

CNG 495

Fall – 2022

Term Project Progress Report
I

Ibrahim Ozkan 2456275

Adil B. Kebapcioglu 2455954

1. Milestones Achieved

1.1. Week 1 - October 31 - November 6

After getting approval for the project, I started making research on the cloud technologies that I will use to deploy our project. I made a research on setting up an instance for AWS EC2 server with desired Linux distro which is Ubuntu 20.04. Also after making research on setting up an instance, I had to search for Nginx installation to Ubuntu with PHP 7.4 version since we are planning to host our RestAPI in the EC2 instance. Lastly, Composer needs to be installed in order to run Laravel on the machine. According to my research I listed each installation steps for AWS EC2 cloud utilization below:

Creating an EC2 Instance

1. First you need to create an account on AWS with your credentials. You also need to provide your credit card details before registering. AWS will not charge anything to your credit card, it will just only verify your credit card only by charging 1\$ (it will be given back to you after registration). [You can visit registration page by clicking this text.](#) The Figure 1 demonstrates initial registration screen of the AWS.

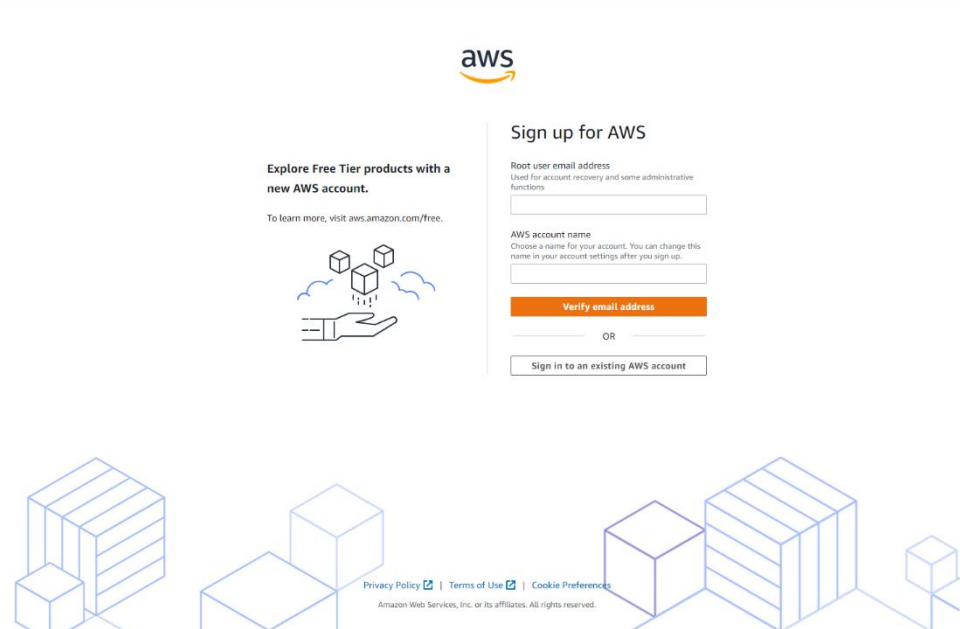


Figure 1 AWS Registration Screen

- After registration, your account needs to be approved by the AWS which can take up to 24 hours. Mine took only 10 minutes.
- When approval to your account is done you will be directed to your console page. Here type “EC2” to the search box on the page and click on “EC2” which is also shown in the Figure 2 and you will be directed to Dashboard of EC2 management.

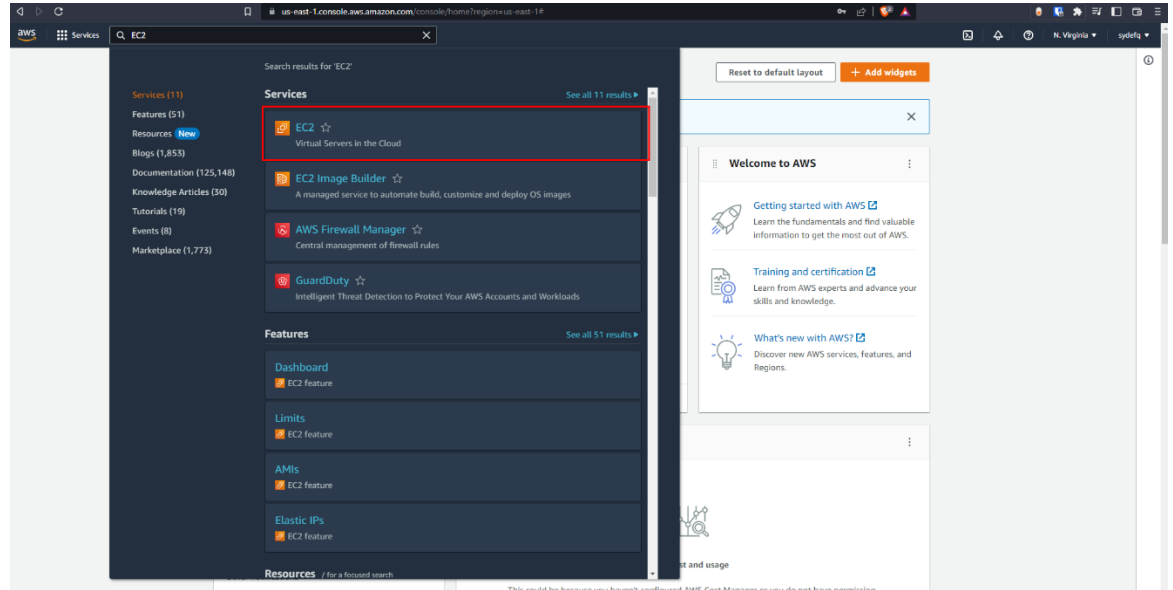


Figure 2 AWS Console Page

- Here inside Dashboard page, to create a new instance click on Launch Instance button as shown in the Figure 3.

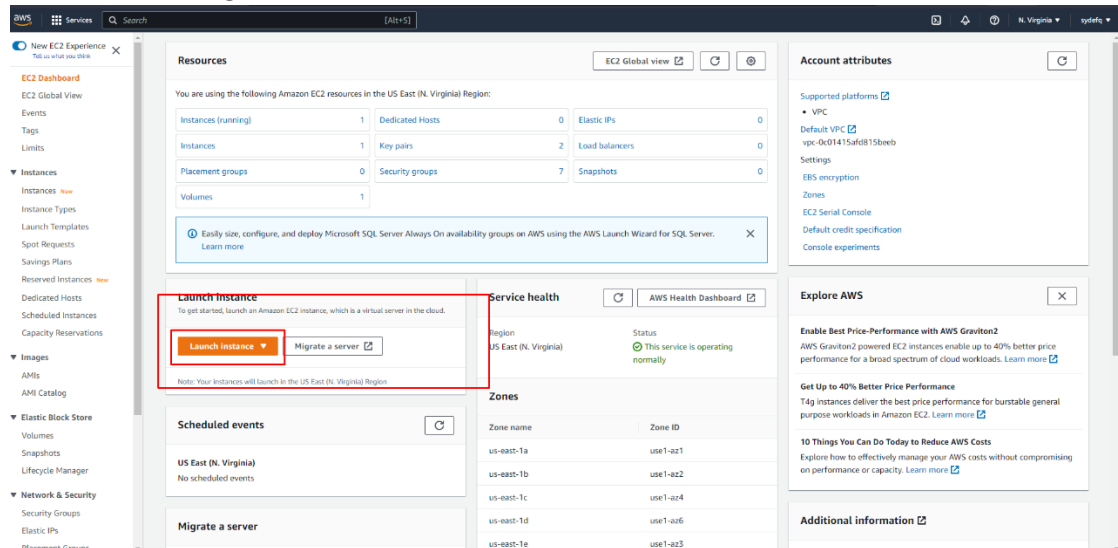


Figure 3 EC2 Dashboard

- In this page you need to fill all necessary fields to create an instance.
 - “Name and tags” field needs to be filled which will be the identifier of your server. Lets name it as “CNG 491”

- “Application and OS Images” part is to decide which OS will run on your system. We will select Ubuntu for desired OS and select “Ubuntu Server 20.04” as distro version. Architecture will be selected as “64-bit (x86)”
- For instance type we will be selecting any of the free-tiers to use free version of the EC2. If you want greater resources for your EC2, you can select paid instance types. Mine is “t2.micro” with 1gb memory, 1 vCPU.
- Key pair field is necessary for SSH connection you can create your key pair and name it whatever you wish. Lets name it as “CNG491”.
- Network settings are necessary for setting up some rules for incoming and outgoing traffic for EC2. To host a web server, select Allow SSH traffic, Allow HTTPS traffic and HTTP traffic.
- Finally, we need to configure our storage. AWS provides free storage up to 30gb. Lets use them all and configure our storage as 30gb.

After you fill all fields page should look like Figure 4, 5 and 6.

The screenshot displays the AWS Management Console's 'Launch an instance' page. The breadcrumb navigation at the top reads 'EC2 > Instances > Launch an instance'. The main heading is 'Launch an instance' with a sub-note: 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.' The 'Name and tags' section has a text input for 'Name' containing 'CNG 491' and a button for 'Add additional tags'. The 'Application and OS Images (Amazon Machine Image)' section includes a search bar and a 'Quick Start' tab. Under 'Quick Start', a grid of AMIs is shown, with 'Ubuntu' selected. Below the grid, the details for the selected AMI are shown: 'Ubuntu Server 20.04 LTS (HVM), 64-bit (x86_64)', 'ami-0145b06a6cc0c0b00', 't2.micro', and 'Free tier eligible'. The 'Summary' section on the right provides a overview of the configuration: 'Number of instances: 1', 'Software Image (AMI): Canonical, Ubuntu, 20.04 LTS', 'Virtual server type (instance type): t2.micro', 'Firewall (security group): New security group', and 'Storage (volumes): 1 volume(s) - 30 GiB'. A 'Free tier' notification box is also present, stating: 'Free tier in your first year includes 750 hours of t2.micro (or t2.nano) in the Regions in which t2.micro is unavailable. Instance usage on free tier AMIs per month: 30 GiB of EBS storage, 2 million IOPS, 1 GB of snapshots, and 100 GB of bandwidth to the internet.' At the bottom right, there are 'Cancel' and 'Launch instance' buttons.

Figure 4 Instance Launch Page 1

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory

On-Demand Linux pricing: 0.0116 USD per Hour

On-Demand Windows pricing: 0.0162 USD per Hour

Free tier eligible

Compare instance types

Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

cng491

Create new key pair

Network settings

Network

vpc-0c01415afd815beeb

Subnet

No preference (Default subnet in any availability zone)

Auto-assign public IP

Enable

Firewall (security groups)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-6' with the following rules:

Summary

Number of instances

1

Software Image (AMI)

Canonical, Ubuntu, 20.04 LTS, ...read more

ami-0149b2da6ccec4bb0

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 30 GiB

Free tier

In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel

Launch Instance

Figure 5 Instance Launch Page 2

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-6' with the following rules:

Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0

Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage

Advanced

1x

30

GiB

gp2

Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems

Edit

Advanced details

Info

Summary

Number of instances

1

Software Image (AMI)

Canonical, Ubuntu, 20.04 LTS, ...read more

ami-0149b2da6ccec4bb0

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 30 GiB

Free tier

In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel

Launch Instance

Figure 6 Instance Launch Page 3

- Click on Launch instance to create a new instance.

1.2. Week 2 - November 7 - November 13

After making research on week 1, I followed the steps that I provided for creating a EC2 instance. After the creation, I connected to terminal of the EC2 instance to setup a Nginx server with PHP and Composer.

To connect terminal of EC2 Instance follow these steps:

1. When you are in your EC2 Dashboard, click on “Running Instances” which will direct you to a page where all your running instances are listed as in Figure 7

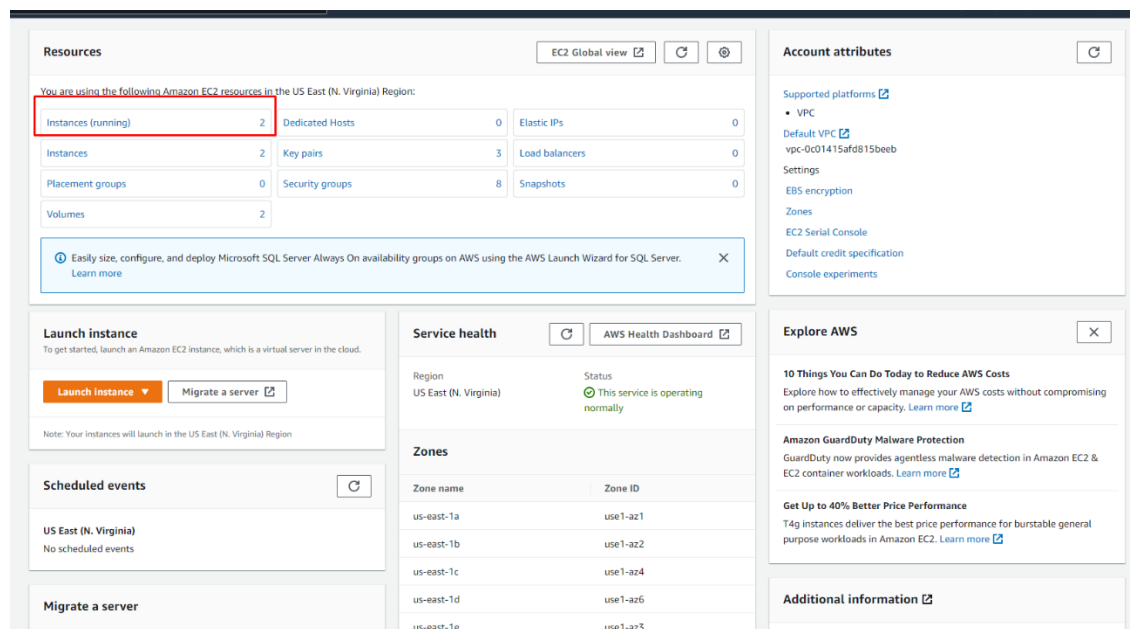


Figure 7 EC2 Dashboard

2. Then, mark the instance that you want to connect and click on the “Connect” button to be directed to the connecting to EC2 page like in Figure 8.

Instances (1/2) [Info](#)

Find instance by attribute or tag (case-sensitive)

Connect

Instance state

Actions

Launch Instances

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input checked="" type="checkbox"/>	491 Laravel Se...	i-0ae663b8c67e4420e	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-3-86-218-135.com...	3.86.218.135	-
<input type="checkbox"/>	CNG 491	i-0f06e060c7c2bfd6	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-18-233-0-30.comp...	18.233.0.30	-

Instance: i-0ae663b8c67e4420e (491 Laravel Server)

Details

Security

Networking

Storage

Status checks

Monitoring

Tags

▼ Instance summary [Info](#)

Instance ID

i-0ae663b8c67e4420e (491 Laravel Server)

IPv6 address

-

Hostname type

IP name: ip-172-31-90-104.ec2.internal

Answer private resource DNS name

IPv4 (A)

Auto-assigned IP address

3.86.218.135 [Public IP]

Public IPv4 address

3.86.218.135 | [open address](#)

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-90-104.ec2.internal

Instance type

t2.micro

VPC ID

vpc-0c01415afd815beeb

Private IPv4 addresses

172.31.90.104

Public IPv4 DNS

ec2-3-86-218-135.compute-1.amazonaws.com | [open address](#)

Elastic IP addresses

-

AWS Compute Optimizer finding

[Opt-in to AWS Compute Optimizer for recommendations.](#) | [Learn more](#)

Figure 8 EC2 Instances Page

3. After that, select “EC2 Instance Connect” and press connect to connect to the terminal of EC2.

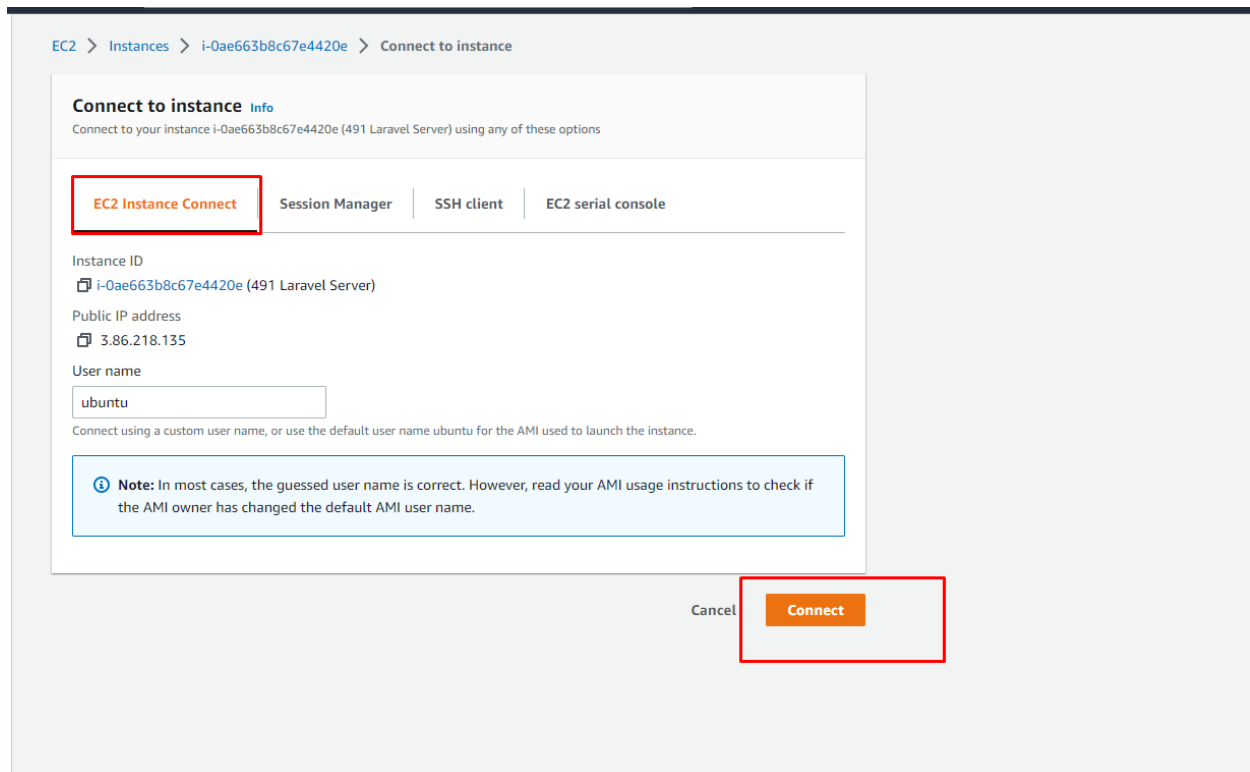


Figure 9 EC2 Instance Connect Page

For installation of Nginx, PHP and Composer, I followed the tutorial provided by the Rahul Gupta which [you can visit the tutorial web page by clicking here](#) (Gupta, 2020). I had difficulties in installation of Composer since it required to have different PHP versions for each Laravel project. For instance, if Laravel project is built with PHP version 8, Composer will give an error for any PHP version under 8 which ours was 7.4. We solved this issue by changing the “composer.json” file inside Laravel project to decrease PHP version requirement.

1.3. Week 3 - November 14 - November 20

Last week of our sprint was about deployment of the project to the AWS via Github. Also, I developed a simple demo application which involved showing current population of a in-door area with increment and decrementing functions of the population as well which more detailed explanation of mobile application will be done in later paragraphs.

Cloning our project to the Github was quite a challenge since Github disabled authentication with username and password for SSH connections. Therefore we needed to create an SSH RSA key for Github inside our server and configure the authentication.

To clone our private Github repository and SSH authentication with RSA key, I followed these steps:

1. First go to directory of “/.ssh” by executing the following command.

```
cd ~/.ssh
```

2. Then execute the following command with providing your Github email to create an SSH key. Press enter for all asked fields.

```
ssh-keygen -t ed25519 -C "your github email"
```

3. After creating your key you need to execute the following command to get your SSH key which will be used in your Github profile settings.

```
cat id_rsa.pub
```

```
root@ip-172-31-90-104:/home/ubuntu/.ssh# ssh-keygen -t ed25519 -C "ibrahim.ozkan@mtu.edu.tr"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:chV1lN01b6dpy3e29+965pb2MnV1+MzEonP3aLxrc ibrahim.ozkan@mtu.edu.tr
The key's randomart image is:
+--[ED25519 256]--+
|
|..o..o..o|
|..o..o..o|
|o..o..o..|
|S..o..o..|
|o..o..o..|
|+o..o..|
|...+..|
|+E+|
+----[SHA256]-----+
root@ip-172-31-90-104:/home/ubuntu/.ssh# ls
authorized_keys
root@ip-172-31-90-104:/home/ubuntu/.ssh# cat id_rsa.pub
cat: id_rsa.pub: No such file or directory
root@ip-172-31-90-104:/home/ubuntu/.ssh# cd ~/.ssh
root@ip-172-31-90-104:~/.ssh# ls
authorized_keys id_ed25519 id_ed25519.pub id_rsa id_rsa.pub known_hosts
root@ip-172-31-90-104:~/.ssh# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQBAQC0ezxp/CSY560Tckl0llw-1Pj6d2jg4V9jw6gC0bg3oB+dp9Ckryrhp62+oTCS0Mk0rhshPLy4IVjAaG6fKJ1B7hzRG+Y1KSJPch0IS1a7pT0QE83ZHLfM4Zto95vA5zJTe/fYt0vH3JuaEx/B+PHUIdkP9-1nk5mB6AdrcXAB+Ce2v1l0F/hsa1dH1gTS
e44M6d0rc/23ICtzt2p3pY0d0h3T0d0A13w gltr5U11Nn4N10e7Z/G/d0Kcalj391Cz0QmTzKJ1d0Vp3d0ZmY21d0ampdrtu7hsq0bz3Cyw1/1sQd0amp/f0b5P+>+7ZPc1v07duHdc415dMal13r05VJm05+Y7J9w0v0B0h8/ndpFlux1Nay4so1a3J0LyfN0nyT0rT05CvTc7yngjflB004/jnp
5ohf8s1R10kySP6F9w0d0p9X2Hfats/03CY2K0B1fM0oma0UfBh51MvyJ0n0LxjwQ2gA8= ad1lbozkart2002@hotmail.com
root@ip-172-31-90-104:~/.ssh#
```

Figure 10 Ubuntu Terminal with Steps for Github

- Copy the following key and open your Github profile settings, click on “SSH and GPG Keys”.

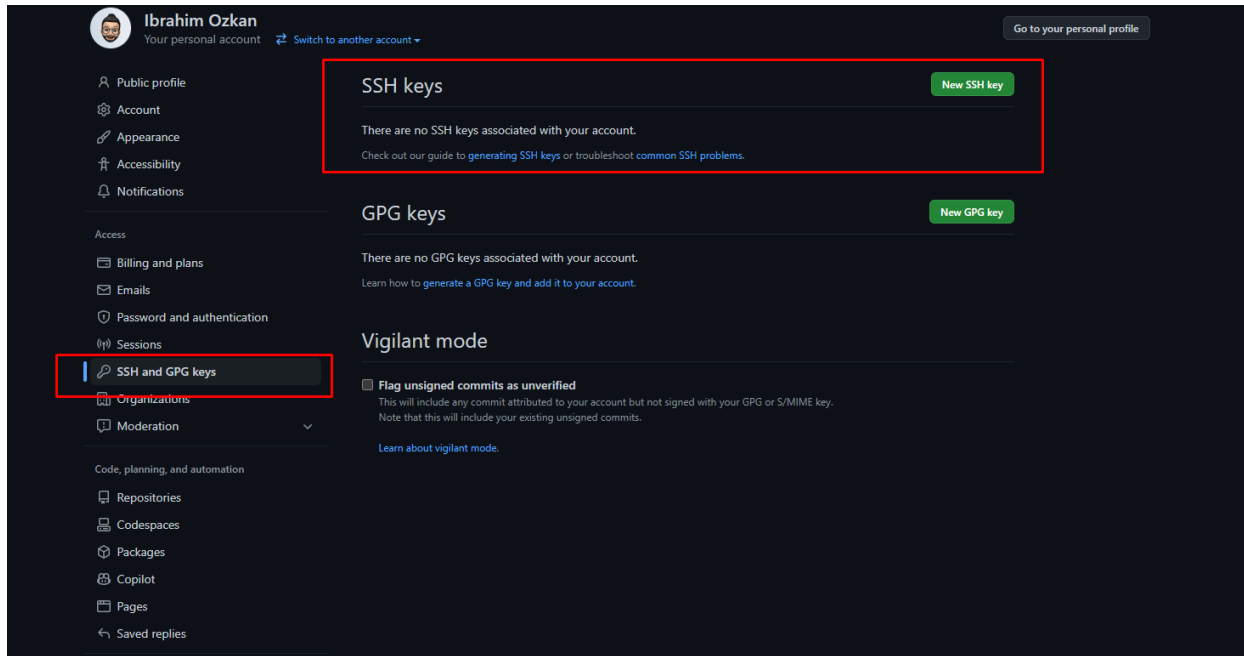


Figure 11 Github Profile Settings

- Click on the “New SSH Key” and paste your SSH key like in Figure 12

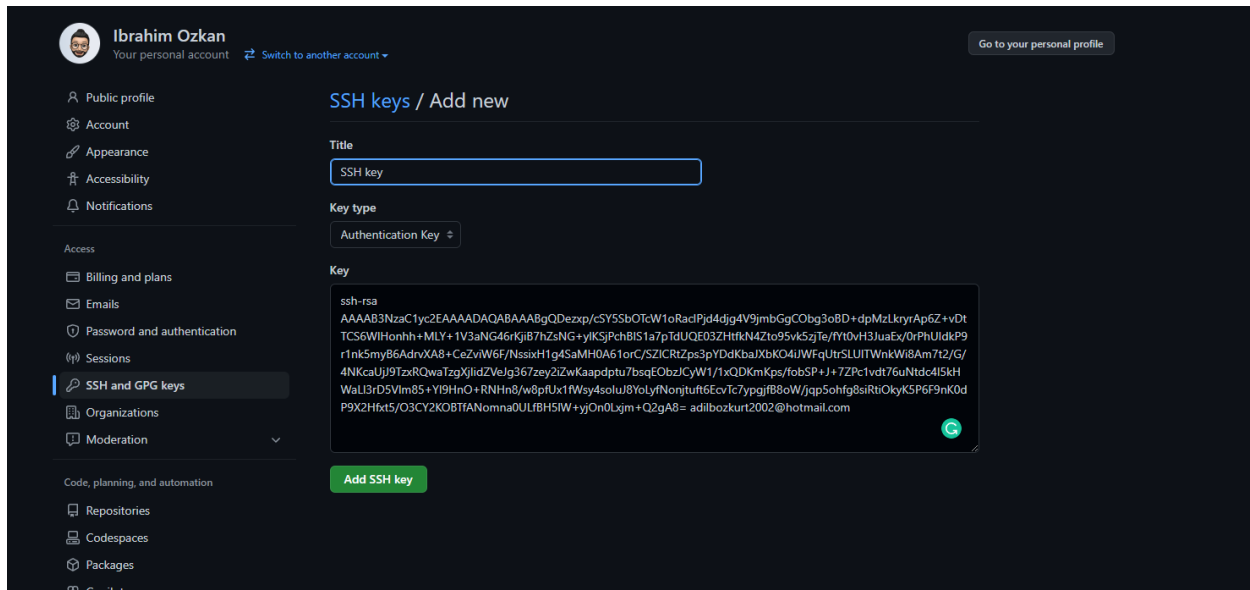


Figure 12 Github Create SSH Key Page

6. You successfully authenticated your Github account with your EC2 instance. Now to clone our repository go to directory by the following command:

```
cd /var/www/html
```

7. Here, execute the following command to clone your repository successfully.

```
git clone git@github.com:ibrahimozkn/CNG495-F22-CloudComputing.git
```

After cloning our repo, which can be visited by clicking on this text, I developed a mobile application for demonstration. Application is just a single screen with few buttons which can be seen in Figure 13.

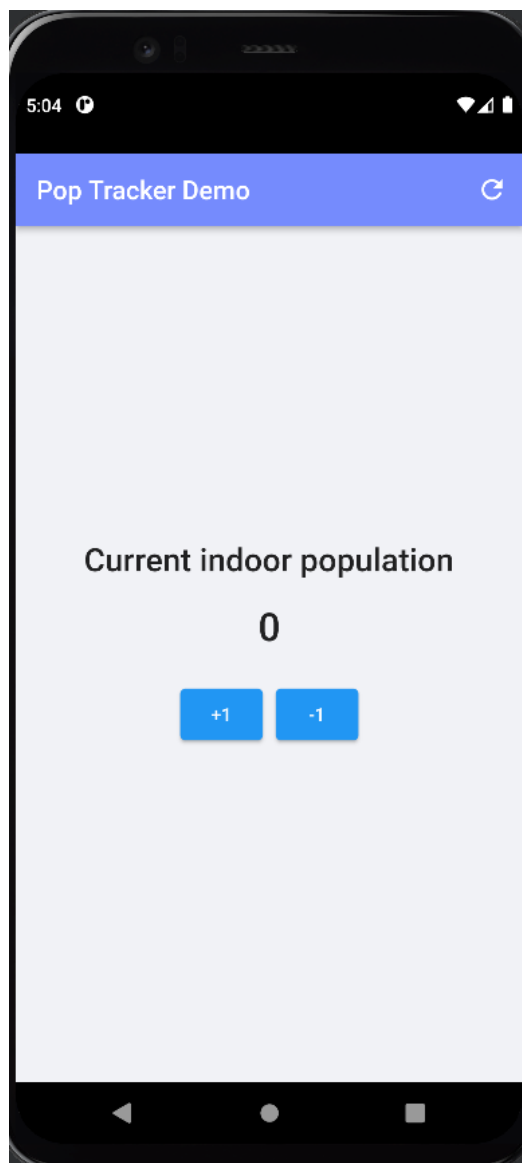


Figure 13 Mobile Application Main Page

Here, current population inside a business premisses is shown in numbers. Since we did not still implemented a Arduino UNO to carry out entrance and departure, I added two buttons to increment and decrement the population via API calls. For API calls, I created a class called DioClient which is using the Dio package of Flutter. You can visit the Github page to check the implemented version of DioClient class.

[DioClient class Github Page](#)

Cloning URL of our repository:

<https://github.com/ibrahimozkn/CNG495-F22-CloudComputing.git>

For API calls there are 3 calls that I use which are:

- Get the population info
- Increment the population
- Decrement the population

When these calls are made, returned JSON data will be used to update the population information. At each increment or decrement API call, Population information call is done to get the up-to-date population count.

2. Milestones Remained

Milest one No	Week	Description	Responsible Student(s)
1	November 28 - December 4	Arduino UNO Configuration & Installation	Ibrahim Ozkan, Adil B. Kebapcioglu
2	December 5 - December 11	Fully implementation of API Database	Adil B. Kebapcioglu
3	December 12 - December 18	Fully implementation of API	Adil B. Kebapcioglu
4	December 19 - December 25	Design UI of the mobile application for all functionalities	Ibrahim Ozkan
5	December 26 - January 1	Setup Google Maps API for map view in mobile application	Ibrahim Ozkan
6	January 2 - January 8	Make mobile application fully functioning	Ibrahim Ozkan

7	January 9 - January 15	Connect all components (API, Arduino, Mobile Applications) together and make system ready to use	Ibrahim Ozkan, Adil B. Kebapcioglu
8	January 16 - January 22		

2.1. Delivery List

Materials that will be delivered when project finishes are listed below:

- Code for API (Laravel)
- Code for Mobile Application (Flutter)
- Instructions for setting up EC2 server, Google API and Laravel Deployment
- APK file for Mobile Application
- Database file of demo
- Arduino code
- Arduino UNO setup instructions