LAB 1

Popular Cloud Computing Services

Full name: Trương Đặng Trúc Lâm

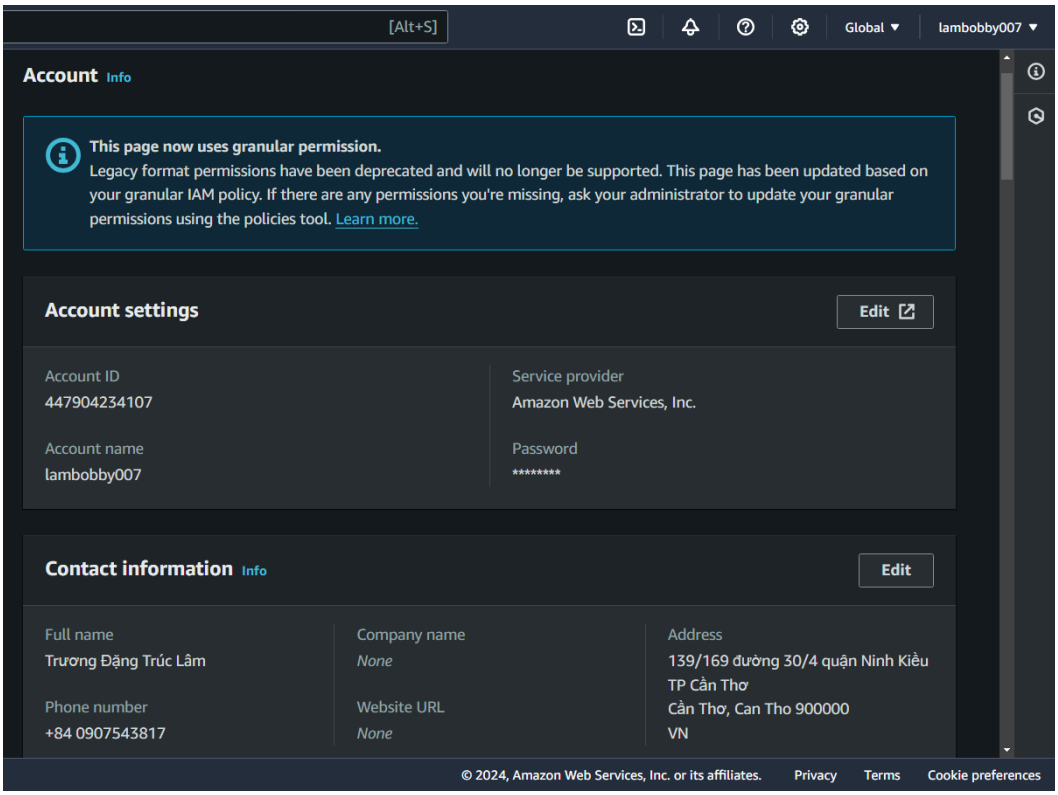
Student ID: B2111933

- Note: screenshots need to be clear and good-looking; submissions must be in PDF format.

This lab aims to help students get familiar with popular public cloud services. Please choose one (AWS, Microsoft Azure, or Google Cloud) to practice.

I choose Amazon Web Services (AWS).

1. **Amazon Web Services (AWS)**
   - Register for an account to try Amazon Web Services (AWS - https://portal.aws.amazon.com/billing/signup#/start), a credit/debit card is required to activate your account. We can try some AWS services free of charge within certain usage limits. AWS calls this the AWS Free Tier.



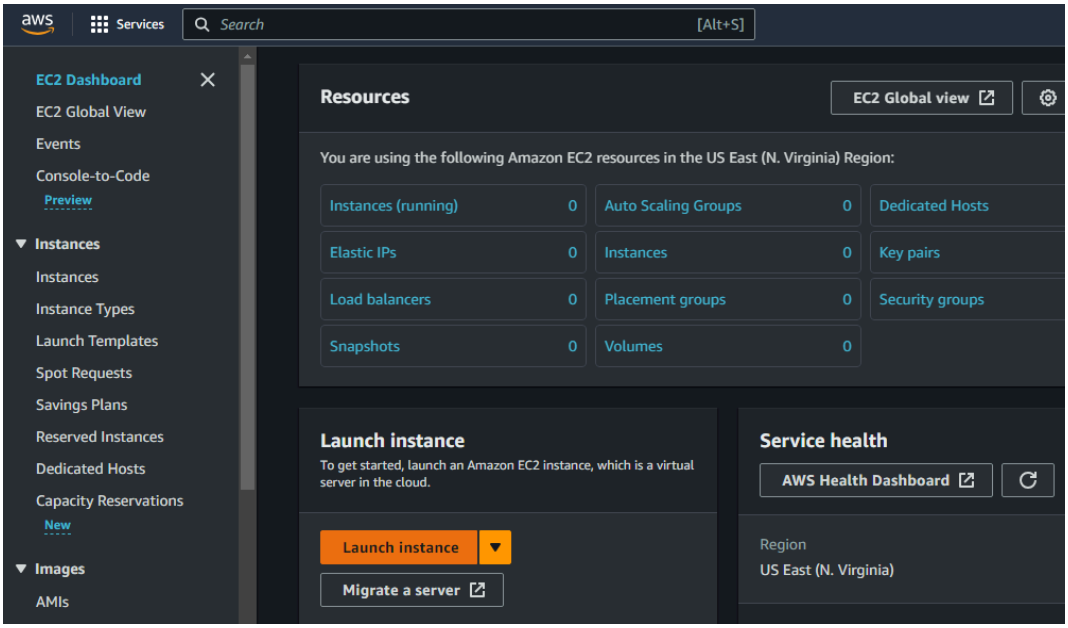
This is my AWS account.

1.1. Launch a Linux Virtual Machine using EC2

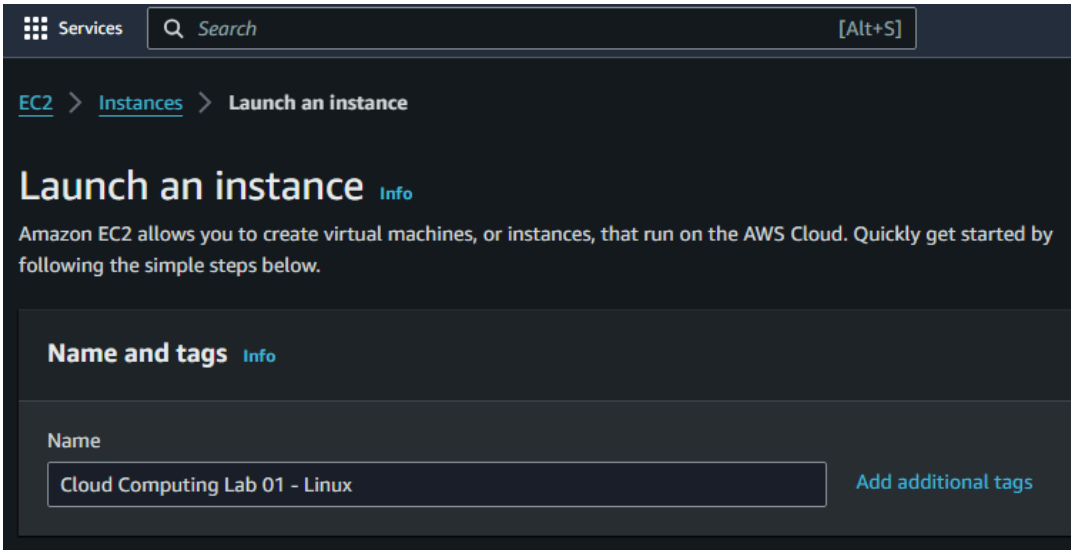
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

(Take screenshots to show that you finish every step on the tutorial)

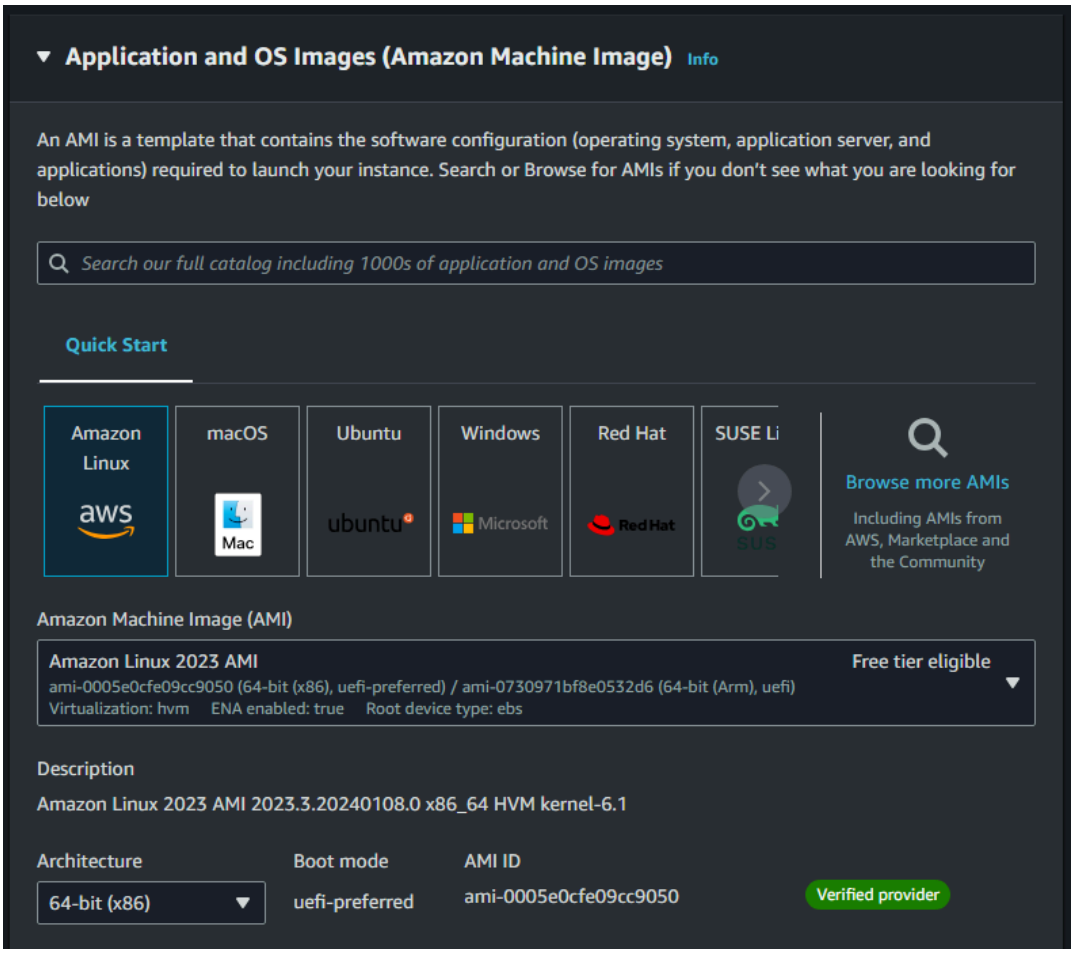
1.1.1. Launch an instance



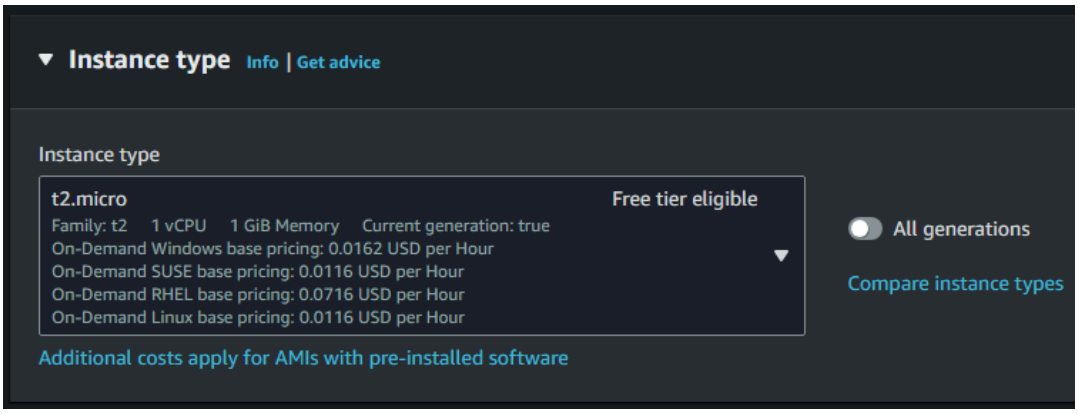
Open the **Amazon EC2 console** at https://console.aws.amazon.com/ec2/
From the **EC2 console dashboard**, choose **Launch instance**.



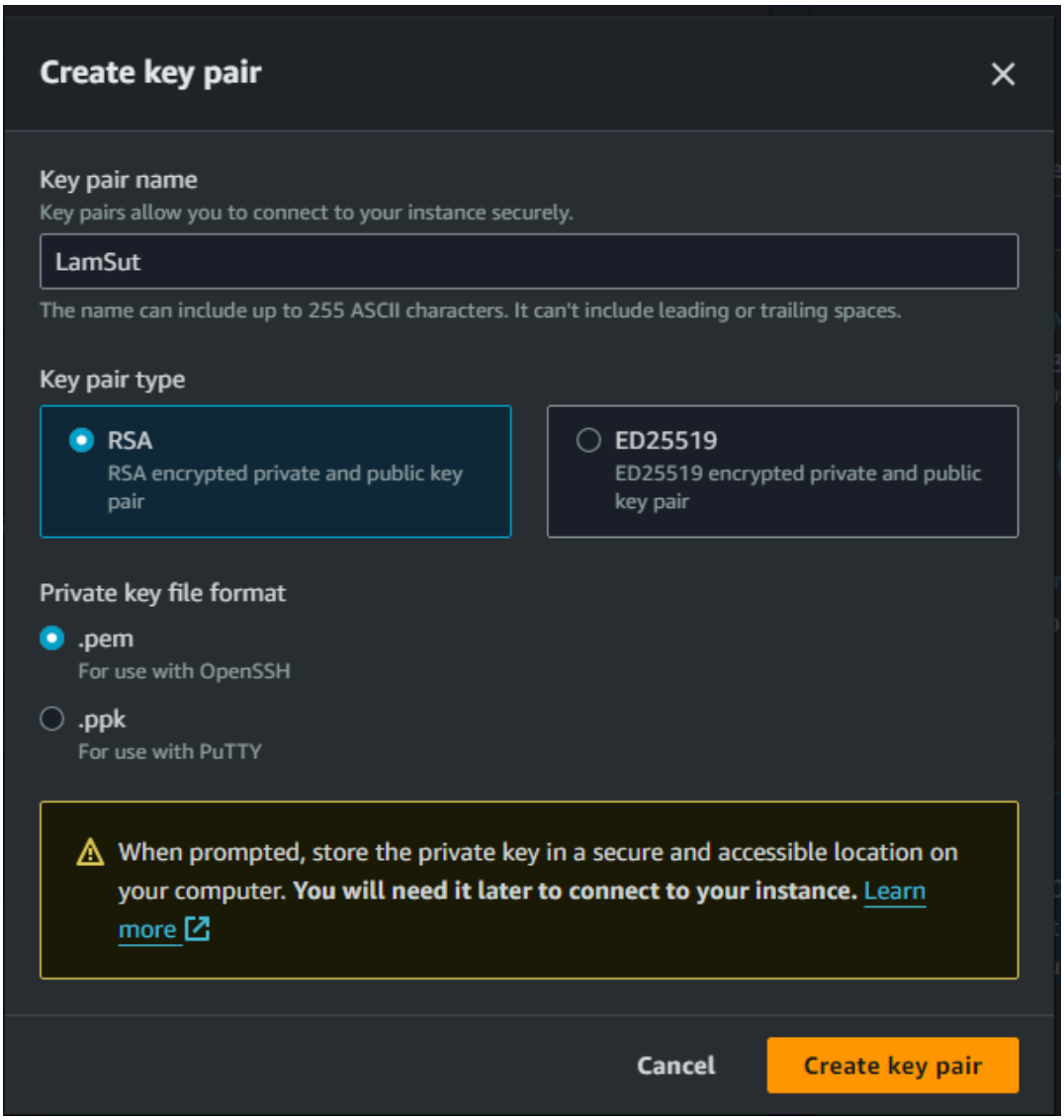
Enter a descriptive name for your instance.

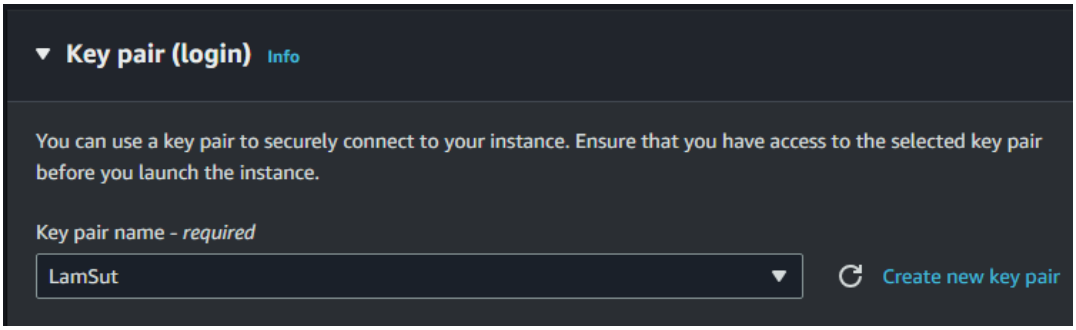


From **Quick Start**, choose **Amazon Linux**.
Continue to **Amazon Machine Image (AMI)**, select an **HVM version of Amazon Linux 2**. Notice that these **AMIs** are marked **Free Tier eligible**.
(**AL2023** is the successor to **Amazon Linux 2**)

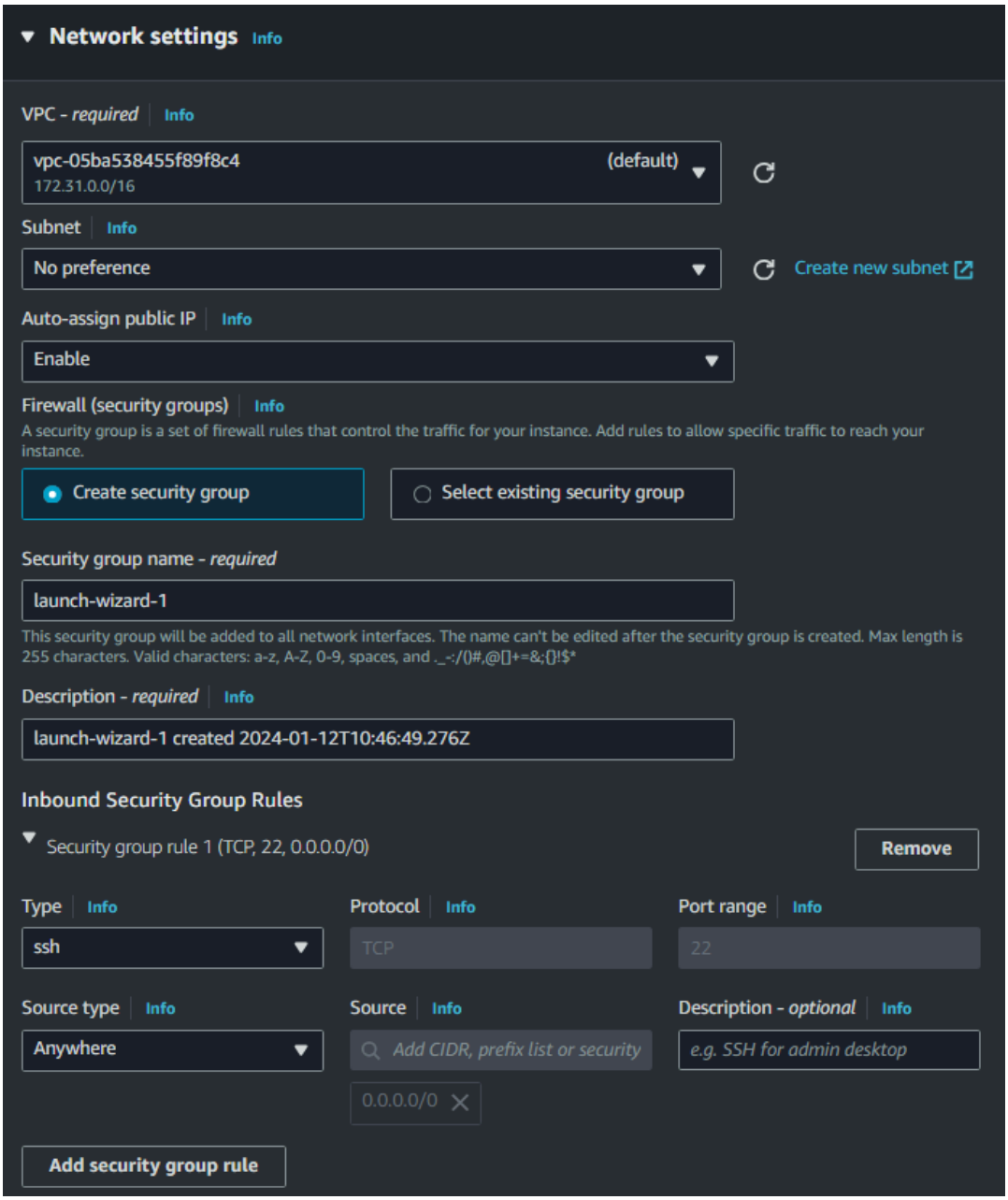


Under **Instance type**, choose the **t2.micro** instance type (default).



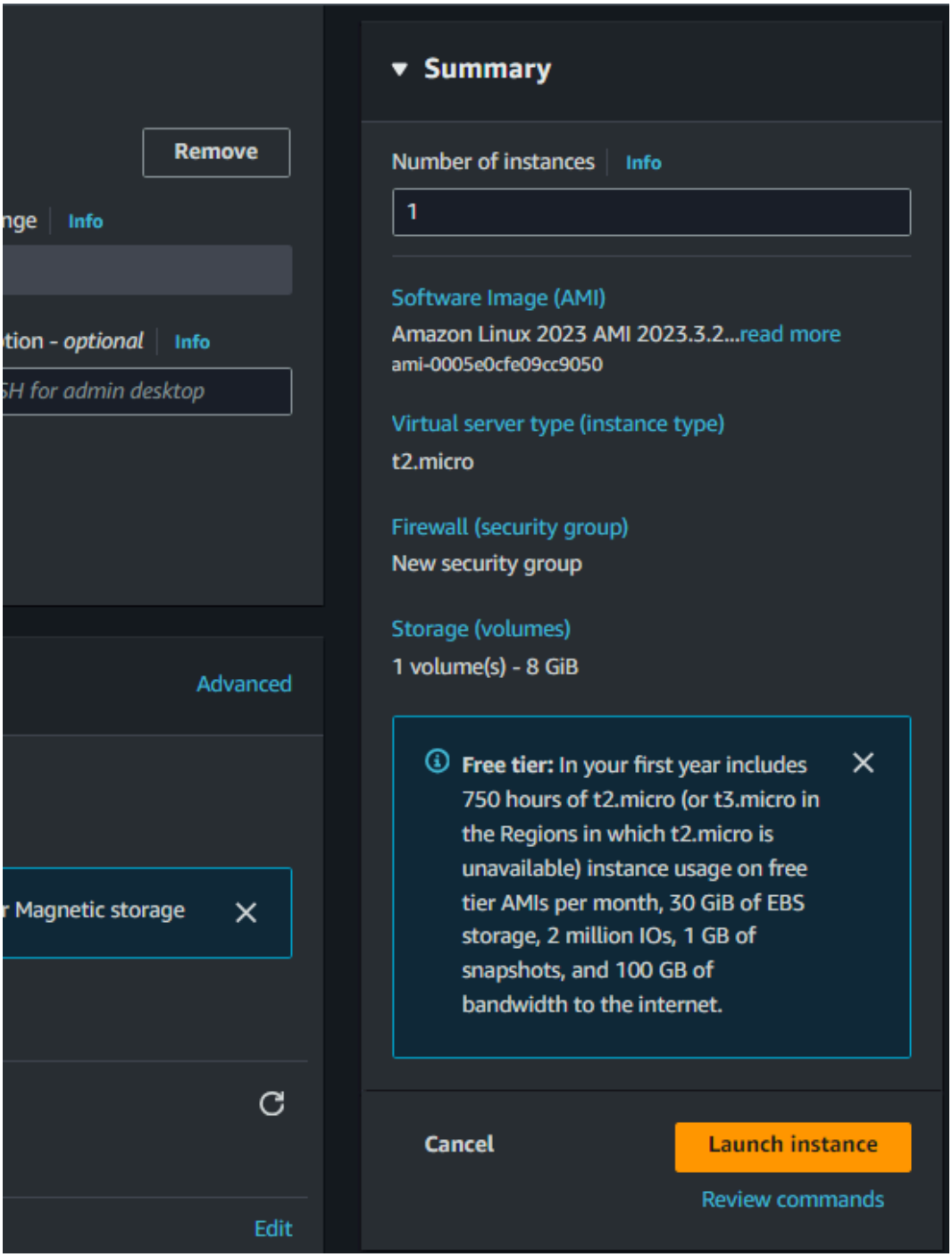


Under **Key pair (login)**, for **Key pair name**, choose the **key pair that you created when getting set up**.

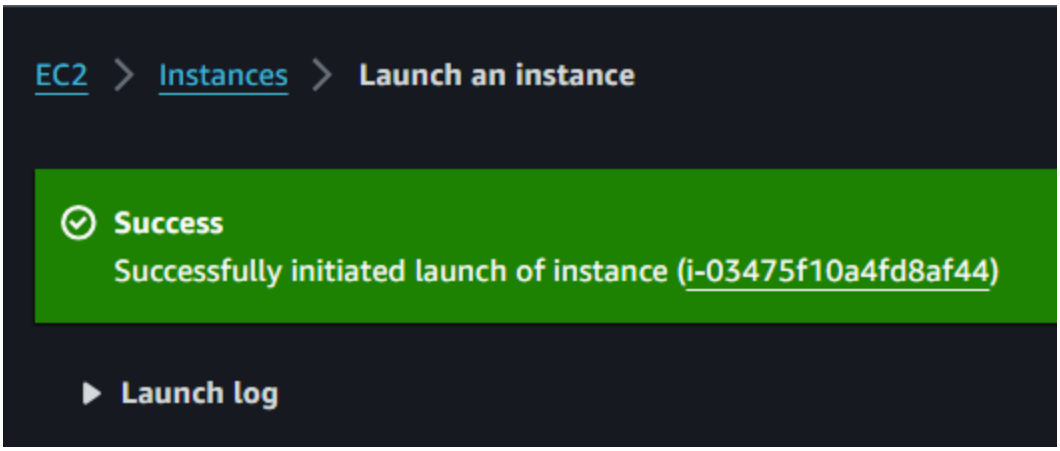


From **Network settings**, **choose Edit**. For **Security group name**, you'll see that the wizard created and selected a security group for you. You can use this security group.

Keep the default selections for the other configuration settings for your instance.

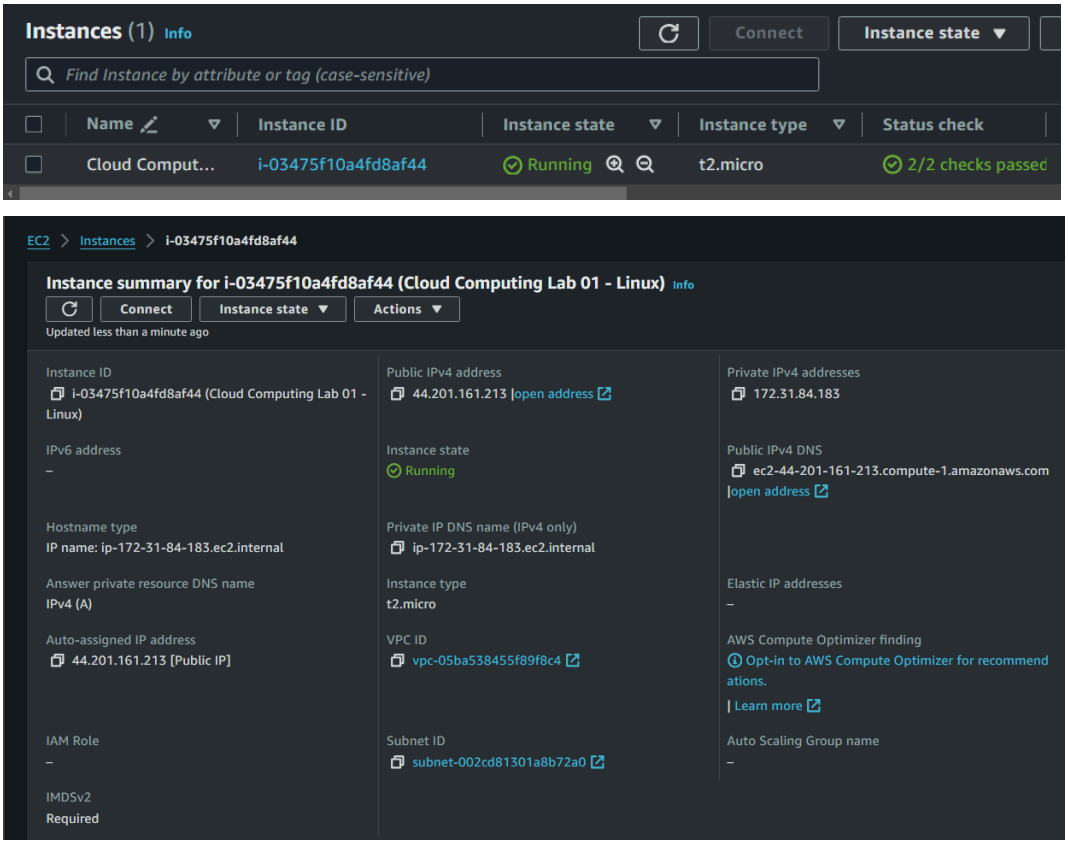


Review a **summary of your instance configuration** in the **Summary** panel, and when you're ready, choose **Launch instance**.



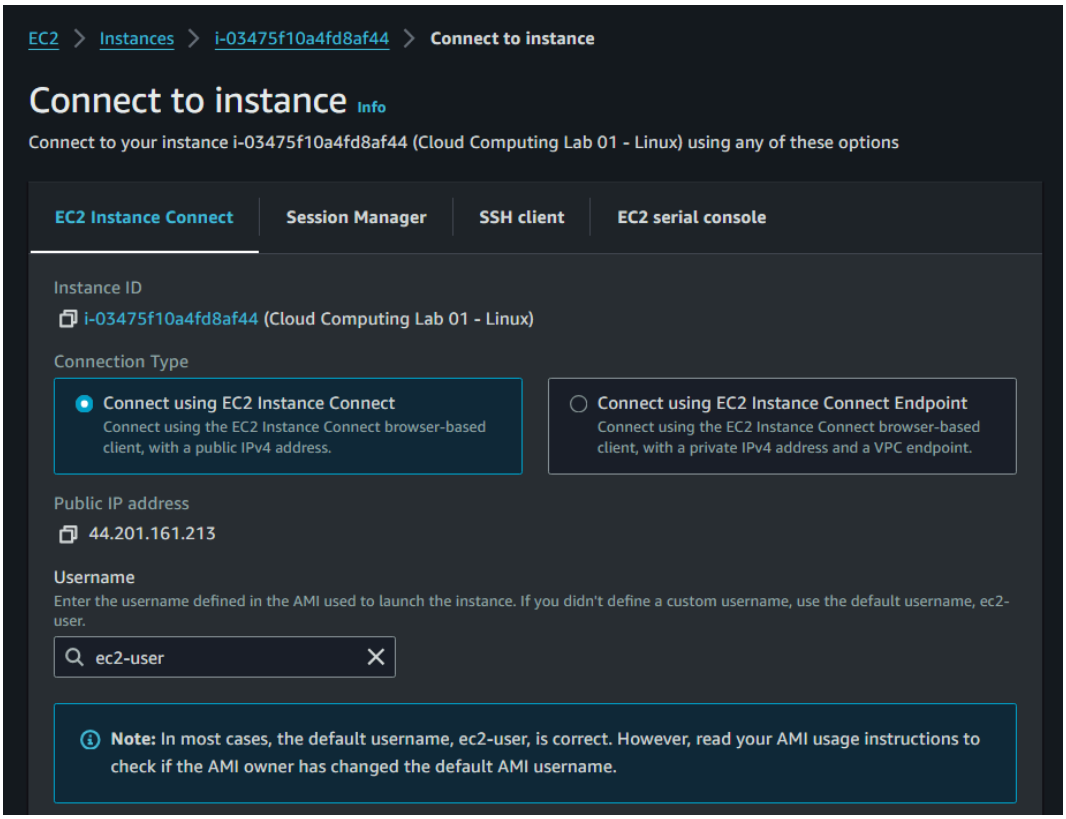
Successfully initiated launch of instance.

Scroll to the end of page and choose **View all instances**.



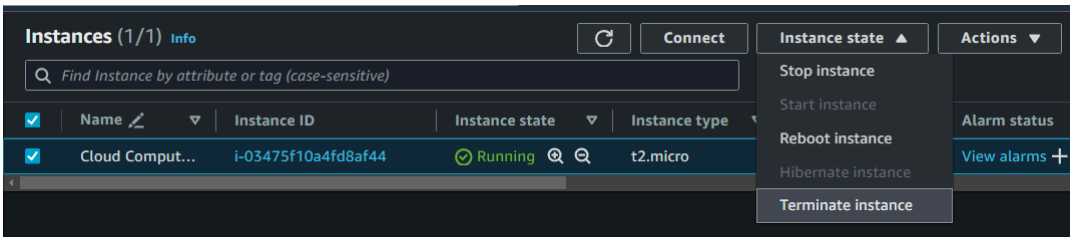
This is the **Instance summary**.

1.1.2. Connect to your instance

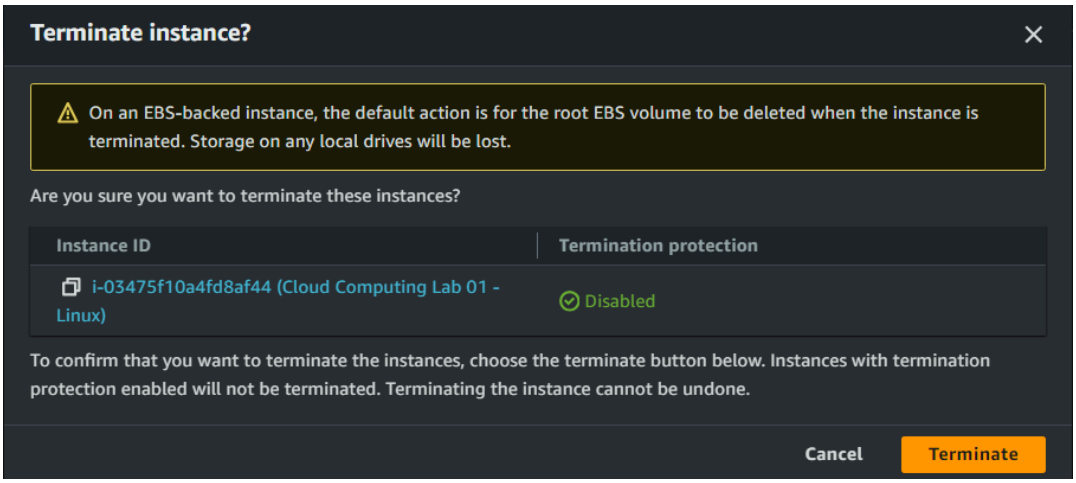


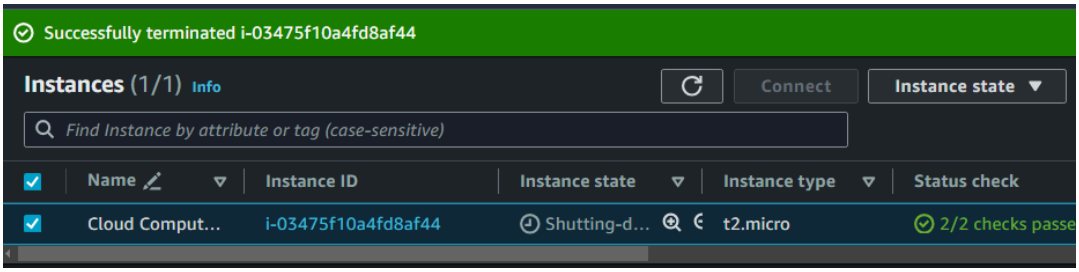
Connect to instance and check the result.

1.1.3. Terminate your instance



In the **navigation pane**, choose **Instances**. In the list of instances, select the instance. Choose **Instance state**, **Terminate Instance**.





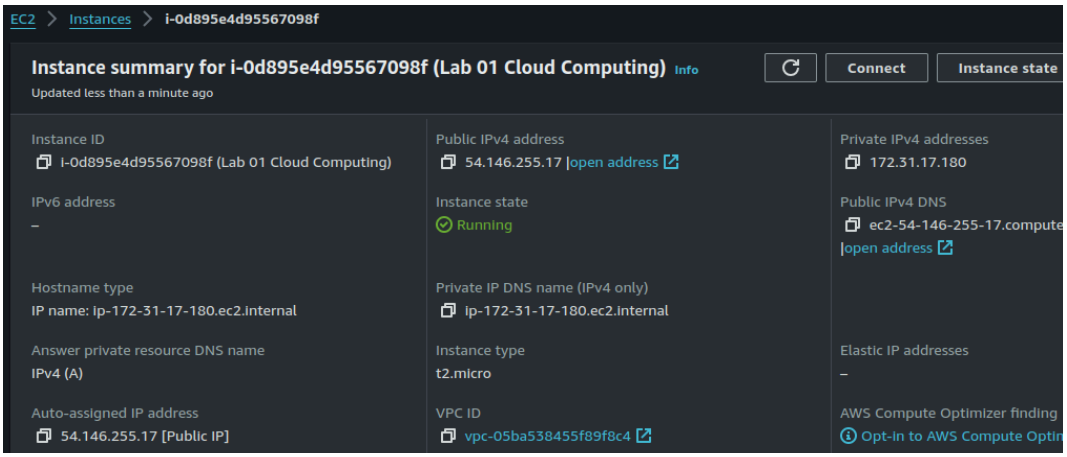
Terminate instance successfully.

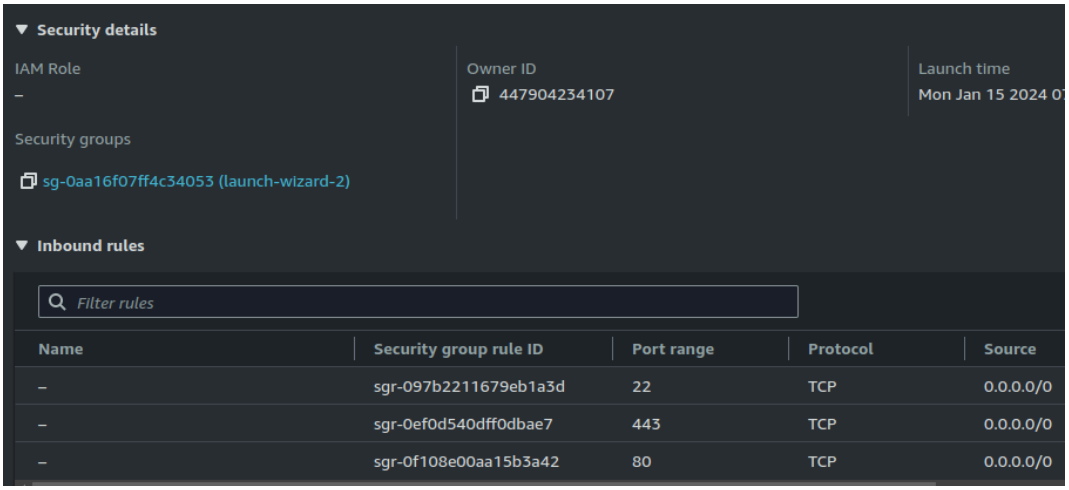
1.2. Install a LAMP Web Server with the Amazon Linux AMI

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-lamp-amazon-linux-2.html

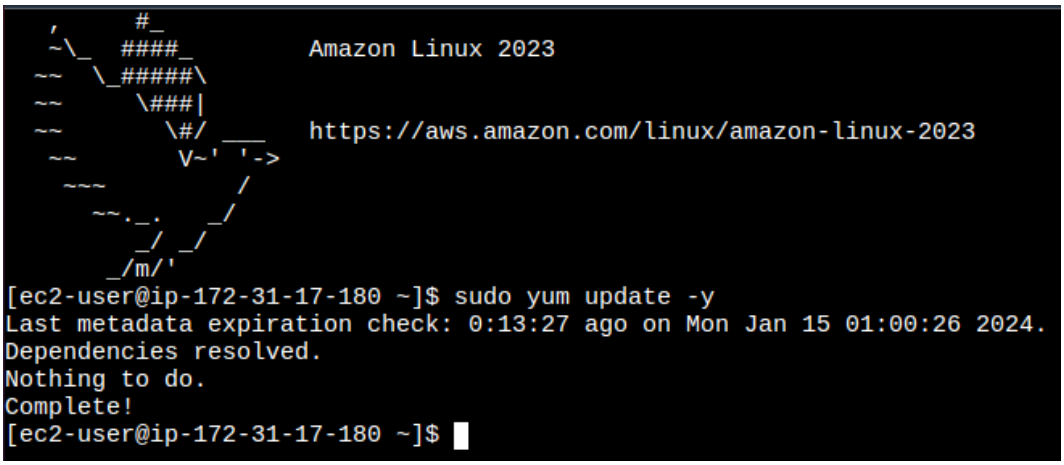
(Take screenshots to show that you finish every step on the tutorial)

1.2.1. Prepare your LAMP Web Server





Create an instance and configure your security group to allow **SSH (port 22)**, **HTTP (port 80)**, **and HTTPS (port 443)** connections.



Connect to your instance and update with command: **$ sudo yum update**

```
[ec2-user@ip-172-31-17-180 ~]$ sudo dnf install mariadb105-server
Last metadata expiration check: 0:25:19 ago on Mon Jan 15 01:00:26 2
Dependencies resolved.
==========================================================================
 Package                                                  Architecture
==========================================================================
Installing:
 mariadb105-server                                        x86_64
Installing dependencies:
 mariadb-connector-c                                      x86_64
 mariadb-connector-c-config                               noarch
 mariadb105                                               x86_64
 mariadb105-common                                        x86_64
 mariadb105-errmsg                                        x86_64
 mysql-selinux                                            noarch
```

Install **MariaDB 10.05** on **Amazon Linux 2023** with command:

$ sudo dnf install mariadb105-server

```
[ec2-user@ip-172-31-17-180 ~]$ sudo dnf install php8.2
Last metadata expiration check: 0:30:08 ago on Mon Jan 15 01:00:26 202
Dependencies resolved.
==========================================================================
 Package                                        Architecture
==========================================================================
Installing:
 php8.2                                         x86_64
Installing dependencies:
 apr                                            x86_64
 apr-util                                       x86_64
 generic-logos-httpd                            noarch
```

Install **PHP 8.2** on **Amazon Linux 2023** with command:

$ sudo dnf install mariadb105-server

```
[ec2-user@ip-172-31-17-180 ~]$ sudo yum install -y httpd
Last metadata expiration check: 0:32:41 ago on Mon Jan 15 01:00:
Package httpd-2.4.58-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-17-180 ~]$
```

Use the **yum install** command to install multiple software packages and all related dependencies at the same time:

$ sudo yum install -y httpd

```
[ec2-user@ip-172-31-17-180 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-17-180 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/htt
[ec2-user@ip-172-31-17-180 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enal
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Mon 2024-01-15 01:40:01 UTC
       Docs: man:httpd.service(8)
   Main PID: 28325 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Reques
      Tasks: 177 (limit: 1114)
     Memory: 13.2M
        CPU: 92ms
     CGroup: /system.slice/httpd.service
             ├─28325 /usr/sbin/httpd -DFOREGROUND
```

Start the **Apache web server**:

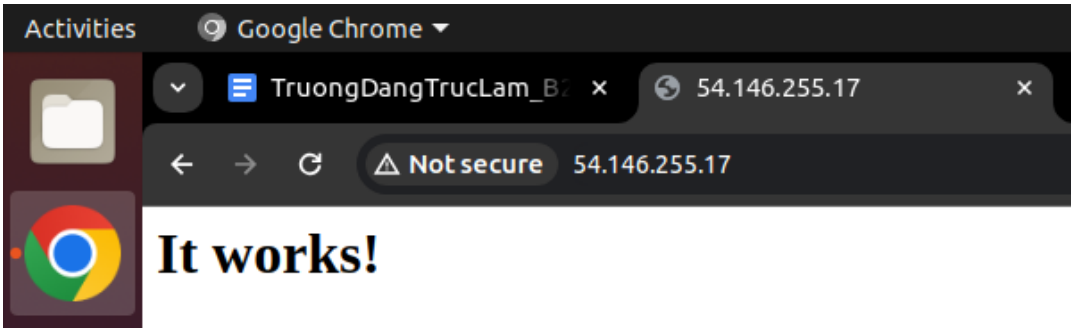
$ sudo systemctl start httpd

Configure the **Apache web server** to start at each system boot:

$ sudo systemctl enable httpd

View the status of the **Apache web server:**

$ sudo systemctl status httpd



Test your web server. In a web browser, type the **public DNS address (or the public IP address)** of your **instance.**

To allow the ec2-user account to manipulate files in this directory, you must modify the ownership and permissions of the directory.

```
[ec2-user@ip-172-31-17-180 ~]$ sudo usermod -a -G apache ec2-user
[ec2-user@ip-172-31-17-180 ~]$ exit
logout
```

Add your user (in this case, **ec2-user**) to the **apache group** then **log out**.

```
[ec2-user@ip-172-31-17-180 ~]$ groups
ec2-user adm wheel apache systemd-journal
[ec2-user@ip-172-31-17-180 ~]$ sudo chown -R ec2-user:apache /var/www
[ec2-user@ip-172-31-17-180 ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;
[ec2-user@ip-172-31-17-180 ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
[ec2-user@ip-172-31-17-180 ~]$
```

Type these commands:

$ sudo chown -R ec2-user:apache /var/www

$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;

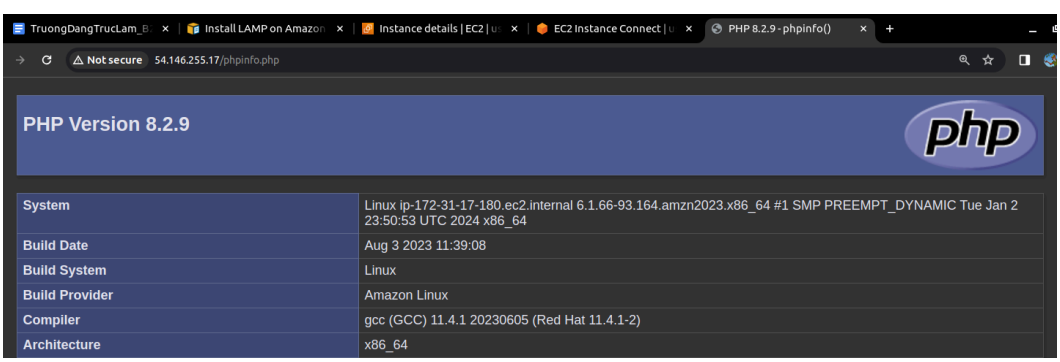
$ find /var/www -type f -exec sudo chmod 0664 {} \;

Now, **ec2-user** (and any future members of the **apache group**) can **add, delete, and edit files** in the **Apache document root**, enabling you to add content, such as a **static website** or a **PHP application**.

1.2.2. Test your LAMP Web Server

```
[ec2-user@ip-172-31-17-180 ~]$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
[ec2-user@ip-172-31-17-180 ~]$
```

Create a **PHP file** in the **Apache document root**:



Connect to the file you just created via:

http://<AWS instance public ID>/phpinfo.php

```
[ec2-user@ip-172-31-17-180 ~]$ echo "<?php phpinfo(); ?>" > /var/
[ec2-user@ip-172-31-17-180 ~]$ rm /var/www/html/phpinfo.php
[ec2-user@ip-172-31-17-180 ~]$
```

Remove the **.php** file via command: **$ rm /var/www/html/phpinfo.php**

1.3. Hosting a WordPress Blog with Amazon Linux

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hosting-wordpress.html

(Take screenshots to show that you finish every step on the tutorial)

1.3.1. Install WordPress

```
[ec2-user@ip-172-31-17-180 ~]$ wget https://wordpress.org/latest.tar.gz
--2024-01-15 07:02:32--  https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... conn
HTTP request sent, awaiting response... 200 OK
Length: 24479697 (23M) [application/octet-stream]
Saving to: 'latest.tar.gz'

latest.tar.gz                          100%[==========================

2024-01-15 07:02:33 (31.0 MB/s) - 'latest.tar.gz' saved [24479697/244796

[ec2-user@ip-172-31-17-180 ~]$
```

Download the latest WordPress installation package with the **wget** command:

$ wget https://wordpress.org/latest.tar.gz

```
[ec2-user@ip-172-31-17-180 ~]$ tar -xzf latest.tar.gz
[ec2-user@ip-172-31-17-180 ~]$
```

Unzip and **unarchive** the installation package. The installation folder is unzipped to a folder called **wordpress**:

$ tar -xzf latest.tar.gz

To create a database user and database for your WordPress installation:

```
[ec2-user@ip-172-31-17-180 ~]$ sudo systemctl start mariadb
[ec2-user@ip-172-31-17-180 ~]$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 10.5.20-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'suttocdo';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> CREATE DATABASE `wordpress-db`;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> exit
Bye
[ec2-user@ip-172-31-17-180 ~]$
```

Start the database server: **$ sudo systemctl start mariadb**
Log in to the database server as the root user: **$ mysql -u root -p**
Configure the database:

> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'suttocdo';

> CREATE DATABASE `wordpress-db`;

> GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";

> FLUSH PRIVILEGES;

To create and edit the wp-config.php file:

```
-]$ cp wordpress/wp-config-sample.php wordpress/wp-config.php
-]$
```

Copy the **wp-config-sample.php** file to a file called **wp-config.php**. This creates a new configuration file and keeps the original sample file intact as a backup.

```
[ec2-user@ip-172-31-17-180 ~]$ nano wordpress/wp-config.php
[ec2-user@ip-172-31-17-180 ~]$
```

Edit file **wp-config.php** with **nano**.

```
  GNU nano 5.8
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://wordpress.org/documentation/article/editing-w
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress-db' );

/** Database username */
define( 'DB_USER', 'wordpress-user' );

/** Database password */
define( 'DB_PASSWORD', 'suttocdo' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt.
define( 'DB_COLLATE', '' );

/**#@+
```

The contents of this file.

```
  GNU nano 5.8                                                    wordpress/wp-config.php
 *
 * Change these to different unique phrases! You can generate these using
 * the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}.
 *
 * You can change these at any point in time to invalidate all existing cookies.
 * This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         '9-}*@3>dJ ||k6B-SEP*>:k0W!9%l!v/n`>,hJ!2muBS);x%7lOU!VkzxS+ijQlX');
define('SECURE_AUTH_KEY',  'H&P[7{>eOP}gXvP+CN% zf(*m^sRtZQbmR%IHiqydQ.1g]rMX44SH|vztV~]7hE!');
define('LOGGED_IN_KEY',    'M0%WG&+LiTk?Lp;xli!<UJiTU.;raNNz!~+QNIM* Z9*y4Y*/vi* Bb?Js.L.pF+');
define('NONCE_KEY',        '$:DjzU+^h1iNVU+I;>v2&1|d=`@L,,X&U*YFMhAaoTP8ji-GXtKeT3sgGF:&7~}+');
define('AUTH_SALT',        ';G);an2m^k/x@Ea(CRE,`wrLjIdG=qo+e -N(*ycw|as)hS0-FR7S!@~FVsd 6NU');
define('SECURE_AUTH_SALT', 'ou8FVGb_)fmc^)f~z.tk~TvhAZb+m0L])x_p~_o5iv+xT!1Kbw7Pr2n3Tb0L68VG');
define('LOGGED_IN_SALT',   'U;L_AN|qb<|LD3#45Q+hx2W%F|uhTf]F4>jPLytt<62wb`ea(dN*?i @tO4*C`e2');
define('NONCE_SALT',       'UlXT79WOH:L`Bk^=/?Gty?b+-JgB+ C)5iZw+Z<(J;<{wnK%o s<R)LW&rJ7lOST');
/**#@-*/
```

Visit https://api.wordpress.org/secret-key/1.1/salt/ to randomly generate a set of **key values** that you can copy and paste into your **wp-config.php** file.

To install your WordPress files under the Apache document root

```
[ec2-user@ip-172-31-17-180 ~]$ cp -r wordpress/* /var/www/html/
[ec2-user@ip-172-31-17-180 ~]$
```

If you want **WordPress** to run at your document root, copy the contents of the wordpress installation directory (but not the directory itself) as follows:

$ cp -r wordpress/* /var/www/html/

To allow WordPress to use permalinks

$ sudo vim /etc/httpd/conf/httpd.conf

```
# Further relax access to the default document root:
<Directory "/var/www/html">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride All

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
</Directory>
```

Change the **AllowOverride None** to **AllowOverride All**.

To install the PHP graphics drawing library on Amazon Linux 2

```
[ec2-user@ip-172-31-17-180 ~]$ sudo yum install php-gd
Last metadata expiration check: 8:20:47 ago on Mon Jan 15 01:00:26 2024.
Dependencies resolved.
================================================================================
 Package                                            Architecture
================================================================================
```

Use the following command to install the PHP graphics drawing library on Amazon Linux 2: **$ sudo yum install php-gd**

To fix file permissions for the Apache web server

```
[ec2-user@ip-172-31-17-180 ~]$ sudo chown -R apache /var/www
[ec2-user@ip-172-31-17-180 ~]$ sudo chgrp -R apache /var/www
[ec2-user@ip-172-31-17-180 ~]$ sudo chmod 2775 /var/www
[ec2-user@ip-172-31-17-180 ~]$ find /var/www -type d -exec sudo chmod 2775 {} \;
[ec2-user@ip-172-31-17-180 ~]$  find /var/www -type f -exec sudo chmod 0644 {} \;
[ec2-user@ip-172-31-17-180 ~]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-17-180 ~]$
```

Run these commands:

$ sudo chown -R apache /var/www
$ sudo chgrp -R apache /var/www
$ sudo chmod 2775 /var/www
$ find /var/www -type d -exec sudo chmod 2775 {} \;
$ find /var/www -type f -exec sudo chmod 0644 {} \;
$ sudo systemctl restart httpd

Run the WordPress installation script with Amazon Linux 2

```
aws   Services   Search                                   [Alt+S]
[ec2-user@ip-172-31-17-180 ~]$ sudo systemctl enable httpd && sudo systemctl enable mariadb
Created symlink /etc/systemd/system/mysql.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/
[ec2-user@ip-172-31-17-180 ~]$
```

Use the **systemctl** command to ensure that the **httpd** and **database** services start at every system boot:

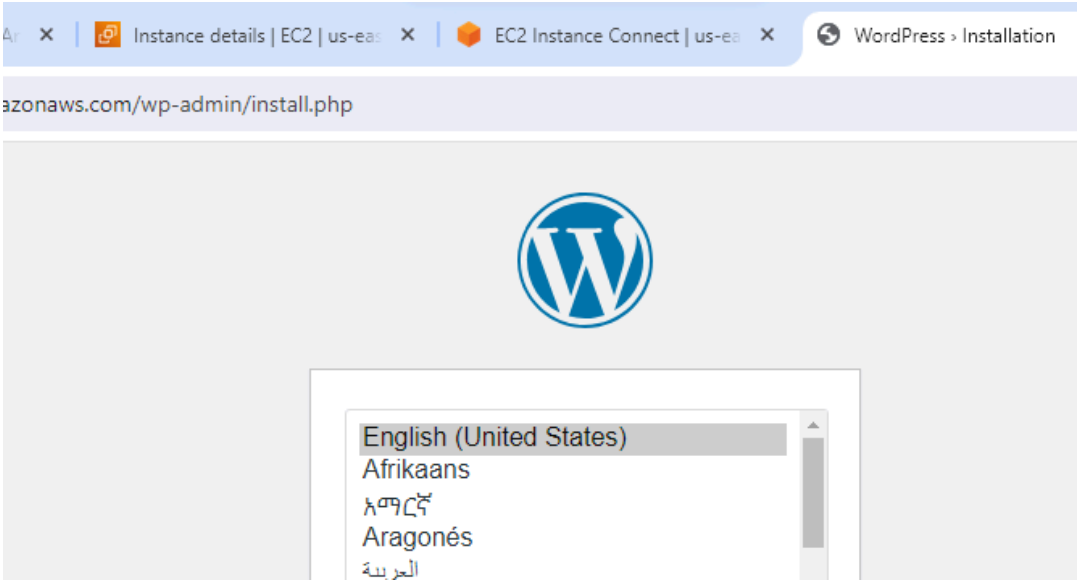
$ sudo systemctl enable httpd && sudo systemctl enable mariadb

```
[ec2-user@ip-172-31-17-180 ~]$ sudo yum install php-mysqlnd
Last metadata expiration check: 13:54:01 ago on Mon Jan 15 0
Dependencies resolved.
================================================================
 Package                                    Architecture
================================================================
Installing:
 php8.2-mysqlnd                             x86_64
```

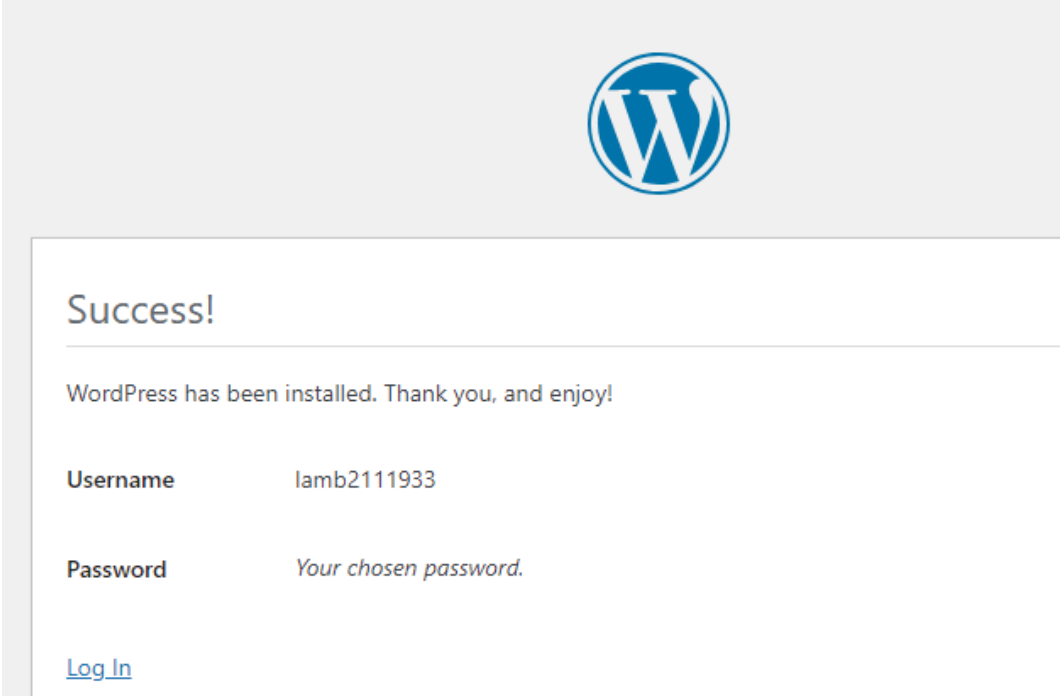
Install the **mysqlnd** extension. Use command:

$ sudo yum install php-mysqlnd

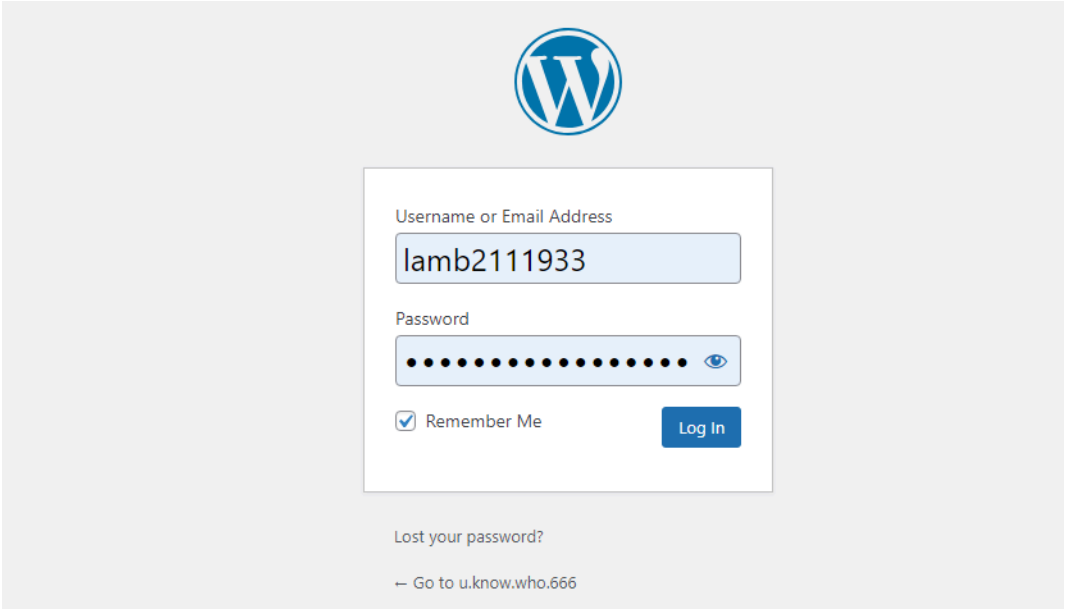
In a web browser, type the URL of your WordPress blog:



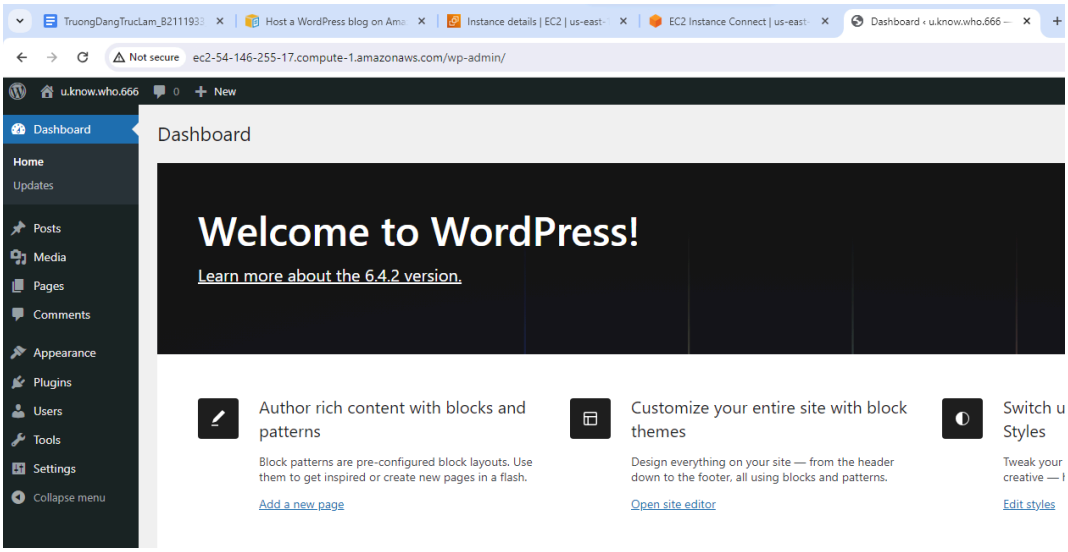
You should see the **WordPress** installation script



After installation



Login to **Wordpress**.



Enjoy your result!

2. Microsoft Azure

Register for an account to try Microsoft Azure (https://azure.microsoft.com/en-us/free/). Azure provides a $200 credit when you create an account. (a credit/debit card is required to activate your account).

2.1. Create a Linux virtual machine in the Azure portal
https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal
(Take screenshots to show that you finish every step on the tutorial)

2.2. Install a LAMP stack on an Azure Linux VM

https://usefulangle.com/post/326/azure-install-linux-apache-mysql-php-phpmyadmin-lamp-stack-ubuntu-18-04

(Take screenshots to show that you finish every step on the tutorial)

2.3. Install WordPress on an Azure Linux VM

https://docs.microsoft.com/en-us/azure/virtual-machines/linux/tutorial-lamp-stack

(Take screenshots to show that you finish every step on the tutorial)

3. Google Cloud

You are strongly recommended to try Google Cloud (https://cloud.google.com/) using your CTU student email account. A credit/debit card is required to activate your account, and then you will receive $300 to explore Google Cloud services

3.1. Creating and starting a VM instance

https://cloud.google.com/compute/docs/instances/create-start-instance

(Take screenshots to show that you finish every step on the tutorial)

3.2. Setting Up LAMP on Compute Engine

https://cloud.google.com/community/tutorials/setting-up-lamp

(Take screenshots to show that you finish every step on the tutorial)

3.3. Install WordPress on Google Cloud

https://themeisle.com/blog/install-wordpress-on-google-cloud/

(Take screenshots to show that you finish every step on the tutorial)

---END---