

System Admin

Training Assignments

|  |  |
| --- | --- |
| Program Code |  |
| Issue/Revision | x/y |
| Effective date | 04/Aug /2023 |

Assignment Day 08. System Software - Day 3

Mục Lục:

[Cài đặt và cấu hình Database PostgreSQL 3](#_Toc618279079)

[Cài đặt và cấu hình Web Server ( Apache2) 4](#_Toc1624863958)

# Cài đặt và cấu hình Database PostgreSQL

**Step 1 — Installing PostgreSQL**

To install PostgreSQL, first refresh your server’s local package index:

$sudo apt update

Then, install the Postgres package along with a -contrib package that adds some additional utilities and functionality:

$sudo apt install postgresql postgresql-contrib

Ensure that the service is started:

$sudo systemctl start postgresql.service

**Step 2 — Using PostgreSQL Roles and Databases**

By default, Postgres uses a concept called “roles” to handle authentication and authorization. One way is to switch over to the postgres account on your server by running the following command:

$sudo -i -u postgres

Then you can access the Postgres prompt by running:

$ psql

This will log you into the PostgreSQL prompt, and from here you are free to interact with the database management system right away.

To exit out of the PostgreSQL prompt, run the following:

Postgres=# \q

This will bring you back to the postgres Linux command prompt. To return to your regular system user, run the exit command:

$exit

Another way to connect to the Postgres prompt is to run the psql command as the postgres account directly with sudo:

$ sudo -u postgres psql

This will log you directly into Postgres without the intermediary bash shell in between.

Again, you can exit the interactive Postgres session by running the following:

Postgres=# \q

# Cài đặt và cấu hình Web Server ( Apache2)

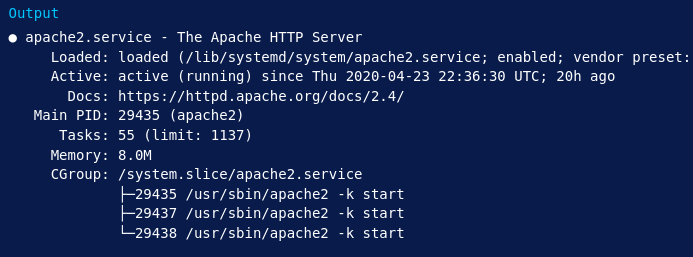
**Step 1 — Installing Apache**

Ubuntu 20.04

$sudo apt update

$sudo apt install apache2

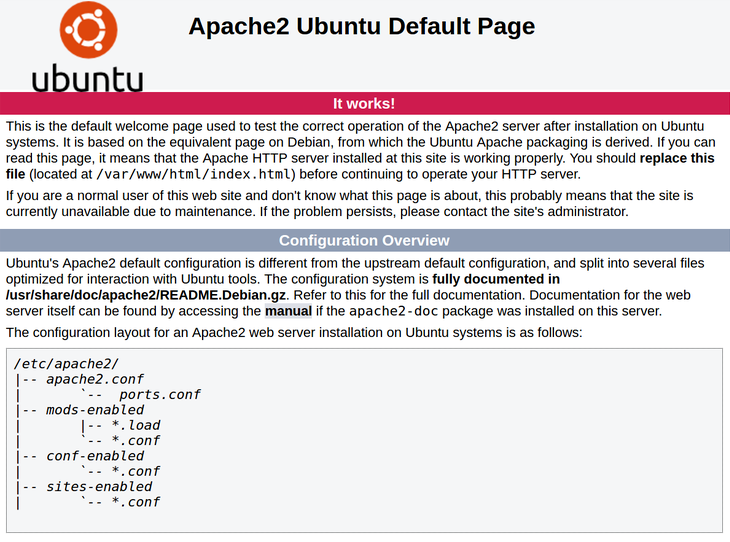
$sudo systemctl status apache2



**Step 2 — Checking your Web Server**

When you have your server’s IP address, enter it into your browser’s address bar:

<http://your_server_ip>



**Step 3 — Managing the Apache Process**

Now that you have your web server up and running, let’s go over some basic management commands using systemctl.

To stop your web server, type:

$sudo systemctl stop apache2

To start the web server when it is stopped, type:

$sudo systemctl start apache2

To stop and then start the service again, type:

$sudo systemctl restart apache2

If you are simply making configuration changes, Apache can often reload without dropping connections. To do this, use this command:

$sudo systemctl reload apache2

By default, Apache is configured to start automatically when the server boots. If this is not what you want, disable this behavior by typing:

$sudo systemctl disable apache2

To re-enable the service to start up at boot, type:

$sudo systemctl enable apache2

Apache should now start automatically when the server boots again.

**Step 4 — Setting Up Virtual Hosts**

Apache on Ubuntu 20.04 has one server block enabled by default that is configured to serve documents from the /var/www/html directory. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying /var/www/html, let’s create a directory structure within /var/www for a your\_domain site, leaving /var/www/html in place as the default directory to be served if a client request doesn’t match any other sites.

Create the directory for your\_domain as follows:

$sudo mkdir /var/www/your\_domain

Next, assign ownership of the directory with the $USER environment variable:

$sudo chown -R $USER:$USER /var/www/your\_domain

The permissions of your web roots should be correct if you haven’t modified your umask value, which sets default file permissions. To ensure that your permissions are correct and allow the owner to read, write, and execute the files while granting only read and execute permissions to groups and others, you can input the following command:

$sudo chmod -R 755 /var/www/your\_domain

Next, create a sample index.html page using nano or your favorite editor:

$sudo nano /var/www/your\_domain/index.html

Inside, add the following sample HTML:

/var/www/your\_domain/index.html

<html>

<head>

<title>Welcome to Your\_domain!</title>

</head>

<body>

<h1>Success! The your\_domain virtual host is working!</h1>

</body>

</html>

Save and close the file when you are finished.

In order for Apache to serve this content, it’s necessary to create a virtual host file with the correct directives. Instead of modifying the default configuration file located at /etc/apache2/sites-available/000-default.conf directly, let’s make a new one at /etc/apache2/sites-available/your\_domain.conf:

$sudo nano /etc/apache2/sites-available/your\_domain.conf

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name:

/etc/apache2/sites-available/your\_domain.conf

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

ServerName your\_domain

ServerAlias www.your\_domain

DocumentRoot /var/www/your\_domain

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Notice that we’ve updated the DocumentRoot to our new directory and ServerAdmin to an email that the your\_domain site administrator can access. We’ve also added two directives: ServerName, which establishes the base domain that should match for this virtual host definition, and ServerAlias, which defines further names that should match as if they were the base name.

Save and close the file when you are finished.

Let’s enable the file with the a2ensite tool:

$sudo a2ensite your\_domain.conf

Disable the default site defined in 000-default.conf:

$sudo a2dissite 000-default.conf

Next, let’s test for configuration errors:

$sudo apache2ctl configtest

You should receive the following output:

Output

Syntax OK

Restart Apache to implement your changes:

$sudo systemctl restart apache2

Apache should now be serving your domain name. You can test this by navigating to http://your\_domain, where you should see something like this:

