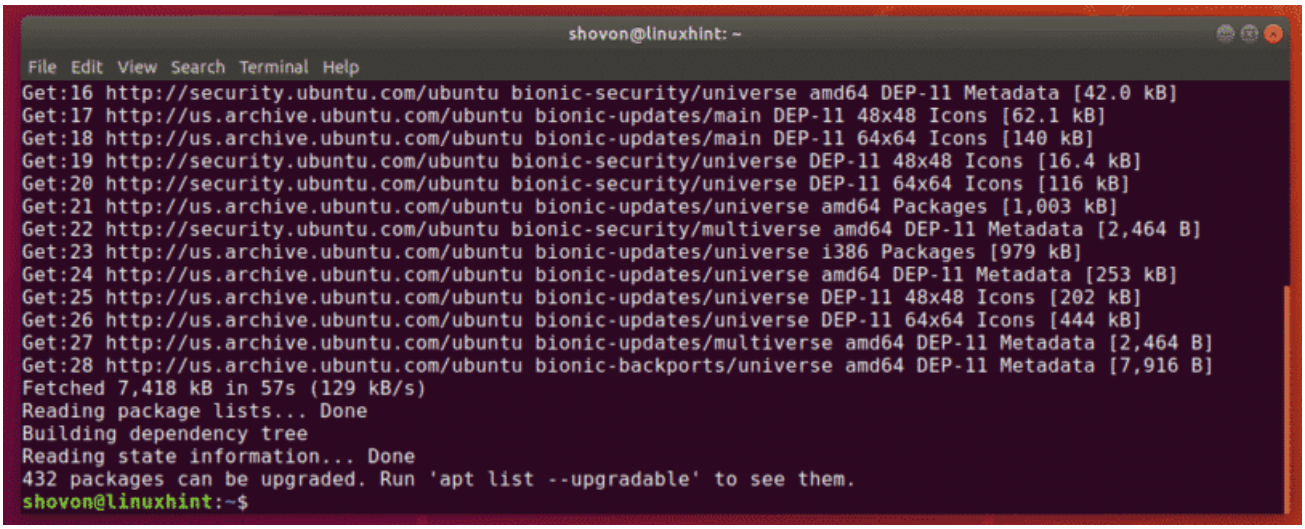
Git and Apache packages are available in the official package repository of Ubuntu. So, you can easily install it with the APT package manager.

First, update the APT package repository cache with the following command:

```
$ sudo apt update
```



The APT package repository cache should be updated.



Now, install Git and Apache with the following command:

```
$ sudo apt install git apache2 apache2-utils
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo apt install git apache2 apache2-utils
```

Now, press **Y** and then press **<Enter>** to confirm the installation.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo apt install git apache2 apache2-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data git-man libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  liberror-perl liblua5.2-0
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom git-daemon-run | git-daemon-sysvinit
  git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils git git-man libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liberror-perl liblua5.2-0
0 upgraded, 12 newly installed, 0 to remove and 432 not upgraded.
Need to get 6,445 kB of archives.
After this operation, 40.8 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Git and Apache should be installed.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/system/apac
he2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /lib/systemd
/system/apache-htcacheclean.service.
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for ureadahead (0.100.0-20) ...
Processing triggers for systemd (237-3ubuntu10.12) ...
Processing triggers for ufw (0.35-5) ...
shovon@linuxhint:~$
```

Configuring Apache HTTP Server for Git:

Now, enable Apache **mod_env**, **mod_cgi**, **mod_alias** and **mod_rewrite** modules with the following command:

```
$ sudo a2enmod env cgi alias rewrite
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo a2enmod env cgi alias rewrite
```

The required Apache modules should be enabled.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo a2enmod env cgi alias rewrite
Module env already enabled
Your MPM seems to be threaded. Selecting cgid instead of cgi.
Enabling module cgid.
Module alias already enabled
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
shovon@linuxhint:~$
```

Now, create a new directory **/var/www/git** for keeping all the Git repositories with the following command:

```
$ sudo mkdir /var/www/git
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo mkdir /var/www/git
```

Now, create a new Apache site configuration **/etc/apache2/sites-available/git.conf** for Git with the following command:

```
$ sudo nano /etc/apache2/sites-available/git.conf
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo nano /etc/apache2/sites-available/git.conf
```

Now, type in the following lines in the configuration file:

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost

SetEnv GIT_PROJECT_ROOT <strong>/var/www/git</strong>
SetEnv GIT_HTTP_EXPORT_ALL
ScriptAlias /git/ /usr/lib/git-core/git-http-backend/

Alias /git /var/www/git

<Directory /usr/lib/git-core>
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
AllowOverride None
Require all granted
</Directory>

DocumentRoot /var/www/html

<Directory /var/www>
Options Indexes FollowSymLinks MultiViews
AllowOverride None
Require all granted
</Directory>


ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

The final configuration file looks as follows. Now, save the configuration file by pressing **<Ctrl>** + **X** followed by **Y** and **<Enter>**.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3        /etc/apache2/sites-available/git.conf        Modified

<VirtualHost *:80>
  ServerAdmin webmaster@localhost

  SetEnv GIT_PROJECT_ROOT /var/www/git
  SetEnv GIT_HTTP_EXPORT_ALL
  ScriptAlias /git/ /usr/lib/git-core/git-http-backend/

  Alias /git /var/www/git

  <Directory /usr/lib/git-core>
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    AllowOverride None
    Require all granted
  </Directory>

  DocumentRoot /var/www/html

  <Directory /var/www>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Require all granted
  </Directory>

  ErrorLog ${APACHE_LOG_DIR}/error.log
  LogLevel warn
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Now, disable the default Apache site configuration with the following command:

```
$ sudo a2dissite 000-default.conf
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo a2dissite 000-default.conf
```

The default site configuration should be disabled.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
shovon@linuxhint:~$
```

Now, enable the Git site configuration with the following command:

```
$ sudo a2ensite git.conf
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo a2ensite git.conf
```

The Git site configuration should be enabled.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo a2ensite git.conf
Enabling site git.
To activate the new configuration, you need to run:
  systemctl reload apache2
shovon@linuxhint:~$
```

Now, restart Apache HTTP server with the following command:

```
$ sudo systemctl restart apache2
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo systemctl restart apache2
```

In order to bootstrap a new Git repository accessible over the Apache HTTP server, you will have to run a few commands. You don't want to do the same thing over and over again just to create a new Git repository. So, I decided to write a shell script for that purpose.

First, create a new shell script **/usr/local/bin/git-create-repo.sh** with the following command:

```
$ sudo nano /usr/local/bin/git-create-repo.sh
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo nano /usr/local/bin/git-create-repo.sh
```

Now, type in the following lines of codes in the shell script.

```
#!/bin/bash

GIT_DIR="/var/www/git"
REPO_NAME=$1

mkdir -p "${GIT_DIR}/${REPO_NAME}.git"
cd "${GIT_DIR}/${REPO_NAME}.git"

git init --bare &> /dev/null
touch git-daemon-export-ok
cp hooks/post-update.sample hooks/post-update
git config http.receivepack true
git update-server-info
chown -Rf www-data:www-data "${GIT_DIR}/${REPO_NAME}.git"
echo "Git repository '${REPO_NAME}' created in ${GIT_DIR}/${REPO_NAME}.git"
```

Once you type in these lines, the shell script should look as follows. Now, save the file by pressing **<Ctrl>** + **X** followed by **Y** and **<Enter>**.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3          /usr/local/bin/git-create-repo.sh          Modified

#!/bin/bash

GIT_DIR="/var/www/git"
REPO_NAME=$1

mkdir -p "${GIT_DIR}/${REPO_NAME}.git"
cd "${GIT_DIR}/${REPO_NAME}.git"

git init --bare &> /dev/null
touch git-daemon-export-ok
cp hooks/post-update.sample hooks/post-update
git config http.receivepack true
git update-server-info

chown -Rf www-data:www-data "${GIT_DIR}/${REPO_NAME}.git"

echo "Git repository '${REPO_NAME}' created in ${GIT_DIR}/${REPO_NAME}.git"

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter  ^_ Go To Line
```

Now, add execute permission to the shell script with the following command:

```
$ sudo chmod +x /usr/local/bin/git-create-repo.sh
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo chmod +x /usr/local/bin/git-create-repo.sh
```

Now, create a new Git repository **test** in the Git project root **/var/www/git** using the **git-create-repo.sh** shell script as follows:

```
$ sudo git-create-repo.sh test
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo git-create-repo.sh test
```

The Git repository **test** should be created.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo git-create-repo.sh test
Git repository 'test' created in /var/www/git/test.git
shovon@linuxhint:~$
```

To access the Git repository, you need the IP address of the Git HTTP server.

```
$ ip a
```

As you can see, the IP address in my case is **192.168.21.208**. It will be different for you. Replace it with yours from now on.

```
shovon@linuxhint: ~
File Edit View Search Terminal Help
shovon@linuxhint:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:fc:86:c3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.21.208/24 brd 192.168.21.255 scope global dynamic noprefixroute
 ens33
       valid_lft 1799sec preferred_lft 1799sec
    inet6 fe80::397:44dd:da7c:c58d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
shovon@linuxhint:~$
```

Now, you can clone the **test** Git repository as follows:

```
$ git clone http://192.168.21.208/git/test.git
```

```
shovon@linuxhint: ~
File Edit View Search Terminal Help
shovon@linuxhint:~$ git clone http://192.168.21.208/git/test.git
```

The Git repository **test** should be cloned.

```
shovon@linuxhint: ~
File Edit View Search Terminal Help
shovon@linuxhint:~$ git clone http://192.168.21.208/git/test.git
Cloning into 'test'...
warning: You appear to have cloned an empty repository.
shovon@linuxhint:~$
```

Now, let's add a new commit to the **test** Git repository.

```
$ cd test/
$ echo "Hello World" > hello
$ git add .
$ git commit -m 'initial commit'
```

```
shovon@linuxhint: ~/test
File Edit View Search Terminal Help
shovon@linuxhint:~$ cd test/
shovon@linuxhint:~/test$ echo "Hello World" > hello
shovon@linuxhint:~/test$ git add .
shovon@linuxhint:~/test$ git commit -m 'initial commit'
[master (root-commit) e336a5f] initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello
shovon@linuxhint:~/test$
```

Now, upload the changes to the **test** Git repository on the server as follows:

```
$ git push origin
```

As you can see, the changes are uploaded just fine.

```
shovon@linuxhint: ~/test
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~/test$ git push origin
Counting objects: 3, done.
Writing objects: 100% (3/3), 224 bytes | 224.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To http://192.168.21.208/git/test.git
 * [new branch]      master -> master
shovon@linuxhint:~/test$
```

Configuring User Authentication:

In this section, I am going to show you how to configure user authentication on the Git repositories in the server.

First, edit the **git.conf** site configuration file as follows:

```
$ sudo nano /etc/apache2/sites-available/git.conf
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo nano /etc/apache2/sites-available/git.conf
```

Now, add the following section in the configuration file.

```
<LocationMatch /git/.*\.git>
AuthType Basic
AuthName "Git Verification"
AuthUserFile /etc/apache2/git.passwd
Require valid-user
</LocationMatch>
```

Here, **/etc/apache2/git.passwd** is the user database file.

The final configuration file should look as follows. Now, save the file by pressing **<Ctrl>** + **X** followed by **Y** and **<Enter>**.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3              /etc/apache2/sites-available/git.conf              Modified

  DocumentRoot /var/www/html

  <Directory /var/www>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Require all granted
  </Directory>

  <LocationMatch /git/.*\.git>
    AuthType Basic
    AuthName "Git Verification"
    AuthUserFile /etc/apache2/git.passwd
    Require valid-user
  </LocationMatch>

  ErrorLog ${APACHE_LOG_DIR}/error.log
  LogLevel warn
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Now, create a new user database file **/etc/apache2/git.passwd** and add a new user (let's say **shovon**) to the database file as follows:

```
$ sudo htpasswd -c /etc/apache2/git.passwd shovon
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo htpasswd -c /etc/apache2/git.passwd shovon
```

Now, type in a new password for the new user and press **<Enter>**.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo htpasswd -c /etc/apache2/git.passwd shovon
New password:
```

Retype the same password and press **<Enter>**.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo htpasswd -c /etc/apache2/git.passwd shovon
New password:
Re-type new password:
```

The user-password pair should be added to the database.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo htpasswd -c /etc/apache2/git.passwd shovon
New password:
Re-type new password:
Adding password for user shovon
shovon@linuxhint:~$
```

Now, restart Apache HTTP server with the following command:

```
$ sudo systemctl restart apache2
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo systemctl restart apache2
```

Now, if you try to clone the **test** repository again, you will be asked to authenticate as you can see in the screenshot below.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ git clone http://192.168.21.208/git/test.git
Cloning into 'test'...
Username for 'http://192.168.21.208':
```

Once you authenticate using the username and password, you will be able to access the Git repository.

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ git clone http://192.168.21.208/git/test.git
Cloning into 'test'...
Username for 'http://192.168.21.208': shovon
Password for 'http://shovon@192.168.21.208':
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
shovon@linuxhint:~$
```

Even when you try to push or pull from the Git repository, you will also be asked for the username and password.

```
shovon@linuxhint: ~/test
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~/test$ git push origin
Username for 'http://192.168.21.208':
```

Once you authenticate, push/pull will work.

```
shovon@linuxhint: ~/test
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~/test$ git push origin
Username for 'http://192.168.21.208': shovon
Password for 'http://shovon@192.168.21.208':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To http://192.168.21.208/git/test.git
   1490a15..ab52606  master -> master
shovon@linuxhint:~/test$
```

You can also set different user database for different Git repositories. This might be useful for projects where a lot of people are working together on the same Git repository.

To set Git repository-wise authentication, first, edit the **git.conf** site configuration file as follows:

```
$ sudo nano /etc/apache2/sites-available/git.conf
```

Now, add the following lines in the configuration file.

```
<Location /git/test.git>
AuthType Basic
AuthName "Git Verification"
AuthUserFile /etc/apache2/git.test.passwd
Require valid-user
</Location>

<Location /git/test2.git>
AuthType Basic
AuthName "Git Verification"
AuthUserFile /etc/apache2/git.test2.passwd
Require valid-user
</Location>
```

For each Git repository **test** and **test2**, a **<Location></Location>** section is defined. A different user database file is used for each Git repository.

The final configuration file should look as follows. Now, save the configuration file by pressing **<Ctrl>** + **X** followed by **Y** and **<Enter>**.

```
shovon@linuxhint: ~/test
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3              /etc/apache2/sites-available/git.conf              Modified

  <Location /git/test.git>
    AuthType Basic
    AuthName "Git Test Repository"
    AuthUserFile /etc/apache2/git.test.passwd
    Require valid-user
  </Location>

  <Location /git/test2.git>
    AuthType Basic
    AuthName "Git Test 2 Repository"
    AuthUserFile /etc/apache2/git.test2.passwd
    Require valid-user
  </Location>


  ErrorLog ${APACHE_LOG_DIR}/error.log
  LogLevel warn
  CustomLog ${APACHE_LOG_DIR}/access.log combined

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Now, you can create the required user databases as follows:

```
$ sudo htpasswd -c /etc/apache2/git.test.passwd USERNAME
$ sudo htpasswd -c /etc/apache2/git.test2.passwd USERNAME
```

Once you're done, restart Apache HTTP server with the following command:

```
$ sudo systemctl restart apache2
```

```
shovon@linuxhint: ~
File  Edit  View  Search  Terminal  Help
shovon@linuxhint:~$ sudo systemctl restart apache2
```

Now, each Git repository should have its own set of users that can access it.

So, that's how you configure Git Server with Apache HTTP Server on Ubuntu. Thanks for reading this article.

ABOUT THE AUTHOR

Shahriar Shovon

Freelancer & Linux System Administrator. Also loves Web API development with Node.js and JavaScript. I was born in Bangladesh. I am currently studying Electronics and Communication Engineering at Khulna University of Engineering & Technology (KUET), one of the demanding public engineering universities of Bangladesh.

RELATED LINUX HINT POSTS

- How to Install and Use Osquery in Ubuntu
- How to Update Ubuntu 20.04 from the Command Line Interface
- How to use apt-cache search to find packages
- How to Set Up Hostname on Ubuntu 20.04 LTS
- Setting Up Static IP Address on Ubuntu 20.04 LTS
- Why you need apt-get clean options?
- Use apt-get to fix missing and broken packages