IT420 Project #2: Build a web app backed by a database w/ AMAZON AWS

Edgar Chicas

OS: Windows 10

Software Used: Putty and PuttyGEN

Website:

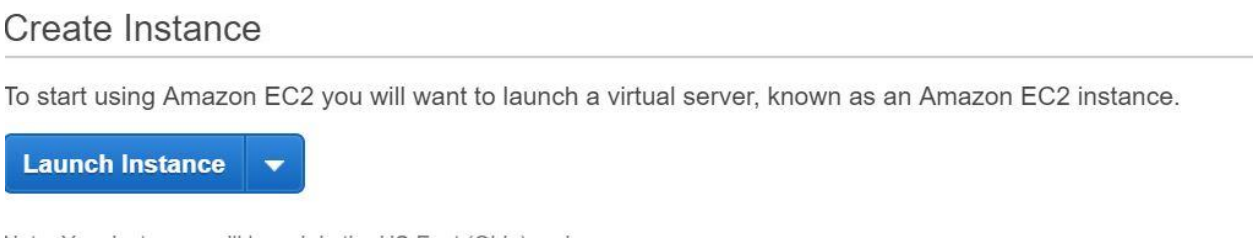
http://ec2-18-218-15-111.us-east-2.compute.amazonaws.com/

or

18.218.15.111

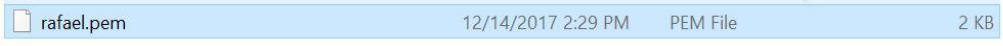
Step 1. AWS (Amazon Web Service)

To start the project we must create a student free tier account in AWS. Then select **EC2(Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud),** then we click on **Create Instance.**



From there choose Ubuntu Server 16.04, and select the second option which is (free tier).

Next on the final step for creating the instance in **Security Groups**, add a rule for HTTP and a rule for HTTPS (port 80 and 443), and change the SSH rule's source to **My IP**. After this, create a new key pair to download so you can SSH into your virtual machine, and download the(.pem file).

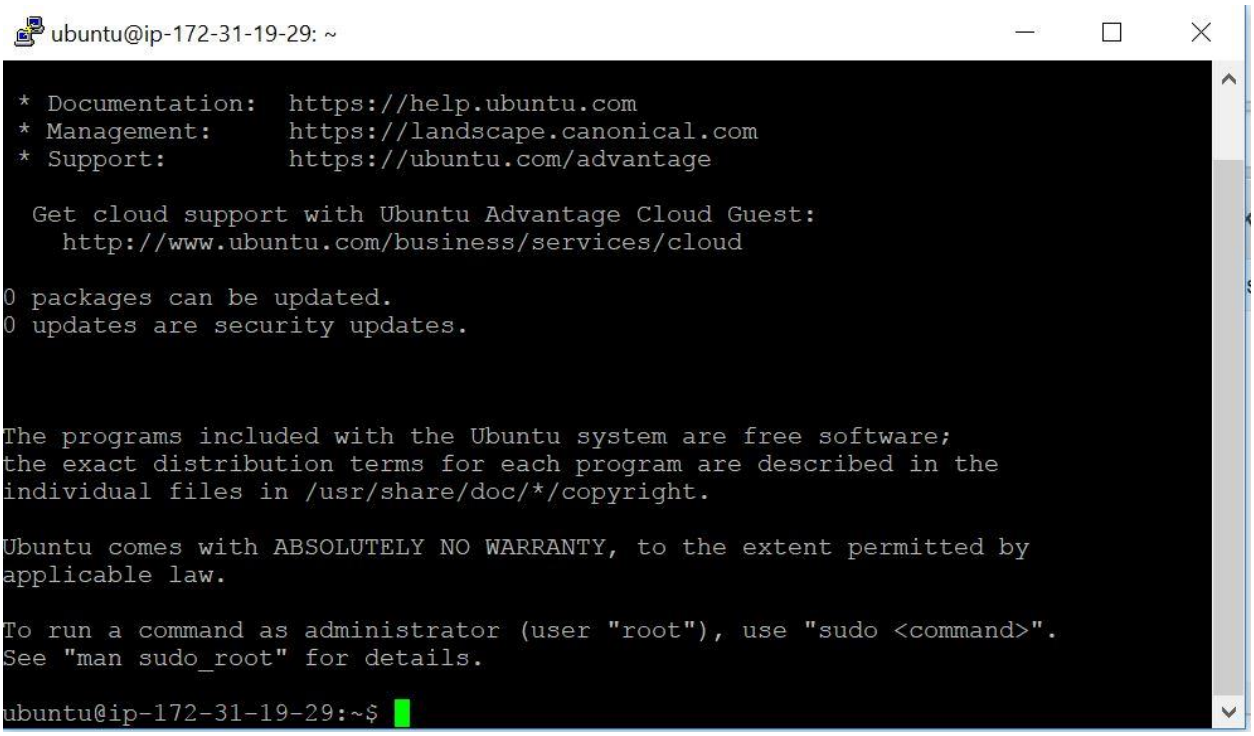


Next, open PuTTY. For the **Host Name field**, copy the **Public DNS (IPv4)** from your AWS dashboard, putting ubuntu@.

Example:

ubuntu@ec2-18-218-15-111.us-east-2.compute.amazonaws.com

The Putty **Port** is set to **22** and **Connection Type** is **SSH**. Next, on the left side panel, scroll done to **Connection** to then expand **SSH** and click on the **Auth** option. Click **Browse** and open the .ppk file that was saved from PuTTYgen. Go back to **Session** and click **Open**, and respond **Yes** to the Security Alert. Example:



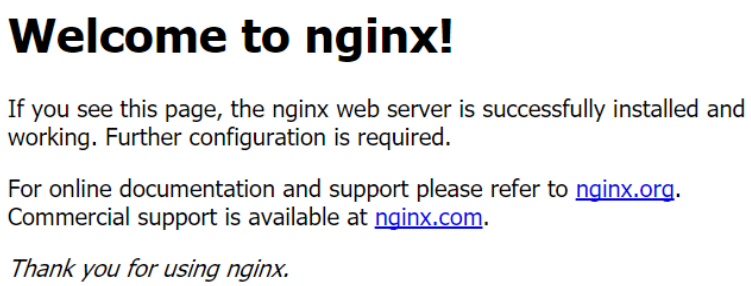
Step 2. Install LEMP

First install the Nginx , which is used to display the web page to users. To install the Nginx server, go to PuTTY and type into your console:

$ ***sudo apt-get update***

$ ***sudo apt-get install nginx***

To make sure your installation was successful, go to your browser and go to the IP address listed in your AWS dashboard under **IPv4 Public IP**. You should see this:



Next, we need to install MySQL to manage our site data.

$ ***sudo apt-get install mysql-server***

Create an administrator password used to access your database. Next, we need to secure our MySQL installation, which can be done by typing:

$ ***sudo mysql_secure_installation***

You will then be asked to enter the password you created in the previous step. Answer N when prompted to validate the password plugin and when prompted to change the root password, and Y to everything else until installation is complete.

Next, we need to install PHP for hypertext processing, which can be done by typing:

$ ***sudo apt-get install php-fpm php-mysql***

After successfully installing php, we need to configure the PHP processor to make it more secure:

$ ***sudo nano /etc/php/7.0/fpm/php.ini***

Find

cgi.fix_pathinfo=1 change it to **cgi.fix_pathinfo=0**

To apply the change you just made, restart PHP by typing:

$ ***sudo systemctl restart php7.0-fpm***

Next, we need to configure Nginx to use the PHP Processor we just installed. Edit the configuration file by typing:

$ ***sudo nano /etc/nginx/sites-available/default***

The configuration file looked like this:

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name 18.218.15.111;



        location / {
                try_files $uri $uri/ =404;
        }
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.0-fpm.sock;
        }
        location ~ /\.ht {
                deny all;
        }
}
```

Next, you need to check the configuration file for syntax errors. This can be done using:

$ ***sudo nginx –t***

Then reload Nginx to apply the changes you made:

$ ***sudo systemctl reload nginx***

Next, we need to create a .php file to make sure that Nginx is properly configured to use the PHP processor. Create a new file called *info.php* using:

$ ***sudo nano /var/www/html/info.php***

Then type the following text into the file and save it.

```
<?php
Phpinfo();
?>
```

Create a MySQL Database for WordPress

Log into the MySQL root account:

$ ***mysql –u root –p***

Create a new database by typing:

mysql> ***CREATE DATABASE wp DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;***

Next, you need to create a new MySQL user for the database you just created this can be done by typing:

mysql> ***GRANT ALL ON wp.* to 'it420'@'localhost' IDENTIFIED BY 'pass123';***

Lastly, flush the privileges exit:

mysql> ***FLUSH PRIVILEGES;***

mysql> ***EXIT;***

Step 3. Configuring Nginx for WordPress

In order for Nginx to properly handle WordPress, we need to go back to our Nginx *default* file and make some changes:

$ sudo nano /etc/nginx/sites-available/default

Make the following changes to your configuration, marked in red:

```
server {
listen 80 default_server;
listen [::]:80 default_server;

root /var/www/html;

index index.php index.html index.htm index.nginx-debian.html;

location = /favicon.ico {
                log_not_found off;
                access_log off;
        }
location = /robots.txt {
                log_not_found off; access_log off; allow all;
        }
location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
  expires max;
  log_not_found off;
}
```

```
server_name 52.14.41.49;

location / {
  #try_files $uri $uri/ =404;
  try_files $uri $uri/ /index.php$is_args$args;
}

location ~ \.php$ {
  include snippets/fastcgi-php.conf;
  fastcgi_pass unix:/run/php/php7.0-fpm.sock;
}

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

Save and close the file, then check for syntax errors:

$ ***sudo nginx -t***

If this responds successful, reload Nginx to apply the changes you made:

$ ***sudo systemctl reload nginx***

You also need to install some PHP extensions so that PHP can communicate with MySQL correctly. Download and install these extensions by typing:

$ ***sudo apt-get update***

$ ***sudo apt-get install php-curl php-gd php-mbstring php-mcrypt php-xml php-xmlrpc***

Once the extensions are done installing, restart PHP so that it can utilize the new extensions by typing:

$ ***sudo systemctl restart php7.0-fpm***

Step 4. Installing WordPress

The last thing we need to install is WordPress, start by changing your directory and downloading the Wordpress .tar.gz file:

$ ***cd /tmp***

$ ***curl -O https://wordpress.org/latest.tar.gz***

Then, extract the .tar.gz file into the current directory:

$ ***tar xzvf latest.tar.gz***

Copy the sample configuration file included in WordPress to the file that WordPress uses:

$ ***cp /tmp/wordpress/wp-config-sample.php /tmp/wordpress/wp-config.php***

Create an *upgrade* directory:

$ ***mkdir /tmp/wordpress/wp-content/upgrade***

And copy all of the contents of the WordPress directory to our website root directory:

$ ***sudo cp -a /tmp/wordpress/. /var/www/html***

And change the permissions of the contents:

$ ***sudo find /var/www/html –type d –exec chmod g+s {} \;***

$ ***sudo chmod g+w /var/www/html/wp-content***

$ ***sudo chmod -R g+w /var/www/html/wp-content/themes***

$ ***sudo chmod -R g+w /var/www/html/wp-content/plugins***

Next, we need to set up the WordPress configuration file. In order to do this, we need to obtain some values from WordPress's key generator, which can be done by typing:

$ ***curl -s https://api.wordpress.org/secret-key/1.1/salt/***

Copy the output of this command. Then open the WordPress configuration file:

$ ***nano /var/www/html/wp-config.php***

Now you need to change some of the preset values to correspond with the ones you created before. Make the changes marked in red, replacing the appropriate fields with your own information:

```
define('DB_NAME', 'wp');
define('DB_USER', 'it420');
define('DB_PASSWORD', 'pass123');
define('FS_METHOD', 'direct');
define('AUTH_KEY',         '[copied values]');
define('SECURE_AUTH_KEY',  '[copied values]');
define('LOGGED_IN_KEY',    '[copied values]');
define('NONCE_KEY',        '[copied values]');
define('AUTH_SALT',        '[copied values]');
define('SECURE_AUTH_SALT', '[copied values]');
define('LOGGED_IN_SALT',   '[copied values]');
define('NONCE_SALT',       '[copied values]');
```

Save and exit, then return to your website.

Installing Siege and Testing

Siege is a program which is used to test the load capacity of websites. You can choose how many "users" you want to access the website and for how long. Start by installing Siege:

$ ***sudo apt-get install siege***

Start by configuring Siege for 25 concurrent users over 1 minute. Open the .siegerc file by typing:

$ ***sudo nano ~/.siegerc***

Then change the following fields:

concurrent = 25

time = 1M

show-logfile = false

Save and close the file, then issue the following command to begin the siege:

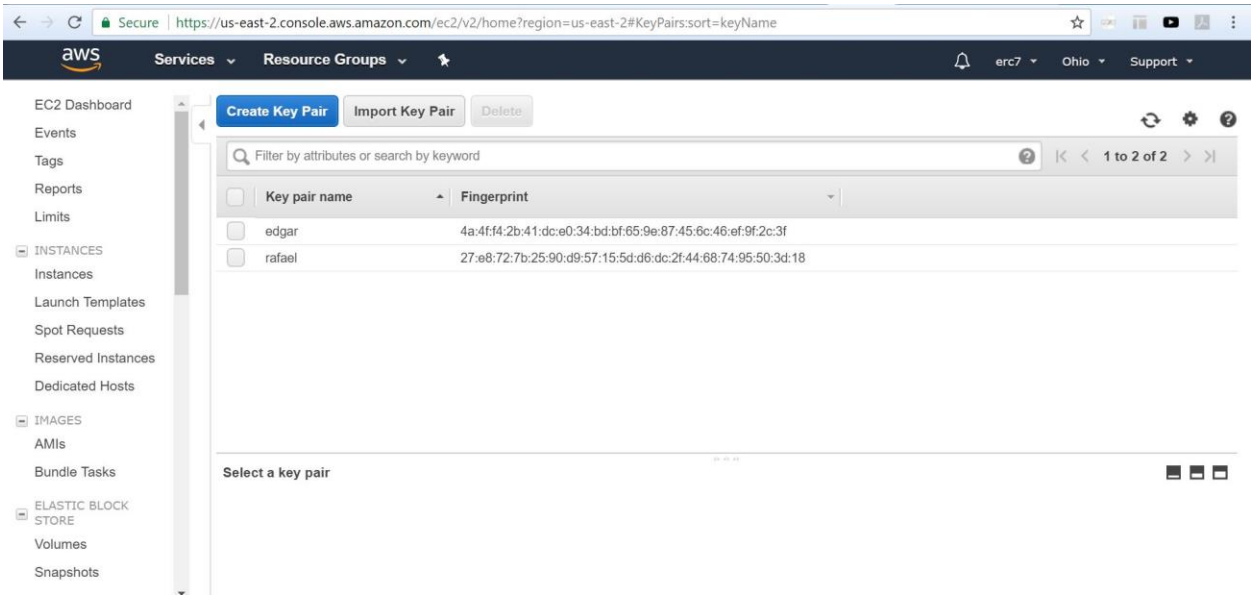
$ siege http://ec2-18-218-15-111.us-east-2.compute.am azonaws.com/ siege http://ec2-18-218-15-111.us-east-2.compute.amazonaws.com/

Test results for my website:

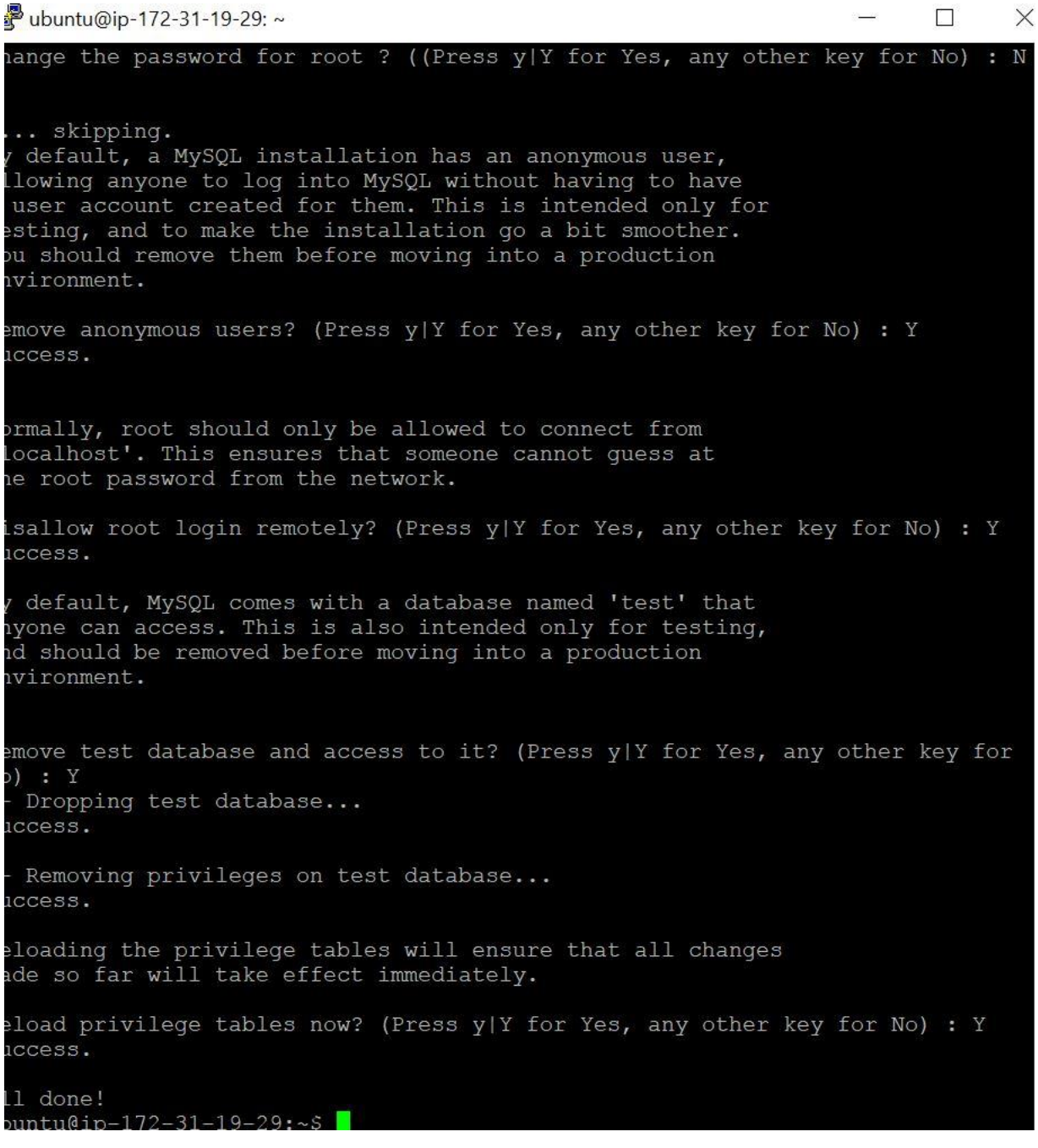
Concurrent Users	Response Time
25	0.10 secs
50	0.42 secs
100	1.28 secs
150	2.11 secs
AVERAGE RESPONSE TIME:	0.9775 secs

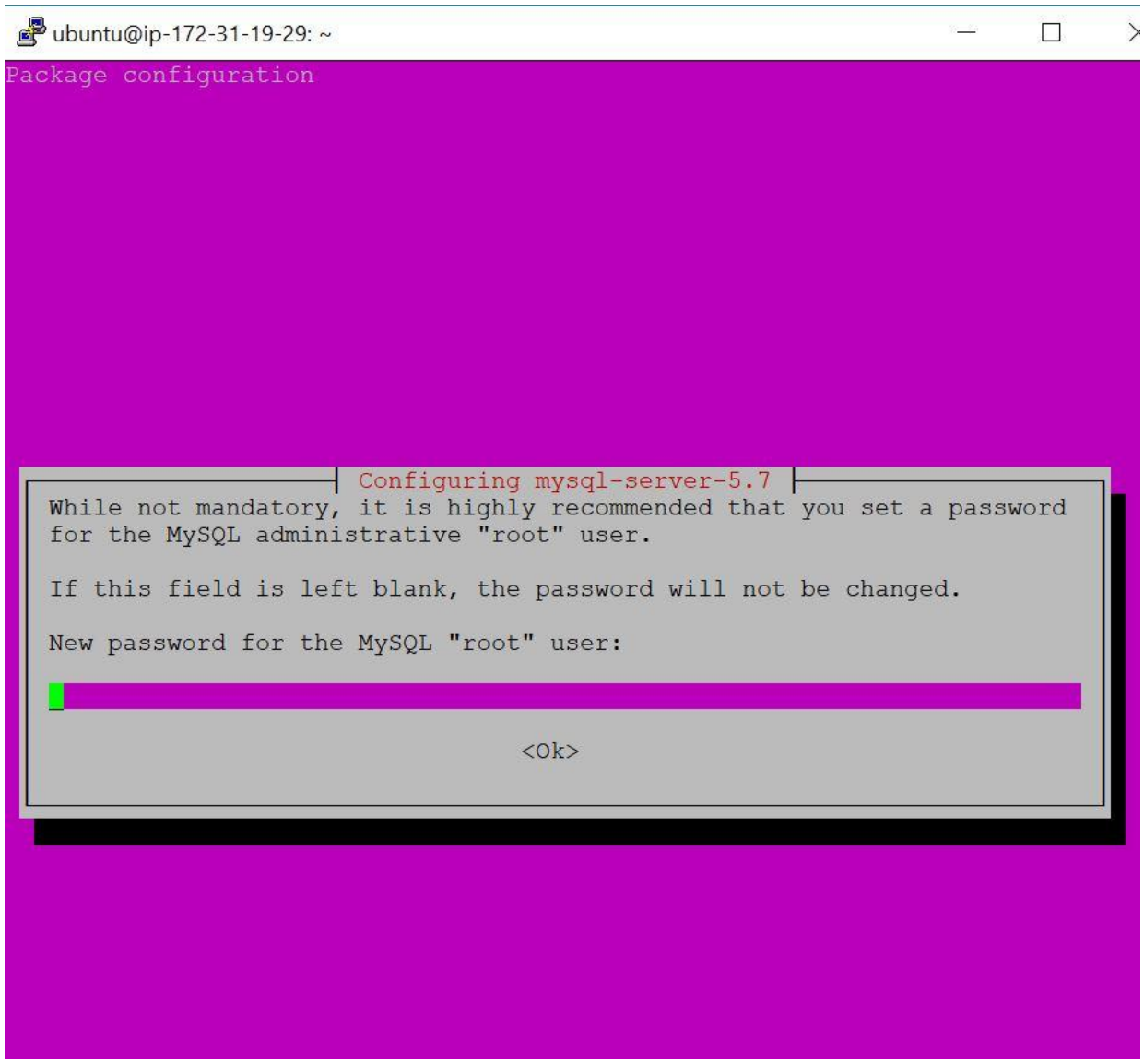
Based on the test results, the free tier configuration in AWS was not meant to handle many concurrent users. I believe for this I need to get the payed AWS to get better response + a better configuration. Siege's practical applications include finding the maximum number of concurrent users your website can handle, gauging the response time to see the speed of your website, and finding the maximum number of transactions it can handle.

My AWS Dashboard



MYSQL


ubuntu@ip-172-31-19-29: ~
hange the password for root ? ((Press y|Y for Yes, any other key for No) : N
... skipping.
y default, a MySQL installation has an anonymous user,
llowing anyone to log into MySQL without having to have
user account created for them. This is intended only for
esting, and to make the installation go a bit smoother.
ou should remove them before moving into a production
nvironment.
emove anonymous users? (Press y|Y for Yes, any other key for No) : Y
uccess.
ormally, root should only be allowed to connect from
localhost'. This ensures that someone cannot guess at
he root password from the network.
isallow root login remotely? (Press y|Y for Yes, any other key for No) : Y
uccess.
y default, MySQL comes with a database named 'test' that
nyone can access. This is also intended only for testing,
nd should be removed before moving into a production
nvironment.
emove test database and access to it? (Press y|Y for Yes, any other key for
) : Y
- Dropping test database...
uccess.
- Removing privileges on test database...
uccess.
eloading the privilege tables will ensure that all changes
ade so far will take effect immediately.
eload privilege tables now? (Press y|Y for Yes, any other key for No) : Y
uccess.
ll done!
buntu@ip-172-31-19-29:~$

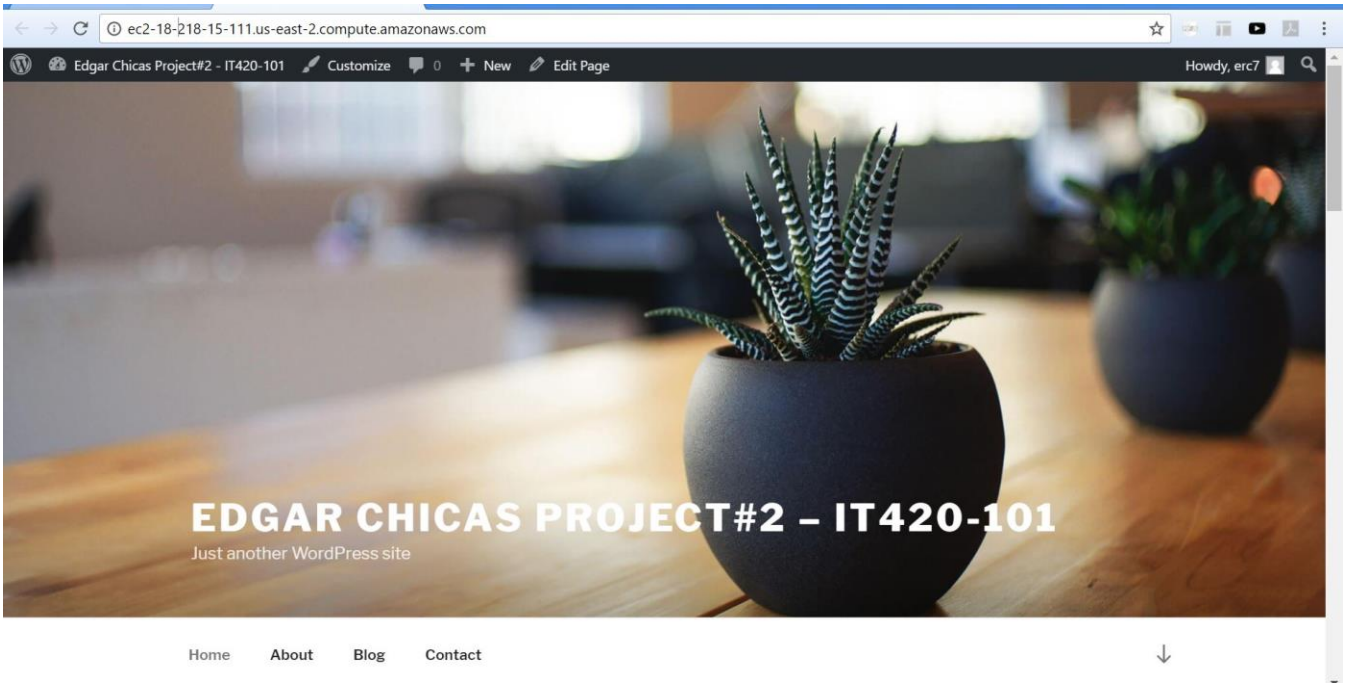

ubuntu@ip-172-31-19-29: ~
Package configuration
Configuring mysql-server-5.7
While not mandatory, it is highly recommended that you set a password
for the MySQL administrative "root" user.
If this field is left blank, the password will not be changed.
New password for the MySQL "root" user:
<Ok>

Siege

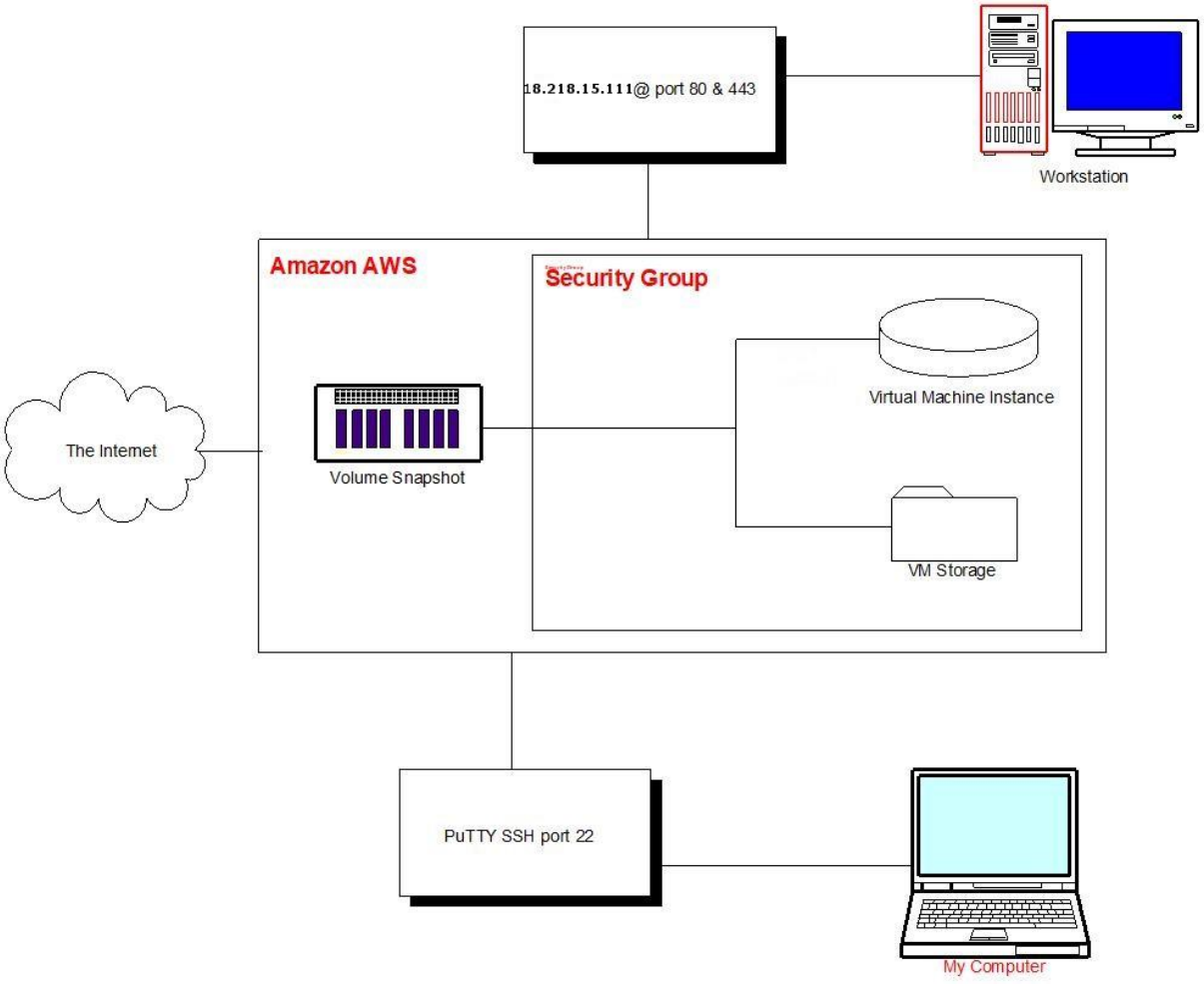
```
ubuntu@ip-172-31-19-29:/tmp$ siege http://ec2-18-218-15-111.us-east-2.compute
.amazonaws.com/
** SIEGE 3.0.8
** Preparing 25 concurrent users for battle.
The server is now under siege...
The server is now under siege...
Lifting the server siege..      done.

Transactions:                   2715 hits
Availability:                 100.00 %
Elapsed time:                  59.89 secs
Data transferred:             156.21 MB
Response time:                  0.55 secs
Transaction rate:              45.33 trans/sec
Throughput:                     2.61 MB/sec
Concurrency:                   24.89
Successful transactions:        2715
Failed transactions:               0
Longest transaction:            0.64
Shortest transaction:           0.07
```

My Website(Finalized)



Network Diagram



GITHUB:

https://github.com/Raff2015/IT420

Reference

https://aws.amazon.com/getting-started/tutorials/launch-an-app/

https://aws.amazon.com/getting-started/tutorials/

https://www.youtube.com/watch?v=fPZuN_fibjM&t=207s