

WEB STACK IMPLEMENTATION (LEMP)

A technology stack is a set of frameworks and tools used to develop a software product. They are acronyms for individual technologies used together for a specific technology product. some examples are;

LAMP (Linux, Apache, MySQL, PHP or Python, or Perl)

LEMP (Linux, Nginx, MySQL, PHP or Python, or Perl)

MERN (MongoDB, ExpressJS, ReactJS, NodeJS),

MEAN (MongoDB, ExpressJS, AngularJS, NodeJS)

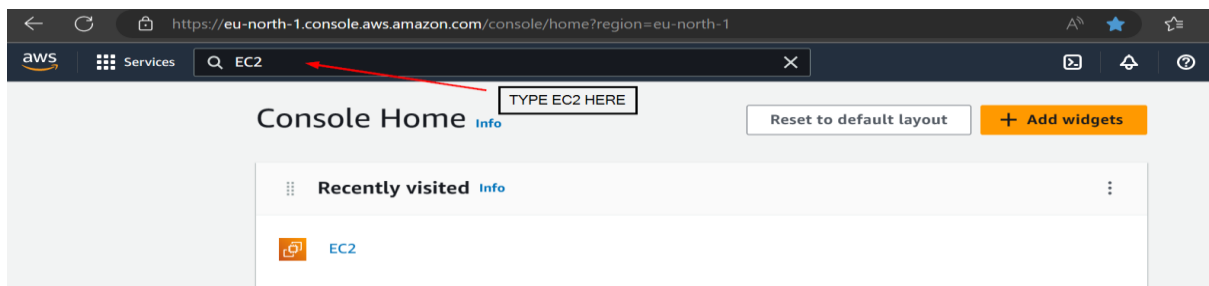
PREREQUISITES

- Aws account running an EC2 instance.
- Internet connection
- Fundamental Knowledge of downloading and installing
- Basics Linux skills

AWS is the biggest Cloud Service Provider, and it offers a free tier account that am going to leverage for this project. Right now, AWS can provide me with a free virtual server called EC2 (Elastic Compute Cloud) for my needs.

IMPLEMENTATION

- Open your PC browser and login to <https://aws.amazon.com/>
- A region is selected by default (change to your closest region if necessary), and from the search bar type EC2 and click.



- From the EC2 dashboard, click on the button “instances (running)” to launch instance.

aws Services Search [Alt+S]

New EC2 Experience Tell us what you think

EC2 Dashboard

- EC2 Global View
- Events
- Limits
- Instances
- Images
- ▼ Elastic Block Store
 - Volumes
 - Snapshots

Resources EC2 Global view

You are using the following Amazon EC2 resources in the Europe (Stockholm) Region:

Instances (running)	0	Auto Scaling Groups	0
Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	6
Load balancers	0	Placement groups	0
Security groups	10	Snapshots	0
Volumes	0		

aws Services Search [Alt+S]

New EC2 Experience Tell us what you think

EC2 Dashboard

EC2 Global View

Events

Limits

Instances Info Connect Instance state Actions **Launch instances**

Find instance by attribute or tag (case-sensitive)

Instance state = running Clear filters

click on this to launch

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Ava
No matching instances found						

aws Services Search [Alt+S]

Name and tags Info

Name

e.g. My Web Server

Enter your project name or anything you like

Add additional tags

▼ **Application and OS Images (Amazon Machine Image) Info**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Select This

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

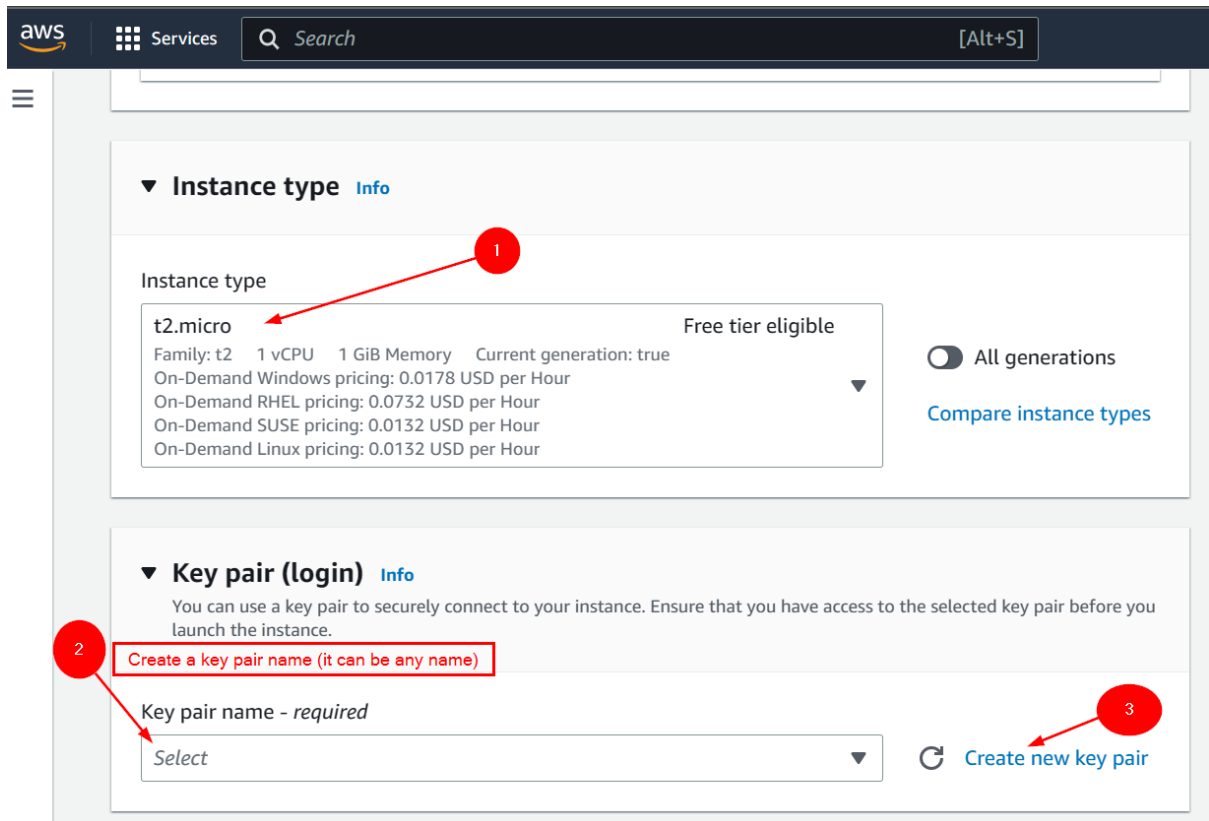
ami-0eb260c4d5475b901 (64-bit (x86)) / ami-0e3f80b3d2a794117 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

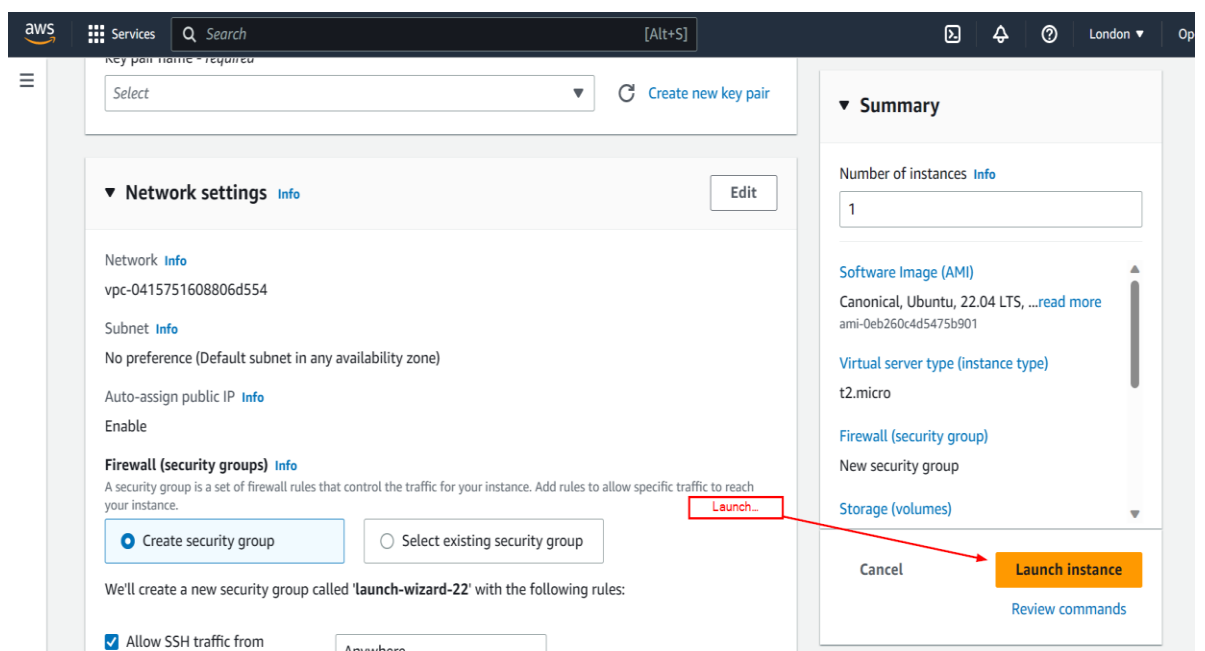
An AMI window displays, type “Ubuntu” on the search bar and hit enter or scroll down to select “Ubuntu Server 22.04 LTS (HVM), SSD Volume Type” based on your system architecture.

- The next step of configuring our EC2 is to select the instance type, preferably a t2 micro - Free tier.
- Create a key pair name, which automatically download into your local system.



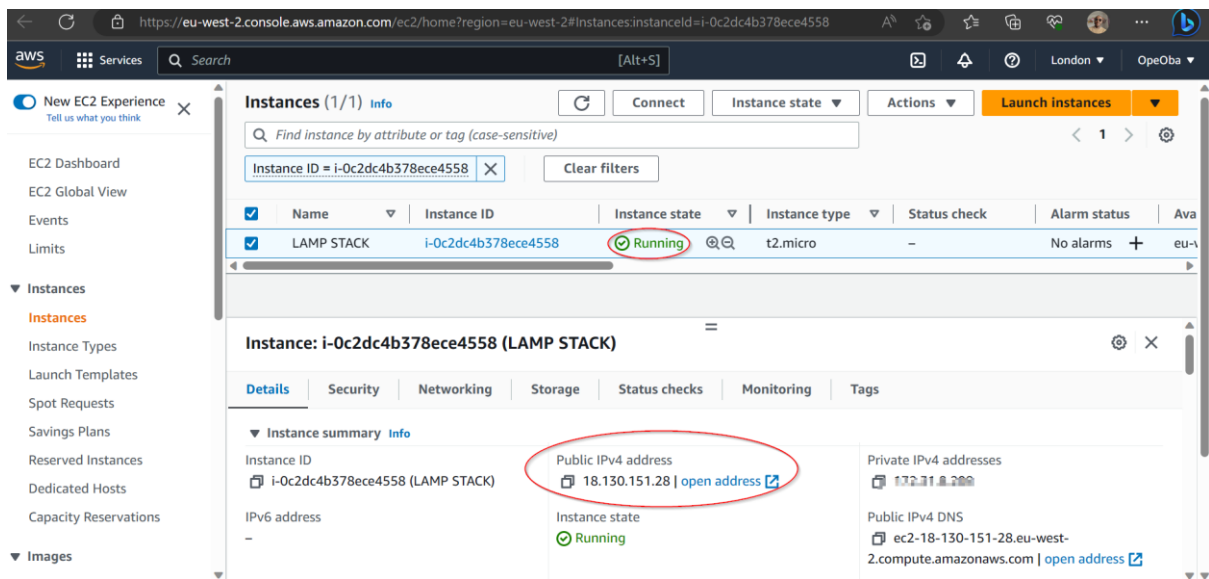
The screenshot shows the AWS Management Console interface for configuring an EC2 instance. The top navigation bar includes the AWS logo, 'Services', a search bar, and a keyboard shortcut '[Alt+S]'. The main content area is divided into sections. The 'Instance type' section is expanded, showing 't2.micro' as the selected instance type. A red circle with the number '1' and an arrow points to the 't2.micro' text. Below it, details for the instance type are listed: Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true. Pricing information for On-Demand Windows, RHEL, SUSE, and Linux is also shown. To the right, there is a toggle for 'All generations' and a link to 'Compare instance types'. The 'Key pair (login)' section is also expanded, showing instructions on how to use a key pair. A red circle with the number '2' and an arrow points to the instruction 'Create a key pair name (it can be any name)'. Below this, there is a dropdown menu for 'Key pair name - required' with 'Select' as the current value. To the right of the dropdown is a 'Create new key pair' button, which is highlighted by a red circle with the number '3' and an arrow.

- Leave the Network settings and Configure storage as Default and Launch instance



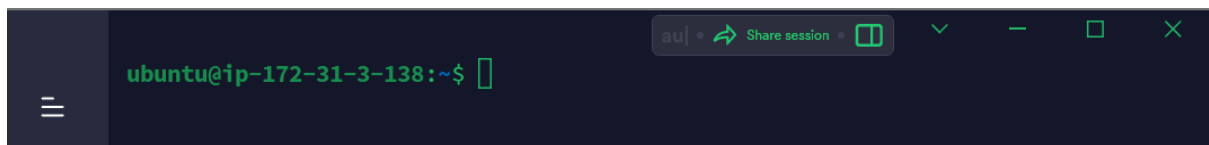
The screenshot shows the AWS Management Console interface for configuring an EC2 instance, specifically the 'Network settings' and 'Summary' sections. The 'Network settings' section is expanded, showing details for the network (vpc-0415751608806d554), subnet (No preference), and auto-assign public IP (Enable). The 'Firewall (security groups)' section is also expanded, showing a 'Launch...' button. A red box highlights this 'Launch...' button. A red arrow points from this button to the 'Launch instance' button in the 'Summary' section. The 'Summary' section shows the number of instances (1), the software image (Canonical, Ubuntu, 22.04 LTS), the virtual server type (t2.micro), the firewall (New security group), and the storage (volumes). The 'Launch instance' button is highlighted in orange, and a 'Review commands' link is also visible.

Done? If you get this, it shows your instance is up and running. Good Job, let's proceed now.



Copy your own Public IP as shown on the above screenshot, now it is time to use the console Yay!!! Open git bash or putty or termius, whichever console works for you, else download.

I am using Termius:



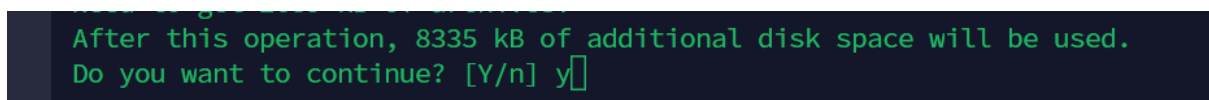
Type **clear**, to have a neat console and proceed. Now let us follow the steps:

STEP 1 – INSTALLING THE NGINX WEB SERVER

In order to display web pages to our site visitors, we are going to employ Nginx, a high-performance web server. We'll use the **apt** package manager to install this package.

```
sudo apt update  
sudo apt install nginx
```

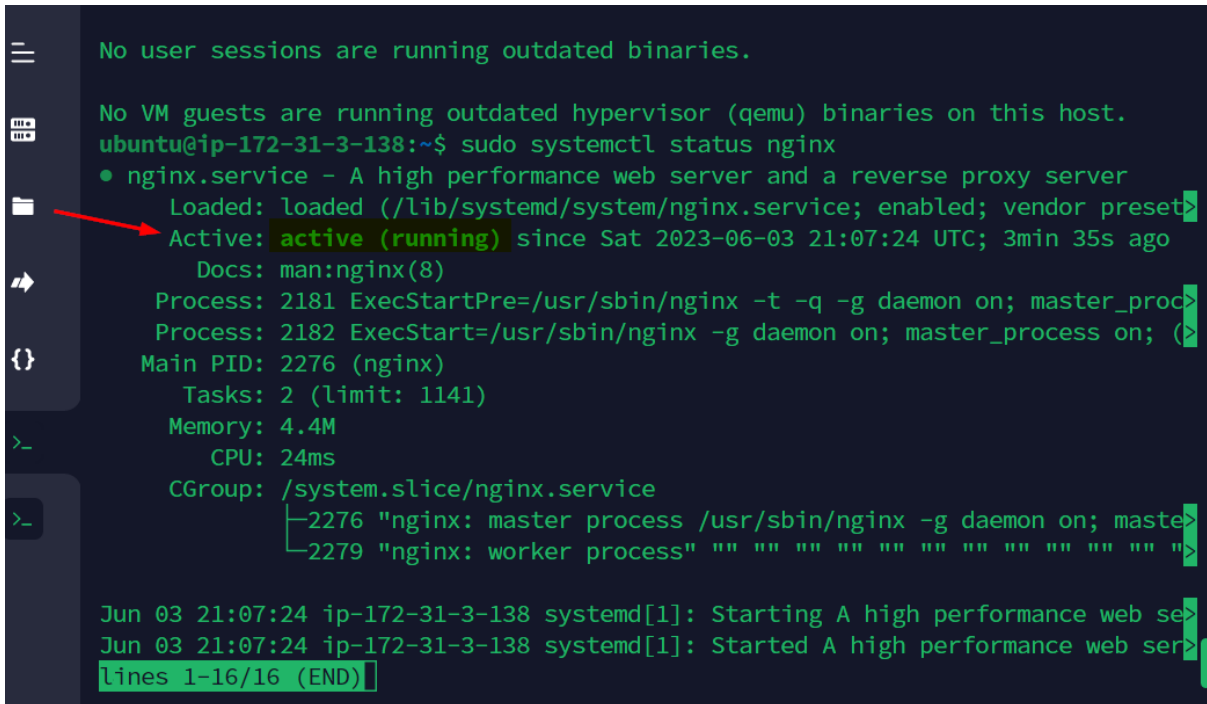
When prompted, enter **Y** to confirm that you want to install Nginx.



Once the installation is finished, the Nginx web server will be active and running on your Ubuntu 22.04 server.

To verify that nginx was successfully installed and is running as a service in Ubuntu, run:

```
$sudo systemctl status nginx
```



```
ubuntu@ip-172-31-3-138:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2023-06-03 21:07:24 UTC; 3min 35s ago
     Docs: man:nginx(8)
  Process: 2181 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on;
  Process: 2182 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
 Main PID: 2276 (nginx)
    Tasks: 2 (limit: 1141)
   Memory: 4.4M
      CPU: 24ms
   CGroup: /system.slice/nginx.service
           └─2276 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2279 "nginx: worker process"

Jun 03 21:07:24 ip-172-31-3-138 systemd[1]: Starting A high performance web server: nginx.
Jun 03 21:07:24 ip-172-31-3-138 systemd[1]: Started A high performance web server: nginx.
lines 1-16/16 (END)
```

If it is green and running, then you did everything correctly – you have just launched your first Web Server in the Clouds!

Before we can receive any traffic by our Web Server, we need to open TCP port 80 which is the default port that web browsers use to access web pages on the Internet

As we know, we have TCP port 22 open by default on our EC2 machine to access it via SSH, so we need to add a rule to EC2 configuration to open inbound connection through port 80:

Let us go back to our instance and add new rule which is port 80:

aws Services Search [Alt+S] London OpeOba

New EC2 Experience Tell us what you think

EC2 Dashboard
EC2 Global View
Events
Limits

Instances

Instances

Instance Types
Launch Templates
Spot Requests
Savings Plans

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Instance ID = i-0c2dc4b378ece4558 X Clear filters

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Ava
<input checked="" type="checkbox"/>	LAMP STACK	i-0c2dc4b378ece4558	Running	t2.micro	-	No alarms	eu-n

Instance: i-0c2dc4b378ece4558 (LAMP STACK)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary Info

Webstack-Implementation: x Launch an instance | EC2 | x Instances | EC2 Managem AWS Solutions Architect C x Step 1 — Installing Apache x

https://eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#Instances:instanceId=i-0c2dc4b378ece4558

aws Services Search [Alt+S] London OpeOba

New EC2 Experience Tell us what you think

EC2 Dashboard
EC2 Global View
Events
Limits

Instances

Instances

Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

Images

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Instance ID = i-0c2dc4b378ece4558 X Clear filters

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Ava
<input checked="" type="checkbox"/>	LAMP STACK	i-0c2dc4b378ece4558	Running	t2.micro	-	No alarms	eu-n

Instance: i-0c2dc4b378ece4558 (LAMP STACK)

Security groups

sg-04fa79dae5522564c (launch-wizard-22)

Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source

CloudShell Feedback Language

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

10°C Mostly cloudy

00:50 03/06/2023

Inbound rules Outbound rules Tags

You can now check network connectivity with Reachability Analyzer Run Reachability Analyzer X

Inbound rules (1/1)

Filter security group rules

Manage tags Edit inbound rules

<input checked="" type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol
<input checked="" type="checkbox"/>	-	sgr-03035e290b7724...	IPv4	SSH	TCP

EC2 > Security Groups > sg-04fa79dae5522564c - launch-wizard-22 > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-03035e290b77244ba	SSH	TCP	22	Custom <input type="text" value="0.0.0.0/0"/>		Delete

Add rule

Cancel Preview changes **Save rules**

aws Services Search [Alt+S] London Op

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

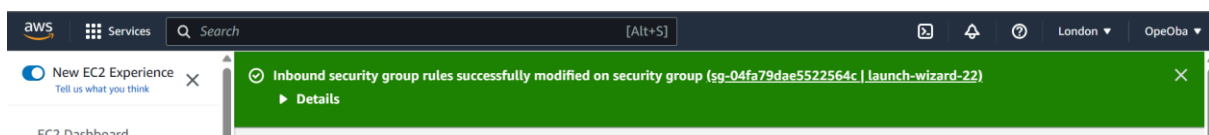
Inbound rules [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-03035e290b77244ba	SSH	TCP	22	Custom <input type="text" value="0.0.0.0/0"/>		Delete
-	Custom TCP	TCP	80	Anywh... <input type="text" value="0.0.0.0/0"/>		Delete

Add rule

Cancel Preview changes **Save rules**

If you have this below, then it is successfully added.



Our server is running and we can access it locally and from the Internet (Source 0.0.0.0/0 means 'from any IP address').

To come out of the page you are on your terminal, press **q**

First, let us try to check how we can access it locally in our Ubuntu shell, run:

```
curl http://localhost:80
or
curl http://127.0.0.1:80
```

As an output you can see some strangely formatted test, do not worry, we only made sure that our Nginx web service responds to ‘curl’ command with some payload.

Now it is time for us to test how our Nginx server can respond to requests from the Internet. Open a web browser of your choice and try to access following url

```
http://<Public-IP-Address>:80
```

Another way to retrieve your Public IP address, other than to check it in AWS Web console, is to use following command:

```
curl -s http://169.254.169.254/latest/meta-data/public-ipv4
```

The URL in browser shall also work if you do not specify port number since all web browsers use port 80 by default.

If you see following page, then your web server is now correctly installed and accessible through your firewall.



STEP 2 — INSTALLING MYSQL

Now that we have our web server up and running, we need to install a [Database Management System \(DBMS\)](#) to be able to store and manage data for our site in a [relational database](#). [MySQL](#) is a popular relational database management system used within PHP environments, so we will use it in this project.

Again, use ‘apt’ to acquire and install this software:

```
$ sudo apt install mysql-server
```

When prompted, confirm installation by typing **Y**, and then **ENTER**.

```
After this operation, 243 MB of additional disk space will be used.  
Do you want to continue? [Y/n] ☐
```

When the installation finished, log in to the MySQL console by typing:


```
$ sudo mysql
```

This will connect to the MySQL server as the administrative database user **root**, which is inferred using `sudo` when running this command. You should see output like this:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

It's recommended that you run a security script that comes pre-installed with MySQL. This script will remove some insecure default settings and lock down access to your database system. Before running the script you will set a password for the **root** user, using `mysql_native_password` as default authentication method. We're defining this user's password as **Lemp.1**.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'Lemp.1';
```

Exit the MySQL shell with:

```
mysql> exit
```

Start the interactive script by running:

```
$ sudo mysql_secure_installation
```

Note: Enabling this feature is something of a judgment call. If enabled, passwords which don't match the specified criteria will be rejected by MySQL with an error. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

Answer **y** for yes, or anything else to continue without enabling.

```
VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?
```

```
Press y|Y for Yes, any other key for No:
```

If your answer “yes”, you’ll be asked to select a level of password validation. Keep in mind that if you enter **2** for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words e.g **Lamp1**.

```
There are three levels of password validation policy:
```

```
LOW      Length >= 8
MEDIUM  Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and
dictionary                                file
```

```
Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Regardless of whether you chose to set up the **VALIDATE PASSWORD PLUGIN**, your server will next ask you to select and confirm a password for the MySQL **root** user. This is not to be confused with the **system root**. The **database root** user is an administrative user with full privileges over the database system. Even though the default authentication method for the MySQL root user dispenses the use of a password, **even when one is set**, you should define a strong password here as an additional safety measure. We’ll talk about this in a moment.

If you enabled password validation, you’ll be shown the password strength for the root password you just entered and your server will ask if you want to continue with that password. If you are happy with your current password, enter **y** for “yes” at the prompt:

```
Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for
Yes, any other key for No) : y
```

For the rest of the questions, press **y** and hit the **ENTER** key at each prompt. This will prompt you to change the root password, remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made.

When you are finished, test if you’re able to log in to the MySQL console by typing:

```
$ sudo mysql -p
```

Notice the **-p** flag in the command, which will prompt you for the password used after changing the **root** user password.

To exit the MySQL console, type:

```
mysql> exit
```

Notice that you need to provide a password to connect as the **root** user.

```
ubuntu@ip-172-31-3-138:~$ sudo mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye
ubuntu@ip-172-31-3-138:~$
```

Our MySQL server is now installed and secured. Next, we will install PHP.

STEP 3 — INSTALLING PHP

We have Nginx installed to serve our content and MySQL installed to store and manage our data. Now you can install [PHP](#) to process code and generate dynamic content for the web server.

Nginx requires an external program to handle PHP processing and act as a bridge between the PHP interpreter itself and the web server. This allows for a better overall performance in most PHP-based websites, but it requires additional configuration. We will need to install **php-fpm**, which stands for “PHP fastCGI process manager”, and tell Nginx to pass PHP requests to this software for processing. Additionally, we will need **php-mysql**, a PHP module that allows PHP to communicate with MySQL-based databases. Core PHP packages will automatically be installed as dependencies.

To install these 2 packages at once, run:

```
sudo apt install php-fpm php-mysql
```

When prompted, type **Y** and press **ENTER** to confirm installation.

```
Setting up php8.1-fpm (8.1.2-1ubuntu2.11) ...  
  
Creating config file /etc/php/8.1/fpm/php.ini with new version  
Created symlink /etc/systemd/system/multi-user.target.wants/php8.1-fpm.service  
→ /lib/systemd/system/php8.1-fpm.service.  
Setting up php-fpm (2:8.1+92ubuntu1) ...  
Processing triggers for man-db (2.10.2-1) ...  
Processing triggers for php8.1-cli (8.1.2-1ubuntu2.11) ...  
Processing triggers for php8.1-fpm (8.1.2-1ubuntu2.11) ...  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
>_  
No services need to be restarted.  
  
>_  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-3-138:~$
```

Once the installation finishes, we can run the following command to confirm our PHP version:

```
php -v
```

```
ubuntu@ip-172-31-3-138:~$ php -v  
PHP 8.1.2-1ubuntu2.11 (cli) (built: Feb 22 2023 22:56:18) (NTS)  
Copyright (c) The PHP Group  
Zend Engine v4.1.2, Copyright (c) Zend Technologies  
with Zend OPcache v8.1.2-1ubuntu2.11, Copyright (c), by Zend Technologies  
ubuntu@ip-172-31-3-138:~$
```

You now have your PHP components installed. Next, you will configure Nginx to use them.

STEP 4 — CONFIGURING NGINX TO USE PHP PROCESSOR

When using the Nginx web server, we can create server blocks to encapsulate configuration details and host more than one domain on a single server. In this guide, we will use **projectLEMP** as an example domain name.

On Ubuntu 22.04, Nginx has one server block enabled by default and is configured to serve documents out of a directory at `/var/www/html`. While this works well for a single site, it can become difficult to manage if you are hosting multiple sites. Instead of modifying `/var/www/html`, we'll create a directory structure within `/var/www` for the **your_domain** website, leaving `/var/www/html` in place as the default directory to be served if a client request does not match any other sites.

Create the root web directory for **your_domain** as follows:

```
sudo mkdir /var/www/projectLEMP
```

Next, assign ownership of the directory with the \$USER environment variable, which will reference your current system user:

```
sudo chown -R $USER:$USER /var/www/projectLEMP
```

Then, open a new configuration file in Nginx's **sites-available** directory using your preferred command-line editor. Here, we'll use **nano**:

```
sudo nano /etc/nginx/sites-available/projectLEMP
```

This will create a new blank file. Paste in the following bare-bones configuration:
the text:

```
#!/etc/nginx/sites-available/projectLEMP

server {
    listen 80;
    server_name projectLEMP www.projectLEMP;
    root /var/www/projectLEMP;

    index index.html index.htm index.php;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ /\.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Press **ctrl + x** to save and Enter

Here's what each of these directives and location blocks do:

- **listen** — Defines what port Nginx will listen on. In this case, it will listen on port **80**, the default port for HTTP.
- **root** — Defines the document root where the files served by this website are stored.

- **index** — Defines in which order Nginx will prioritize index files for this website. It is a common practice to list **index.html** files with a higher precedence than **index.php** files to allow for quickly setting up a maintenance landing page in PHP applications. You can adjust these settings to better suit your application needs.
- **server_name** — Defines which domain names and/or IP addresses this server block should respond for. **Point this directive to your server's domain name or public IP address.**
- **location /** — The first location block includes a **try_files** directive, which checks for the existence of files or directories matching a URI request. If Nginx cannot find the appropriate resource, it will return a 404 error.
- **location ~ /\.php\$** — This location block handles the actual PHP processing by pointing Nginx to the fastcgi-php.conf configuration file and the **php7.4-fpm.sock** file, which declares what socket is associated with **php-fpm**.
- **location ~ /\.ht** — The last location block deals with **.htaccess** files, which Nginx does not process. By adding the deny all directive, if any **.htaccess** files happen to find their way into the document root, they will not be served to visitors.

Activate your configuration by linking to the config file from Nginx's **sites-enabled** directory:

```
sudo ln -s /etc/nginx/sites-available/projectLEMP
/etc/nginx/sites-enabled/
```

This will tell Nginx to use the configuration next time it is reloaded. You can test your configuration for syntax errors by typing:

```
sudo nginx -t
```

You shall see following message:

```
ubuntu@ip-172-31-3-138:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
ubuntu@ip-172-31-3-138:~$
```

If any errors are reported, go back to your configuration file to review its contents before continuing.

We also need to disable default Nginx host that is currently configured to listen on port 80, for this run:

```
sudo unlink /etc/nginx/sites-enabled/default
```

When you are ready, reload Nginx to apply the changes:

```
sudo systemctl reload nginx
```

Your new website is now active, but the web root `/var/www/projectLEMP` is still empty. Create an `index.html` file in that location so that we can test that your new server block works as expected:

```
sudo echo 'Hello! This is LEMP from hostname' $(curl -s  
http://169.254.169.254/latest/meta-data/public-hostname) 'with  
public IP' $(curl -s http://169.254.169.254/latest/meta-  
data/public-ipv4) > /var/www/projectLEMP/index.html
```

Now go to your browser and try to open your website URL using IP address:

```
http://<Public-IP-Address>:80
```

If you see the text from `'echo'` command you wrote to `index.html` file, then it means your Nginx site is working as expected.

In the output you will see your server's public hostname (DNS name) and public IP address. You can also access your website in your browser by public DNS name, not only by IP – try it out, the result must be the same (port is optional)

```
http://<Public-DNS-Name>:80
```

You can leave this file in place as a temporary landing page for your application until you set up an `index.php` file to replace it. Once you do that, remember to remove or rename the `index.html` file from your document root, as it would take precedence over an `index.php` file by default.

Your LEMP stack is now fully configured. Time to go and create a PHP script to test that Nginx is in fact able to handle `.php` files within our newly configured website.

STEP 5 – TESTING PHP WITH NGINX

Our LEMP stack should now be completely set up.

At this point, our LEMP stack is completely installed and fully operational.

We can test it to validate that Nginx can correctly hand `.php` files off to our PHP processor.

We can do this by creating a test PHP file in our document root. Open a new file called `info.php` within your document root in your text editor:

```
sudo nano /var/www/projectLEMP/info.php
```

Type or paste the following lines into the new file. This is valid PHP code that will return information about your server:

```
<?php  
phpinfo();
```

You can now access this page in your web browser by visiting the domain name or public IP address you've set up in your Nginx configuration file, followed by `/info.php`:

```
http://`server_domain_or_IP`/info.php
```

You will see a web page containing detailed information about your server:

<div> <div>←</div> <div>↺</div> <div> <div>⚠ Not secure</div> <div>18.169.238.210/info.php</div> <div> <div>📶</div> <div>★</div> <div>👤</div> <div>⋮</div> <div>💬</div> </div> </div> </div>	
<div>PHP Version 8.1.2-1ubuntu2.11</div>	
System	Linux ip-172-31-3-138 5.19.0-1025-aws #26~22.04.1-Ubuntu
Build Date	Feb 22 2023 22:56:18
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/fpm
Loaded Configuration File	/etc/php/8.1/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/fpm/conf.d
Additional .ini files parsed	/etc/php/8.1/fpm/conf.d/10-mysqlnd.ini, /etc/php/8.1/fpm/conf.d/20-calendar.ini, /etc/php/8.1/fpm/conf.d/20-curl.ini, /etc/php/8.1/fpm/conf.d/20-ffi.ini, /etc/php/8.1/fpm/conf.d/20-gd.ini, /etc/php/8.1/fpm/conf.d/20-gettext.ini, /etc/php/8.1/fpm/conf.d/20-iconv.ini, /etc/php/8.1/fpm/conf.d/20-intl.ini, /etc/php/8.1/fpm/conf.d/20-ldap.ini, /etc/php/8.1/fpm/conf.d/20-mbstring.ini, /etc/php/8.1/fpm/conf.d/20-mcrypt.ini, /etc/php/8.1/fpm/conf.d/20-mysql.ini, /etc/php/8.1/fpm/conf.d/20-mysqli.ini, /etc/php/8.1/fpm/conf.d/20-pdo_mysql.ini, /etc/php/8.1/fpm/conf.d/20-readline.ini, /etc/php/8.1/fpm/conf.d/20-redis.ini, /etc/php/8.1/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.1/fpm/conf.d/20-sysvshm.ini, /etc/php/8.1/fpm/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled

After checking the relevant information about your PHP server through that page, it's best to remove the file you created as it contains sensitive information about your PHP environment and your Ubuntu server. You can use `rm` to remove that file:

```
sudo rm /var/www/your_domain/info.php
```

You can always regenerate this file if you need it later.

STEP 6 – RETRIEVING DATA FROM MYSQL DATABASE WITH PHP

In this step we will create a test database (DB) with simple “To do list” and configure access to it, so the Nginx website would be able to query data from the DB and display it.

At the time of this writing, the native MySQL PHP library `mysqlnd` doesn't support `caching_sha2_authentication`, the default authentication method for MySQL 8. We'll need to create a new user with the `mysql_native_password` authentication method in order to be able to connect to the MySQL database from PHP.

We will create a database named **lemp_database** and a user named **lemp_user**, but you can replace these names with different values.

First, connect to the MySQL console using the **root** account:

```
sudo mysql -p
```

To create a new database, run the following command from your MySQL console:

```
mysql> CREATE DATABASE `lemp_database`;
```

Now you can create a new user and grant him full privileges on the database you have just created.

The following command creates a new user named `example_user`, using `mysql_native_password` as default authentication method. We're defining this user's password as `password`, but you should replace this value with a secure password of your own choosing.

```
Mysql> CREATE USER 'lemp_user'@'%' IDENTIFIED WITH mysql_native_password BY 'password';
```

Now we need to give this user permission over the `lemp_database` database:

```
mysql> GRANT ALL ON lemp_database.* TO 'lemp_user'@'%';
```

This will give the **lemp_user** user full privileges over the **lemp_database** database, while preventing this user from creating or modifying other databases on your server.

Now exit the MySQL shell with:

```
mysql> exit
```

We can test if the new user has the proper permissions by logging in to the MySQL console again, this time using the custom user credentials:

```
mysql -u lemp_user -p
```

```
ubuntu@ip-172-31-3-138:~$ mysql -u lemp_user -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Notice the **-p** flag in this command, which will prompt you for the password used when creating the **lemp_user** user. After logging in to the MySQL console, confirm that you have access to the **lemp_database** database:

```
mysql> SHOW DATABASES;
```

This will give you the following output:

```
mysql> SHOW DATABASES;
+-----+
| Database                |
+-----+
| information_schema      |
| lemp_database           |
| performance_schema      |
+-----+
3 rows in set (0.03 sec)

mysql> 
```

Next, we'll create a test table named **todo_list**. From the MySQL console, run the following statement:

```
CREATE TABLE lemp_database.todo_list (item_id INT AUTO_INCREMENT, content
VARCHAR(255), PRIMARY KEY(item_id));
```

Insert a few rows of content in the test table. You might want to repeat the next command a few times, using different VALUES

```
mysql> INSERT INTO lemp_database.todo_list (content) VALUES ("My first important item is Devops");
```

To confirm that the data was successfully saved to your table, run:

```
mysql> SELECT * FROM lemp_database.todo_list;
```

You'll see the following output:

```
mysql> SELECT * FROM lemp_database.todo_list;
+-----+-----+
| item_id | content |
+-----+-----+
|      1 | My first important item is Devops |
|      2 | My second important item is Football |
|      3 | My third important item is Holiday |
|      4 | Lastly, Enjoyment and Peace all day |
+-----+-----+
4 rows in set (0.00 sec)
```

After confirming that we have valid data in our test table, we can exit the MySQL console:

```
mysql> exit
```

Now we will create a PHP script that will connect to MySQL and query for our content. Create a new PHP file in your custom web root directory using your preferred editor. We'll use nano for that:

```
nano /var/www/projectLEMP/todo_list.php
```

The following PHP script connects to the MySQL database and queries for the content of the **todo_list** table, displays the results in a list. If there is a problem with the database connection, it will throw an exception.

Copy this content into your **todo_list.php** script:

```
<?php
$user = "lemp_user";
$password = "password";
$database = "lemp_database";
$table = "todo_list";

try {
```

```

$db = new PDO("mysql:host=localhost;dbname=$database", $user,
$password);
echo "<h2>TODO</h2><ol>";
foreach($db->query("SELECT content FROM $table") as $row) {
    echo "<li>" . $row['content'] . "</li>";
}
echo "</ol>";
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}

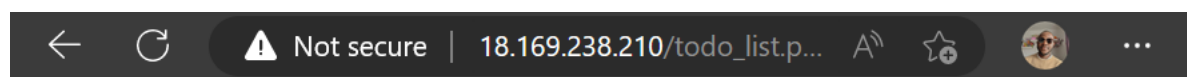
```

Save and close the file when you are done editing.

You can now access this page in your web browser by visiting the domain name or public IP address configured for your website, followed by `/todo_list.php`:

```
http://<Public_domain_or_IP>/todo_list.php
```

You should see a page like this, showing the content you've inserted in your test table:



TODO

1. My first important item is Devops
2. My second important item is Football
3. My third important item is Holiday
4. Lastly, Enjoyment and Peace all day

That means our PHP environment is ready to connect and interact with our MySQL server.

Congratulations!

In this project, we have built a flexible foundation for serving PHP websites and applications to our visitors, using Nginx as web server and MySQL as database management system.

PS: Remember to terminate your EC2 instance.

