

DEPLOY A FLASK WEB APPLICATION IN AWS USING EC2

A project report submitted

In partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING

by

Onteru Nagaraju	(18131A05C6)
Palli Vamsi Krishna	(18131A05D3)
P. Siddharth	(18131A05D4)
S.Tejeswar Reddy	(18131A05G2)

Under the esteemed guidance of

Dr. CH.Sita Kumari
Asst.Professor,
GVPCOE(A).



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GVP COLLEGE OF ENGINEERING (A)

Approved by AICTE, New Delhi and Affiliated to JNTUKakinada
Re-accredited by NAAC with "A" Grade with a CGPA of 3.47/4.00
VISAKHAPATNAM – 530048



COLLEGE OF ENGINEERING
(AUTONOMOUS)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GVP COLLEGE OF ENGINEERING (A)**

Approved by AICTE, New Delhi and Affiliated to JNTUKakinada
Re-accredited by NAAC with "A" Grade with a CGPA of 3.47/4.00
VISA KHAPATNAM – 530048

CERTIFICATE

This is to certify that the project work entitled "**Deploy A Flask Web Application In AWS Using EC2**" being submitted by **Onteru Nagaraju (18131A05C6), P. Vamsi Krishna (18131A05D3), P. Siddharth (18131A05D4), S. Tejeswar Reddy (18131A05G2)** in partial fulfillment of the requirement the award of the degree of "Bachelor of technology" in Computer Science and Engineering is a record of bonafide work done by them under my supervision during the academic year 2020-2021.

Guide

Dr. CH. Sitha Kumari,
Assistant Professor
Department of CSE
GVPCOE(A)

Head Of The Department

Dr. D N D Harini,
Head of Department
Department of CSE
GVPCOE(A)

DECLARATION

We hereby declare that the project report entitled "**Deploy A Flask Web Application In AWS Using EC2**" submitted by us to **Gayatri Vidya Parishad College of Engineering(Autonomous)** in partial fulfilment of the requirement for the award of the degree of B.Tech (Computer Science and Engineering) is a record of bonafide major project work carried out by us under the guidance of **Dr. Ch.Sitha Kumari**. I further declare that the work reported in this main project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or university.

By

Onteru Nagaraju	(18131A05C6)
Palli Vamsi Krishna	(18131A05D3)
P. Siddharth	(18131A05D4)
S.Tejeswar Reddy	(18131A05G2)

ACKNOWLEDGEMENT

We thank **Prof. A. B. Koteswara Rao, principal, Gayatri Vidya Parishad College of Engineering (Autonomous)** for extending his utmost support and cooperation in providing all the provisions for the successful completion of the project. We consider it our privilege to express our deepest gratitude **Dr D N D Harini, Head of Department, Computer Science and Engineering** for his valuable suggestions and constant motivation that greatly helped the project work to get successfully completed.

We would like to thank our internal guide **Dr. Ch. Sitha Kumari**. We are thankful for Your support, guidance and valuable suggestions. We also thank all the members of the staff in Computer Science Engineering for their sustained help in our pursuits. We thank all those who contributed directly or indirectly in successfully carrying out this work

Project Members

Onteru Nagaraju	(18131A05C6)
P. Vamsi Krishna	(18131A05D3)
P. Siddharth	(18131A05D4)
S. Tejeswar Reddy	(18131A05G2)

ABSTRACT

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centres globally. Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications. For example, healthcare companies are using the cloud to develop more personalized treatments for patients.

This report briefly explains how to use frameworks and develop an application in Python in a very short time. An application to manage employee CRUD database is taken as a case study. An attempt is made to provide High quality user experience in the application. The web application will be hosted in a public cloud using AWS, Cloud services provided by Amazon. EC2 will be useful in launching virtual servers, configure security and networking, and manage storage. Amazon provides an enriched portal to do all the required administrative actions in a simple way. VPC (Virtual Private Cloud) feature provided by Amazon will enhance the security to Website.

In a traditional web hosting environment, Web servers may not respond if there are huge number of requests per second at peak times due to limitation of hardware resources. However, in the cloud environment, the resources are many and elasticity feature of a cloud will smoothen the peak loads. Cloud environments provides automatic balancing of work load among storage servers and compute servers. Access to the AWS resources can be restricted using the Identity and Access Management (IAM). AWS also provides VPC, which can be used to launch AWS resources in a logically isolated virtual network.

Flask, popular Python web framework, will be explored which helps in developing web applications quickly in an integrated way with MySQL.

INDEX

Name of the chapter:	Page No.
1. INTRODUCTION	
1.1 Introduction of the Project.....	1
2. FLASK APPLICATION.....	3
2.1 What is Flask ?	3
2.2 SqlAlchemy.....	4
2.3 Integrating flask and sqlalchemy.....	6
2.4 Applications of Flask.....	7
2.5 Creating Flask Crud Application	9
3. VPC.....	10
3.1 What is VPC?	10
3.2 Creating a Non-default VPC.....	13
3.3 Viewing Information about Your VPC.....	16
4. LAUNCHING INSTANCE USING EC2.....	17
4.1 Introduction to EC2.....	17
4.2 Launching an EC2 instance	19
4.3 Configuring Details to Ec2.....	25
4.4 Connecting to Linux Instance from Windows using PuTTY.....	30
5. ADVANTAGES, DISADVANTAGES AND APPLICATIONS.....	39
6. RESULTS AND CONCLUSION.....	40
7. REFERENCES.....	41

CHAPTER ONE

INTRODUCTION

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file.

Flask is also extensible and doesn't force a particular directory structure or require complicated boilerplate code before getting started.

It uses Bootstrap toolkit to style your application so it is more visually appealing. Bootstrap will help you incorporate responsive web pages in your web application so that it also works well on mobile browsers without writing your own HTML, CSS, and JavaScript code to achieve these goals. The toolkit will allow you to focus on learning how Flask works. Flask uses the Jinja template engine to dynamically build HTML pages using familiar Python concepts such as variables, loops, lists, and so on.

For businesses, the cloud has the potential to transform operations, as well as cut costs. Offices running computer networks would no longer have to deal with software installation for each computer, as well as licenses. This alleviates a tremendous IT load. Uses of the cloud include data storage, offering remote access to any work related data. In addition to all the uses of cloud computing, from an IT or administrative view point of view, cloud computing is fairly easy to manage. Cloud computing reduces the load on servers, and the IT team as well. It centralizes and unifies computing standards. A new implementation can quickly take on cloud behaviour as soon as it is deployed on the cloud. Every user who accesses the product will have access to the same standard product.

The objectives of the project are :

- Create a CRUD APPLICATION which can create, remove, update, delete records of employee and give a user friendly experience using flask framework .

- Creating a VPC with public subnet so as to access all services of public cloud provided by amazon web services
- Launching our instance on EC2 for running applications on the Amazon Web Services (AWS) infrastructure.

CHAPTER TWO

FLASK

2.1 What is Flask?

Flask is a web application framework written in Python. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is a micro web Framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies and several common framework related tools

Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

2.1.1 Installation:

We will require two package to setup your environment. virtualenv for a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries and the next will be Flask itself. virtualenv

pip install virtualenv Flask

pip install Flask

2.2 SQLAlchemy?

Using raw SQL in Flask web applications to perform CRUD operations on database can be tedious. Instead, **SQLAlchemy**, a Python toolkit is a powerful **OR Mapper** that gives application developers the full power and flexibility of SQL. Flask-SQLAlchemy is the Flask extension that adds support for SQLAlchemy to your Flask application.

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

2.3 Integrating Flask and SQLAlchemy

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.

SQLAlchemy connects to relational databases and what relational databases are really good at are relations

For the common case of having one Flask application all you have to do is to create your Flask application, load the configuration of choice and then create the **SQLAlchemy** object by passing it the application.

2.2.2 Installation

Step 1 – Install Flask-SQLAlchemy extension.

Pip install flask-sqlalchemy

Step 2 – You need to import SQLAlchemy class from this module.

From flask_sqlalchemy import SQLAlchemy

2.4 Applications of flask

- **WSGI** Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.
- **Werkzeug** It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.
- **jinja2** - jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.
- **Flask** is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have form a validation support. Instead, Flask supports the extensions to add such functionality to the application.

2.5 Block Diagram

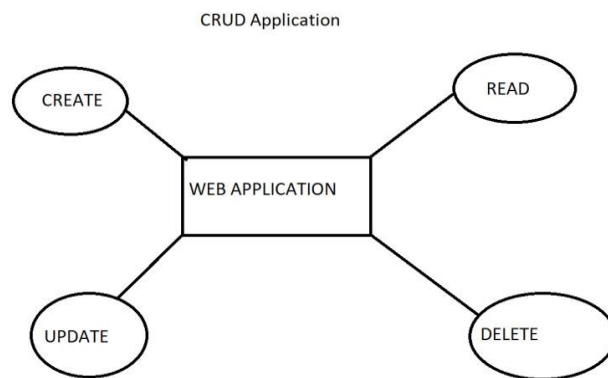


Fig 2.1 : CRUD Application

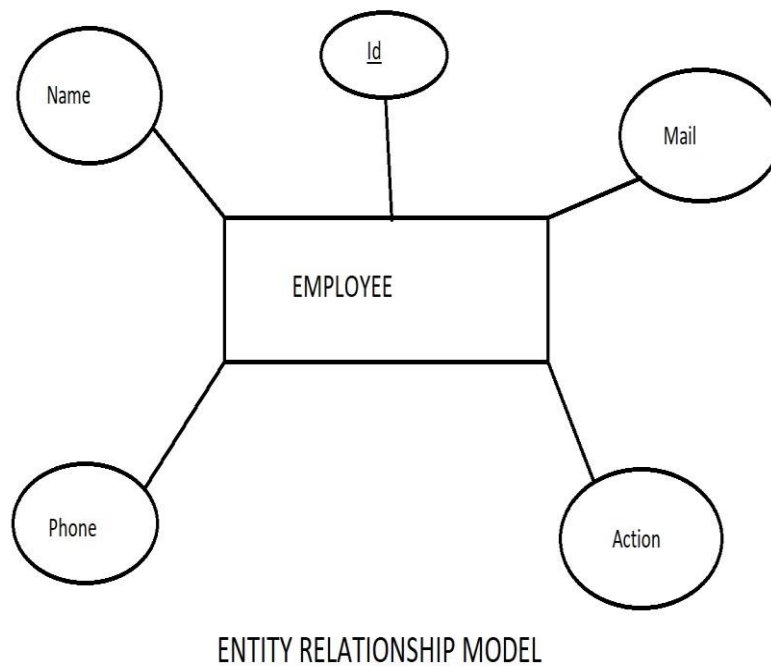


Fig 2.2: Entity Relationshio Model

2.6 Creating flask Application

Create a static folder and download BootstrapCDN folder and jquery files in there. In the templates folder we include the html files. Application initial run application is app.py, this is application is run by flask first. CSS and Scripts are added in the static folder.

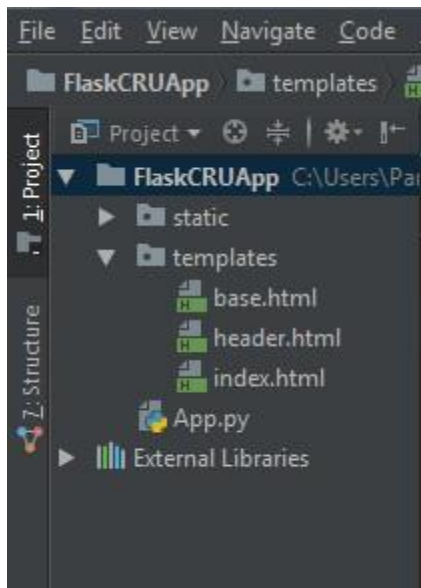
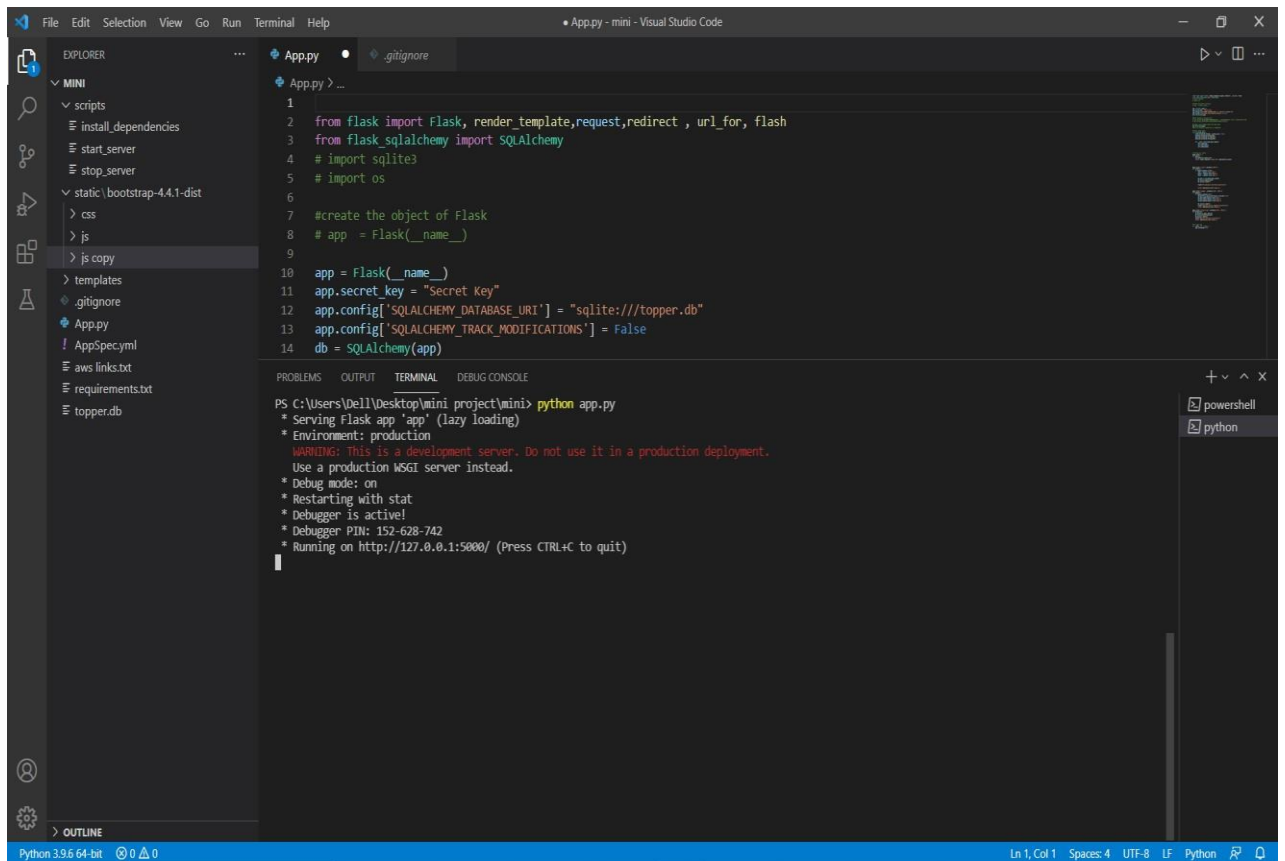


Fig 2.3 : Project Setup

In this code we have created our Mysql Database configuration. you can see that our database name is crud. And this is our database model.

So now run the complete project and this will be the result.

Running the code:



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays the contents of App.py, which is a Flask application. The terminal shows the output of running the application, including a warning about the development server and the URL to access the application.

```
1
2 from flask import Flask, render_template, request, redirect, url_for, flash
3 from flask_sqlalchemy import SQLAlchemy
4 # import sqlite3
5 # import os
6
7 #create the object of Flask
8 # app = Flask(__name__)
9
10 app = Flask(__name__)
11 app.secret_key = "Secret Key"
12 app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///topper.db"
13 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
14 db = SQLAlchemy(app)
```

PS C:\Users\De1\Desktop\mini project\mini> python app.py
* Serving flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 152-628-742
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Fig 2.4 : Project Overview Code Window

Output of our Website is:

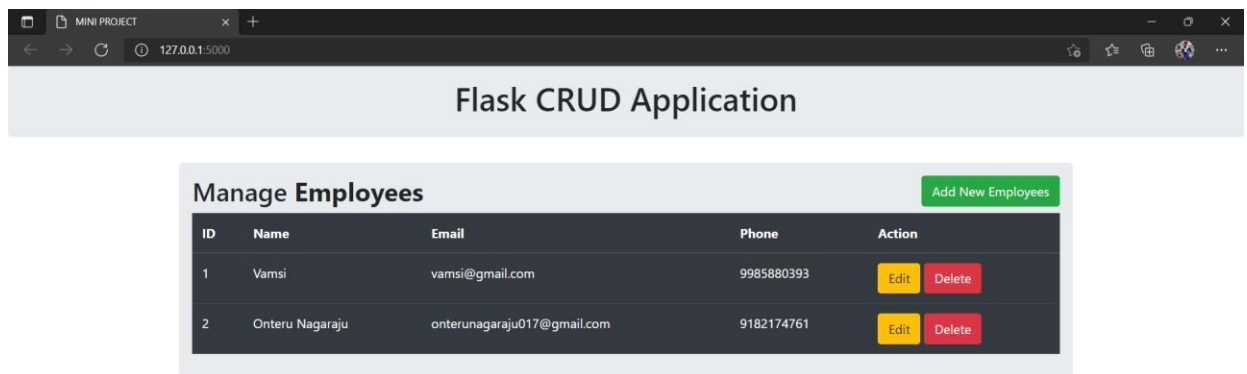


Fig 2.5 : Web Application overlook

CHAPTER THREE

VPC

3.1 What is Virtual Private Cloud?

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. Amazon VPC is the networking layer for Amazon EC2.

The following are the key concepts for VPCs:

- **Virtual private cloud (VPC)** — A virtual network dedicated to your AWS account.
- **Subnet** — A range of IP addresses in your VPC.
- **Route table** — A set of rules, called routes, that are used to determine where network traffic is directed.
- **Internet gateway** — A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.
- **VPC endpoint** — enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network. For more information, see AWS Private Link and VPC endpoints.
- **CIDR block** — Classless Inter-Domain Routing. An internet protocol address allocation and route aggregation methodology. For more information, see Classless Inter-Domain Routing in Wikipedia.

3.1.1 Access Amazon VPC

You can create, access, and manage your VPCs using any of the following interfaces:

- **AWS Management Console** — provides a web interface that you can use to access your VPCs.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including Amazon VPC, and is supported on Windows, Mac, and Linux.
- **AWS SDKs** — Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling.

□ **Query API** — provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Amazon VPC, but it requires that your application handle low-level details such as generating the hash to sign the request, and error handling.

3.2 creating a non-default VPC

3.2.1 Requirements

- A GitHub repository with files for your working Flask application. Make sure that the application that you want to deploy into AWS has secured required environment variables and is ready to deploy.
- Create a free account or sign in to your AWS console.
- A credit card for AWS to have on file in case you surpass the Free Tier eligibility options. It is worth noting that you should take extra precautions if you are deploying an app onto AWS.

□ Tmux to run the application in a terminal session

3.2.1 Create a User Account:

In order to deploy fast and easily, create an AWS account. Upon logging back into your account, you have the option to login as a Root user or an IAM user. I would recommend logging in as a Root user account to perform tasks requiring unrestricted access or creating an IAM user account that holds all of the permissions that a Root user would have. IAM users have the ability to work on the AWS dashboard with secure control access that can be modified.

3.3 Creating a Non- Default VPC

3.3.1 Create the VPC

In this step, you'll use the Amazon VPC wizard in the Amazon VPC console to create a VPC. The wizard performs the following steps for you:

- Creates a VPC with a /16 IPv4 CIDR block (a network with 65,536 private IP addresses).
 - Attaches an internet gateway to the VPC.
 - Creates a size /24 IPv4 subnet (a range of 256 private IP addresses) in the VPC.
 - Creates a custom route table, and associates it with your subnet, so that traffic can flow between the subnet and the internet gateway.
1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
 2. In the navigation bar, on the top-right, take note of the **AWS Region** in which you'll be creating the VPC. Ensure that you continue working in the same Region for the rest of this exercise, as you cannot launch an instance into your VPC from a different Region.
 3. In the navigation pane, choose **VPC dashboard**. From the dashboard, choose **Launch VPC Wizard**.

Note

Do not choose **Your VPCs** in the navigation pane; you cannot access the VPC wizard using the **Create VPC** button on that page.

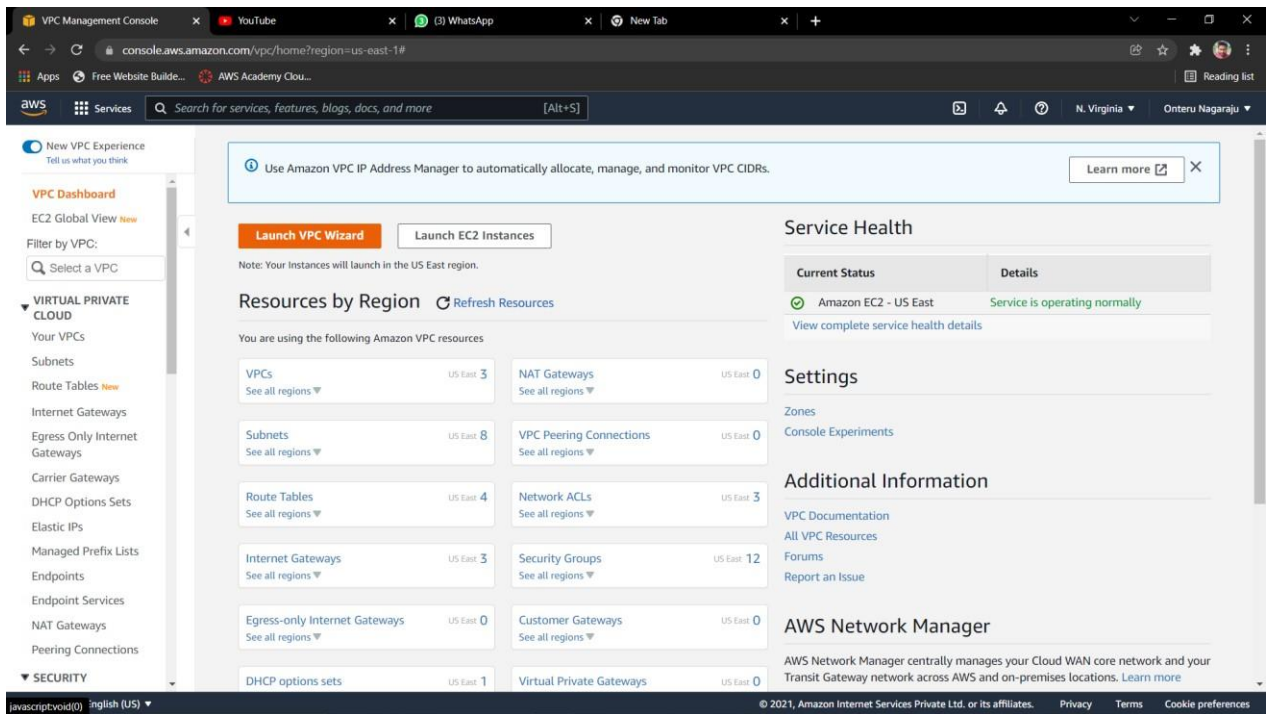


Fig 3.1 : Launching VPC

4. Choose **VPC with a Single Public Subnet**, and then choose **Select**.

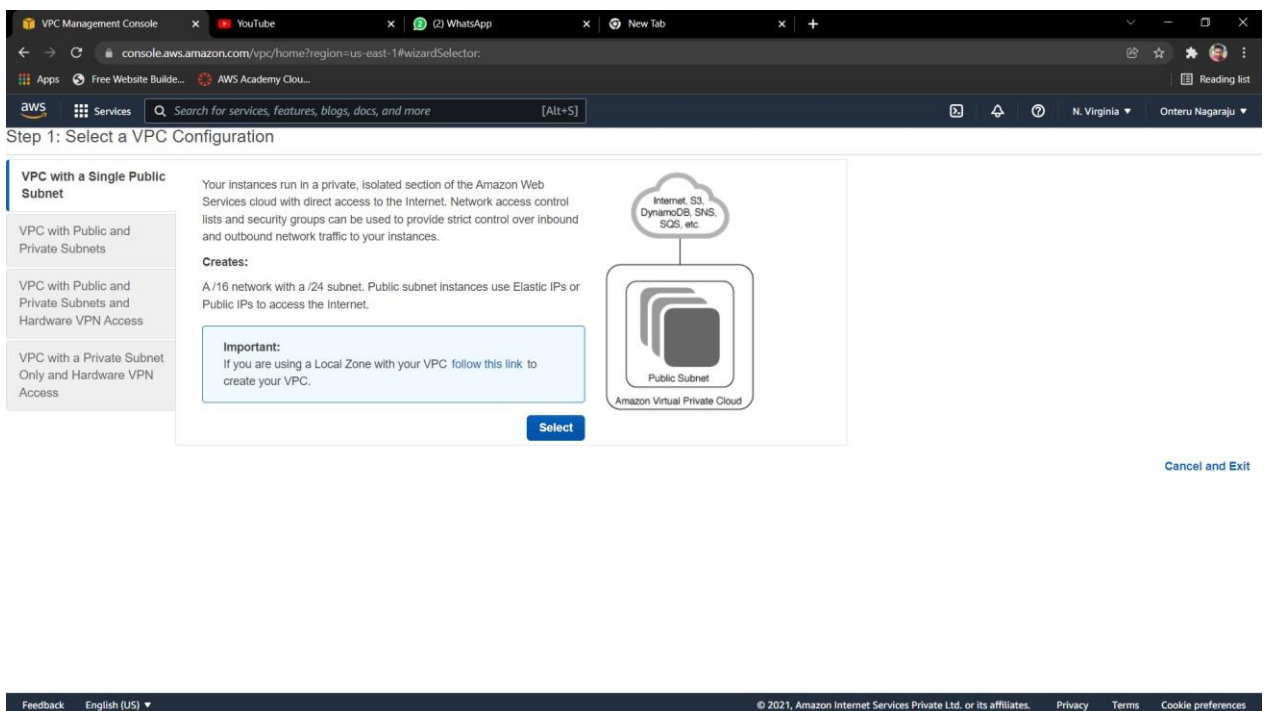


Fig 3.2 : selecting VPC Configuration

5. On the configuration page, enter a name for your VPC in the **VPC name** field; for example, my-
vpc, and enter a name for your subnet in the **Subnet name** field.

This helps you to identify the VPC and subnet in the Amazon VPC console after you've created them. For this exercise, leave the rest of the configuration settings on the page, and choose **Create VPC**.

The screenshot shows the AWS VPC Management Console interface. The browser tabs include 'VPC Management Console', 'YouTube', 'WhatsApp', and 'New Tab'. The address bar shows the URL 'console.aws.amazon.com/vpc/home?region=us-east-1#wizardFullpagePublicOnly'. The console header includes the AWS logo, 'Services', a search bar, and the region 'N. Virginia' with the user 'Onteru Nagaraju'. The main content area is titled 'Step 2: VPC with a Single Public Subnet'. It contains several configuration fields: 'IPv4 CIDR block' set to '10.0.0.0/16' (65531 IP addresses available), 'IPv6 CIDR block' with radio buttons for 'No IPv6 CIDR Block' (selected), 'Amazon provided IPv6 CIDR block', and 'IPv6 CIDR block owned by me'. The 'VPC name' field contains 'My VPC'. Below this, 'Public subnet's IPv4 CIDR' is set to '10.0.0.0/24' (251 IP addresses available), 'Availability Zone' is set to 'No Preference', and 'Subnet name' is 'Public subnet'. A note states 'You can add more subnets after Amazon Web Services creates the VPC.'. There is an 'Add Endpoint' button under 'Service endpoints'. At the bottom, 'Enable DNS hostnames' is set to 'Yes' and 'Hardware tenancy' is set to 'Default'. The bottom right has 'Cancel and Exit', 'Back', and 'Create VPC' buttons. The footer includes 'Feedback', 'English (US)', and copyright information for Amazon Internet Services Private Ltd.

Fig 3.3 : Selecting VPC with a Single Public Subnet

6. A status window shows the work in progress. When the work completes, choose **OK** to close the status window.

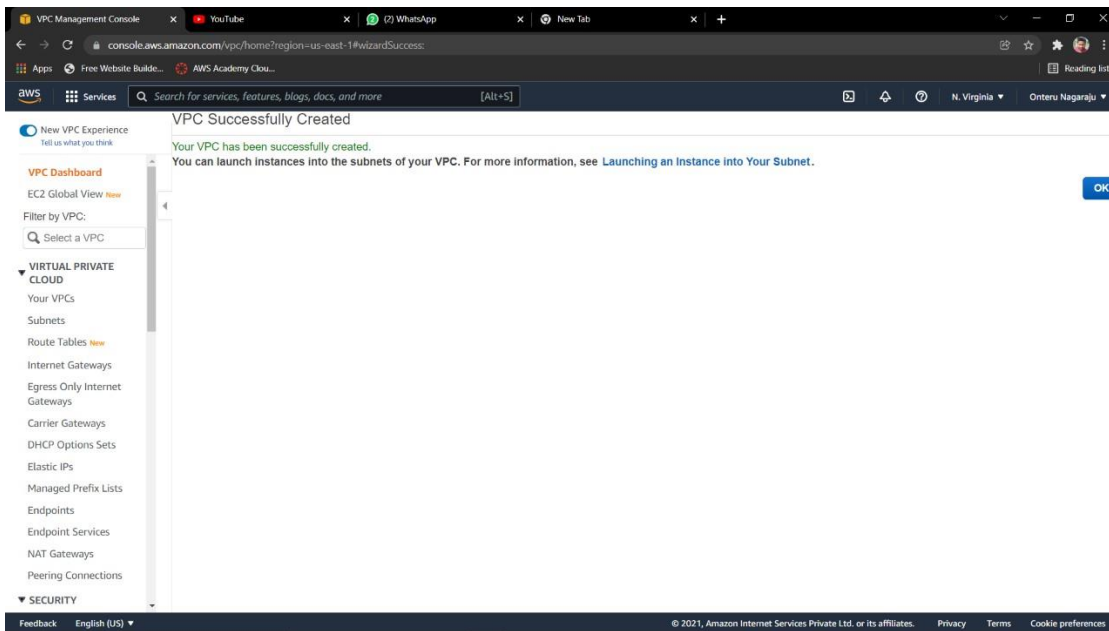


Fig 3.4 : VPC Created

The **Your VPCs** page displays your default VPC and the VPC that you just created. The VPC that you created is a nondefault VPC, therefore the **Default VPC** column displays **No**.

3.4 View information about your VPC:

After creating the VPC, you can view information about the subnet, the internet gateway, and the route tables. The VPC that you created has two route tables — a main route table that all VPCs have by default, and a custom route table that was created by the wizard. The custom route table is associated with your subnet, which means that the routes in that table determine how the traffic for the subnet flows. If you add a new subnet to your VPC, it uses the main route table by default.

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs**. Take note of the name and the ID of the VPC that you created (look in the **Name** and **VPC ID** columns). You will use this information to identify the components that are associated with your VPC.
3. In the navigation pane, choose **Subnets**. The console displays the subnet that was created when you created your VPC. You can identify the subnet by its name in **Name** column, or you can use the VPC information that you obtained in the previous step and look in the **VPC** column.

4. In the navigation pane, choose **Internet Gateways**. You can find the internet gateway that's attached to your VPC by looking at the **VPC** column, which displays the ID and the name (if applicable) of the VPC.
5. In the navigation pane, choose **Route Tables**. There are two route tables associated with the VPC. Select the custom route table (the **Main** column displays **No**), and then choose the **Routes** tab to display the route information in the details pane:
 - The first row in the table is the local route, which enables instances within the VPC to communicate. This route is present in every route table by default, and you can't remove it.
 - The second row shows the route that the Amazon VPC wizard added to enable traffic destined for the internet (0.0.0.0/0) to flow from the subnet to the internet gateway.
6. Select the main route table. The main route table has a local route, but no other routes.

CHAPTER FOUR

LAUNCHING INSTANCE USING EC2

4.1 Elastic Cloud Compute

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment.

Amazon EC2 offers the broadest and deepest compute platform with choice of processor, storage, networking, operating system, and purchase model. It offers the fastest processors in the cloud and we are the only cloud with 400 Gbps Ethernet networking. It has the most powerful GPU instances for machine learning training and graphics workloads, as well as the lowest cost-per-inference instances in the cloud.

4.2 Launching Instance

When launching an EC2 instance into a VPC, you must specify the subnet in which to launch the instance. In this case, we will launch an instance into the public subnet of the VPC you created. We will use the Amazon EC2 launch wizard in the Amazon EC2 console to launch your instance.

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar, on the top-right, ensure that you select the same Region in which you created your VPC.
3. From the dashboard, choose Launch Instance.
4. On the first page of the wizard, choose the AMI that you want to use. For this exercise, choose an Amazon Linux AMI or a Windows AMI.
5. On the Choose an Instance Type page, you can select the hardware configuration and size of the instance to launch. By default, the wizard selects the first available instance type based on the AMI you selected. You can leave the default selection, and then choose Next: Configure Instance Details.

6. On the Configure Instance Details page, select the VPC that you created from the Network list, and the subnet from the Subnet list. Leave the rest of the default settings, and go through the next pages of the wizard until you get to the Add Tags page.
7. On the Add Tags page, you can tag your instance with a Name tag; for example Name=My Webserver. This helps you to identify your instance in the Amazon EC2 console after you've launched it. Choose Next: Configure Security Group when you are done.
8. On the Configure Security Group page, the wizard automatically defines the launch-wizard-x security group to allow you to connect to your instance. Choose Review and Launch.

Important

The wizard creates a security group rule that allows all IP addresses (0.0.0.0/0) to access your instance using SSH or RDP. This is acceptable for the short exercise, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your instance.

9. On the **Review Instance Launch** page, choose **Launch**.
10. In the **Select an existing key pair or create a new key pair** dialog box, you can choose an existing key pair, or create a new one. If you create a new key pair, ensure that you download the file and store it in a secure location. You'll need the contents of the private key to connect to your instance after it's launched.

To launch your instance, select the acknowledgment check box, and then choose **Launch Instances**.
11. On the confirmation page, choose **View Instances** to view your instance on the **Instances** page. Select your instance, and view its details in the **Description** tab. The **Private IPs** field displays the private IP address that's assigned to your instance from the range of IP addresses in your subnet.

4.3 Configuring Details to EC2

Click on the *Services* tab at the top of the webpage. Click on “EC2” under the *Compute* tab or type the name into the search bar to access the EC2 dashboard.

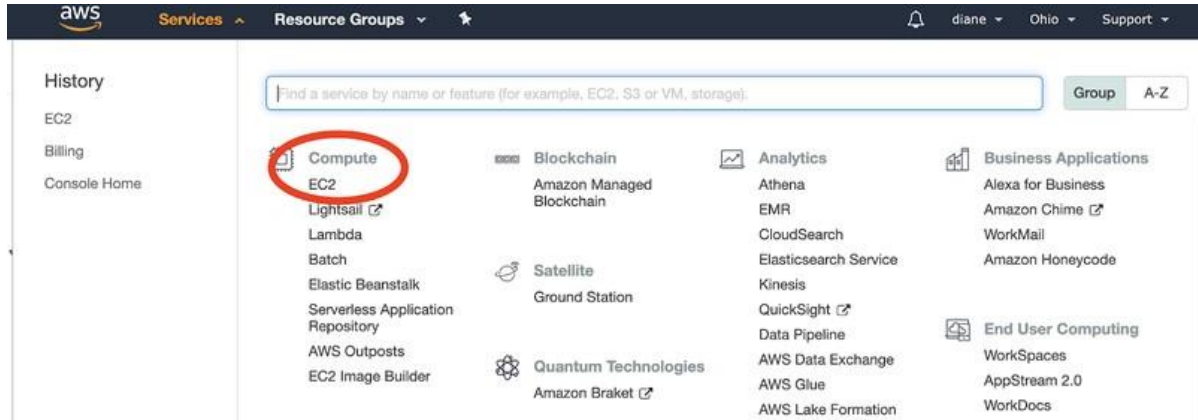


Fig 5.1 : Configuring Details to EC2

EC2 is a virtual server in the cloud where the our web app will live.

Look for the *Launch Instance* section of the web page. It should be an orange button labeled *Launch Instance* as shown below. You can see that the section says “Note: Your instances will launch in the US East (Ohio) Region”. This may vary for your EC2 dashboard, as you want to make sure that your instances are within the US states if you are a developer in the US.

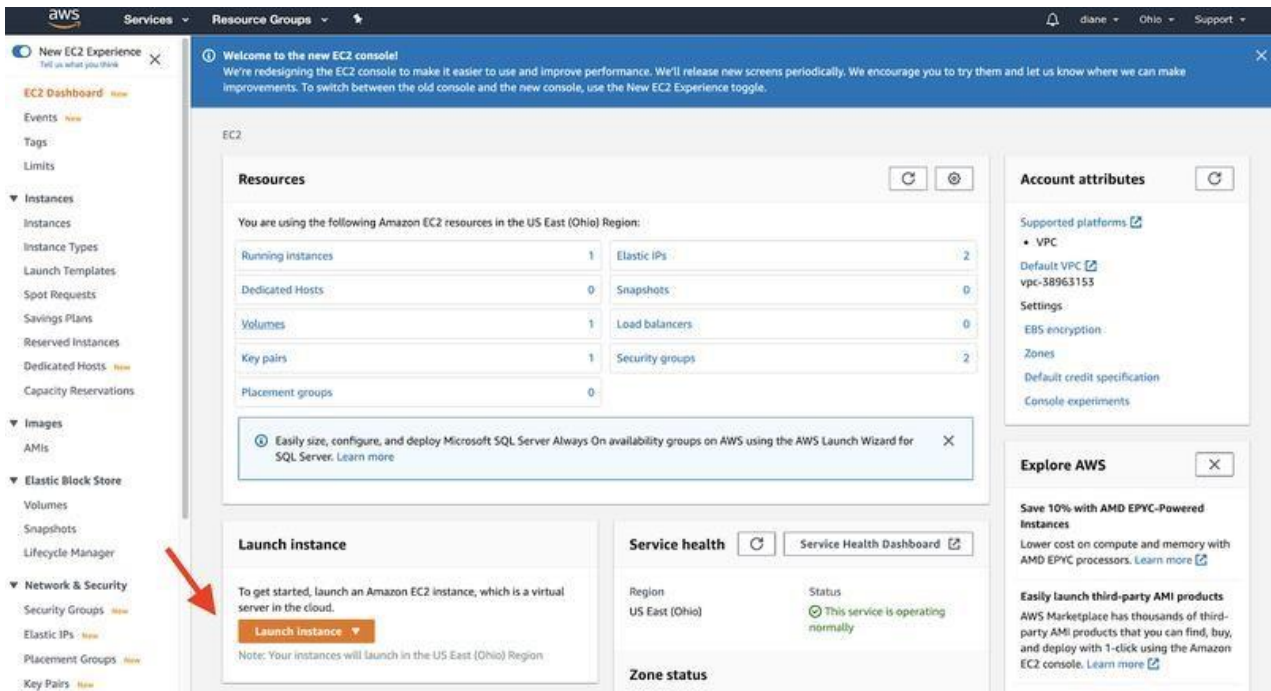


Fig 4.2 : Launching EC2 Instance

STEP 1: Select Free Tier Instances for the Machine Image

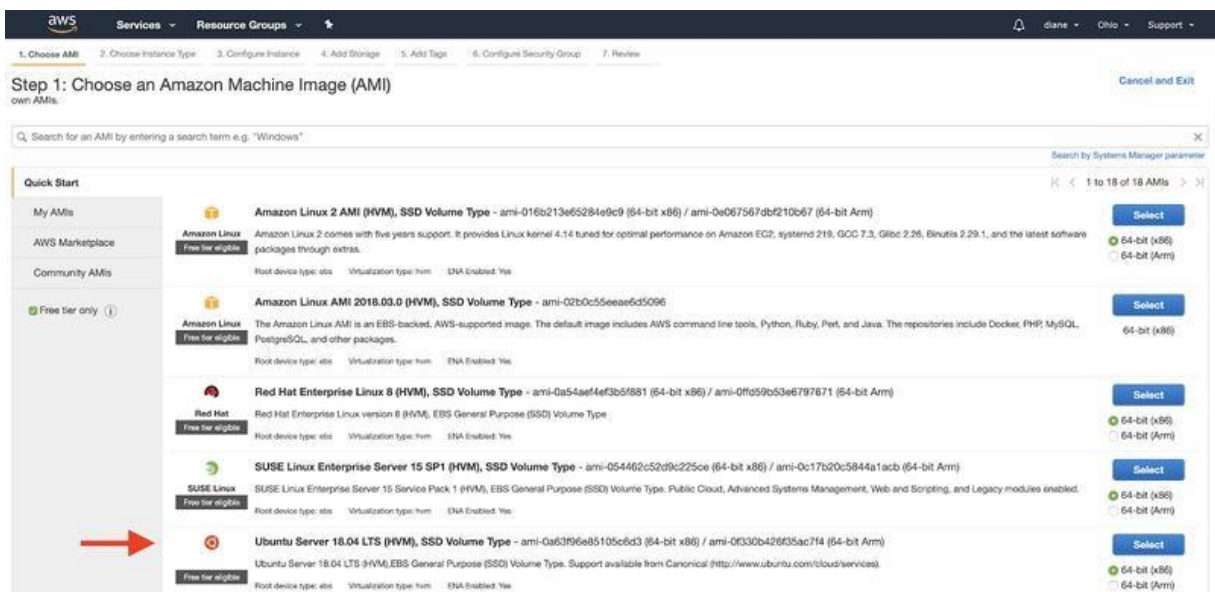


Fig 4.3 : Selecting OS

STEP 2: Choose the Instance Type:

Select the **Instance** with the *Free tier eligible* option. After checking the instance you want to use, click on **Next: Configure Instance Details**.

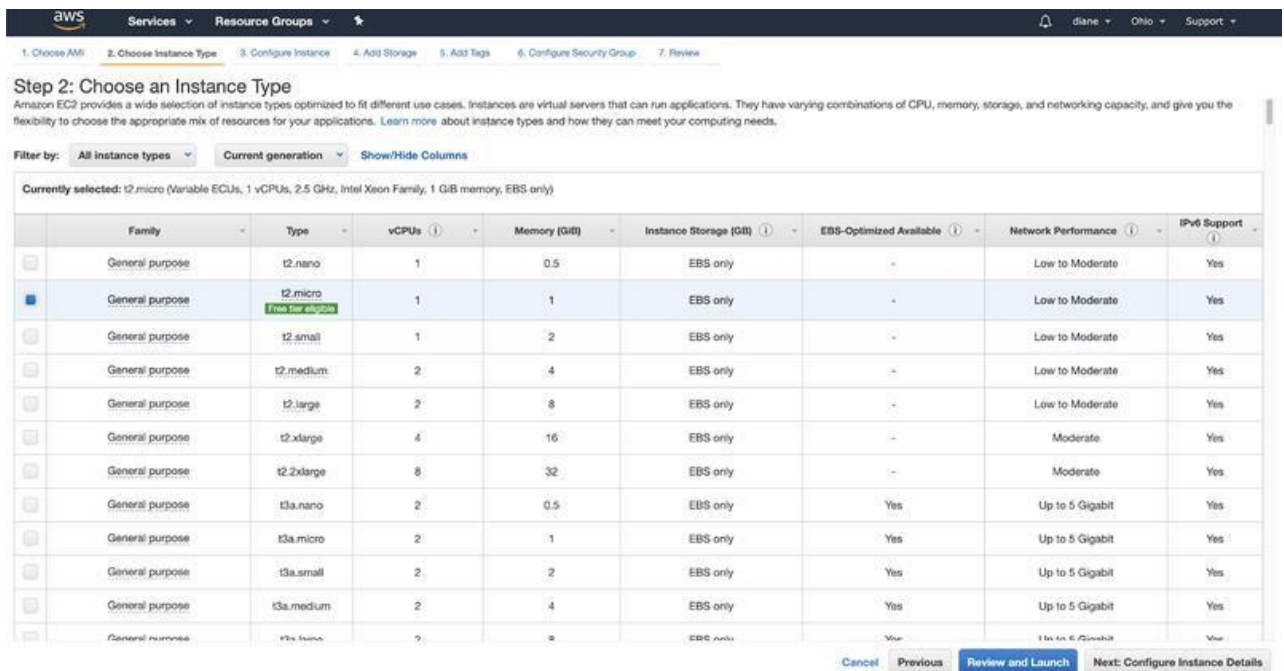


Fig 4.4 : Selecting instance Type

STEP 3: Configure Instance Details

View the default settings and select Network as your Non-default VPC and move on to **Next: Add Storage**.

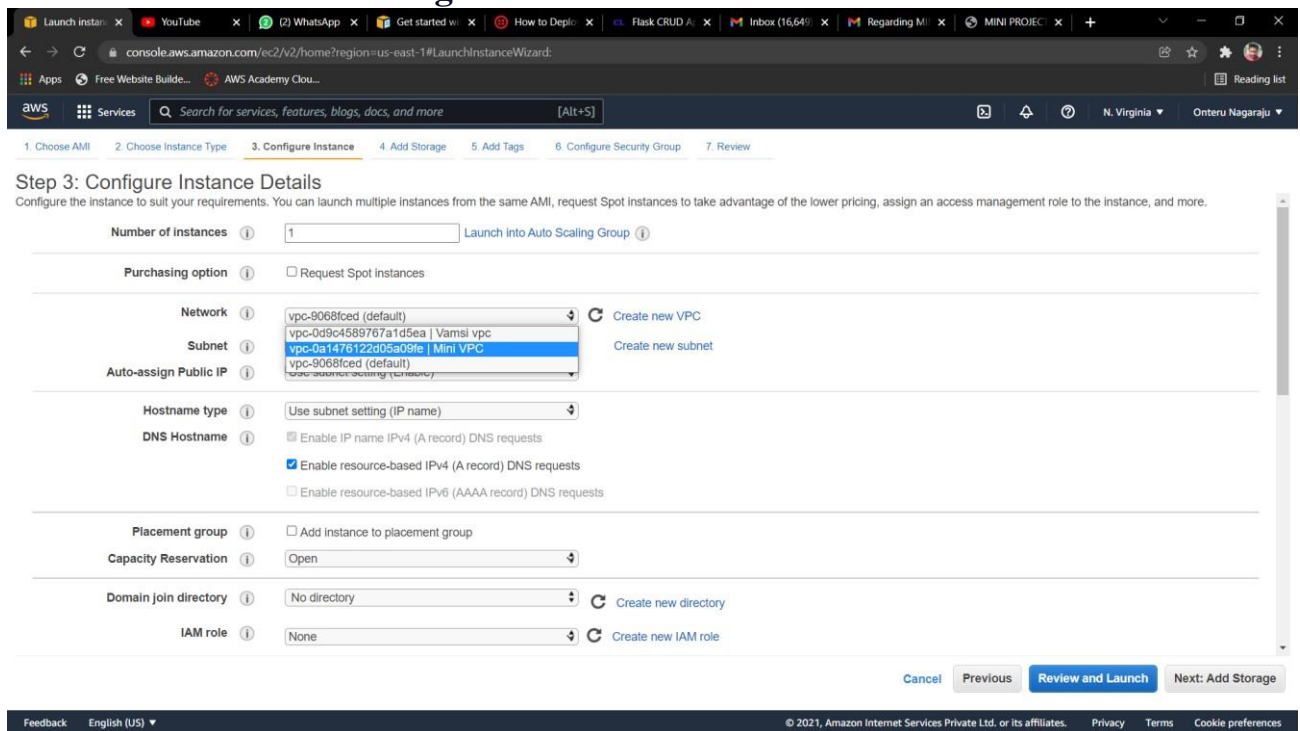


Fig 4.5 : Selecting Network

STEP 4: Add storage

Select the amount of storage necessary to run your application. Free tier eligible customers can get up to 30 GB. The storage used will depend on your application. For example, if your application requires image file storage, heavy graphic rendering, or storing user data it will use more storage. In this case, you will have to be careful and make sure your app does not go over the free GB allocation. If you go over your limit for the month, you will be charged.

After you select the size you want, click **Next: Add Tags**.

The screenshot shows the 'Step 4: Add Storage' configuration page in the AWS Management Console. At the top, a progress bar indicates the current step is '4. Add Storage', with previous steps being '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '5. Add Tags', '6. Configure Security Group', and '7. Review'. Below the progress bar, the title 'Step 4: Add Storage' is followed by a descriptive paragraph. The main configuration area contains a table with columns for 'Volume Type', 'Device', 'Snapshot', 'Size (GiB)', 'Volume Type', 'IOPS', 'Throughput (MB/s)', 'Delete on Termination', and 'Encryption'. A single row is visible for the 'Root' volume, with device '/dev/sda1', snapshot 'snap-021b833c41d050331', size '8', volume type 'General Purpose SSD (gp2)', IOPS '100 / 3000', throughput 'N/A', 'Delete on Termination' set to 'On', and 'Encryption' set to 'Not Encrypted'. Below the table is an 'Add New Volume' button. At the bottom, a blue-bordered box contains a note about the free tier storage limit.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-021b833c41d050331	8	General Purpose SSD (gp2)	100 / 3000	N/A	On	Not Encrypted

[Free tier eligible customers can get up to 30 GB of EBS General Purpose \(SSD\) or Magnetic storage. \[Learn more\]\(#\) about free usage tier eligibility and usage restrictions.](#)

Fig 4.6 : Adding Storage

STEP 5: Add Tags

[Tags are used](#) to categorize your AWS resources for different use cases to easily keep track of your resources. This would apply if you are working on large scale projects and need to organize the AWS billing costs in a preferred structure. Thus, it might not be necessary to add tags to your machine especially if you only plan on using AWS one time for this specific application. Go ahead and click **Next: Configure Security Group**.

The screenshot shows the 'Step 5: Add Tags' configuration page in the AWS Management Console. At the top, a progress bar indicates the current step is '5. Add Tags', with previous steps being '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '6. Configure Security Group', and '7. Review'. Below the progress bar, the title 'Step 5: Add Tags' is followed by a descriptive paragraph. The main configuration area contains a table with columns for 'Key', 'Value', 'Instances', and 'Volumes'. The 'Key' and 'Value' columns have input fields with character limits (128 and 256 respectively). Below the table, a message states 'This resource currently has no tags' and provides instructions on how to add tags. At the bottom, there is an 'Add Tag' button and a note about the maximum number of tags.

Key	Value	Instances	Volumes
(128 characters maximum)	(256 characters maximum)		

This resource currently has no tags.

Choose the [Add tag](#) button or [click to add a Name tag](#).
Make sure your [IAM policy](#) includes permissions to create tags.

[Add Tag](#) (Up to 50 tags maximum)

Fig 4.7 : Adding tags

STEP 6: Security Group – KEY VALUE PAIR

Configuring the security is the last step before launching the AWS EC2 instance. If this is your first time creating a security group, select *SSH* in the dropdown under **Type**. Everything else in this section should be set to default to *TCP* at Port 22.

Fig 4.8 : Configuring Security Group

Here's an example of the security types and ports you can use in your web application. Since we want people around the world to access the site, set the **Source** to these protocols to "0.0.0.0/0".

Here's the list of security types and protocols you should use. You can add additional rules to the following list as well:

Set **Type** *HTTP*, **Protocol** *TCP*, **Port range** 80, and **Source** to "0.0.0.0/0".

Set **Type** *HTTP*, **Protocol** *TCP*, **Port range** 80, and **Source** to ":::/0".

Set **Type** *Custom TCP*, **Protocol** *TCP*, **Port range** 8080, and **Source** to "0.0.0.0/0".

Set **Type** *SSH*, **Protocol** *TCP*, **Port range** 22, and **Source** to "0.0.0.0/0".

Set **Type** *HTTPS*, **Protocol** *TCP*, **Port range** 443, and **Source** to "0.0.0.0/0".

Launch and create a key pair

Click "Review and Launch" to confirm your security group for your project. After you review your settings and click "Launch" you will be prompted to select an existing key pair or create a new key pair.

Click on the drop down menu and select **Create a new key pair**. This is essential to access your AWS instance securely through your machine. Give your key pair a memorable name. For the purposes of this article, the key pair name is “dianetwilio-test”.

Click on **Download Key Pair** after creating your key pair. This will download the private key file (learn more about `.pem` files or about public key cryptography).

Drag the `.pem` file to a secure location. It is absolutely crucial that you keep file `.pem` safe, as this is the **ONLY** way to access your web application.

Drag the
this

Select an existing key pair or create a new key pair ✕

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name
diane-twilio-test

Download Key Pair

... You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel **Launch Instances**

Fig 4.10 : Downloading Key pair

STEP 7 : Review and launch the instance

Give your AWS dashboard some time to launch the instance. Your **IPv4 Public IP** is the address you need to access your web application. In this article, the public IP address for the instance is “52.15.127.3”.

The instance has been launched once the **Instance State** tab says *running* along with a green circle.

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

1 to 2 of 2

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name
	i-0fba656be0293d48	t2.micro	us-east-2b	running	Initializing	None	ec2-52-15-127-3.us-east-2.amazonaws.com	52.15.127.3	-	diane-twilio-test

Fig 4.11 : Instance State

STEP 8: Assign an Elastic IP address to your instance

In the previous step, you launched your instance into a public subnet — a subnet that has a route to an internet gateway. However, the instance in your subnet also needs a public IPv4 address to be able to communicate with the internet. By default, an instance in a nondefault VPC is not assigned a public IPv4 address. In this step, you'll allocate an Elastic IP address to your account, and then associate it with your instance.

To allocate and assign an Elastic IP address

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Elastic IPs**.

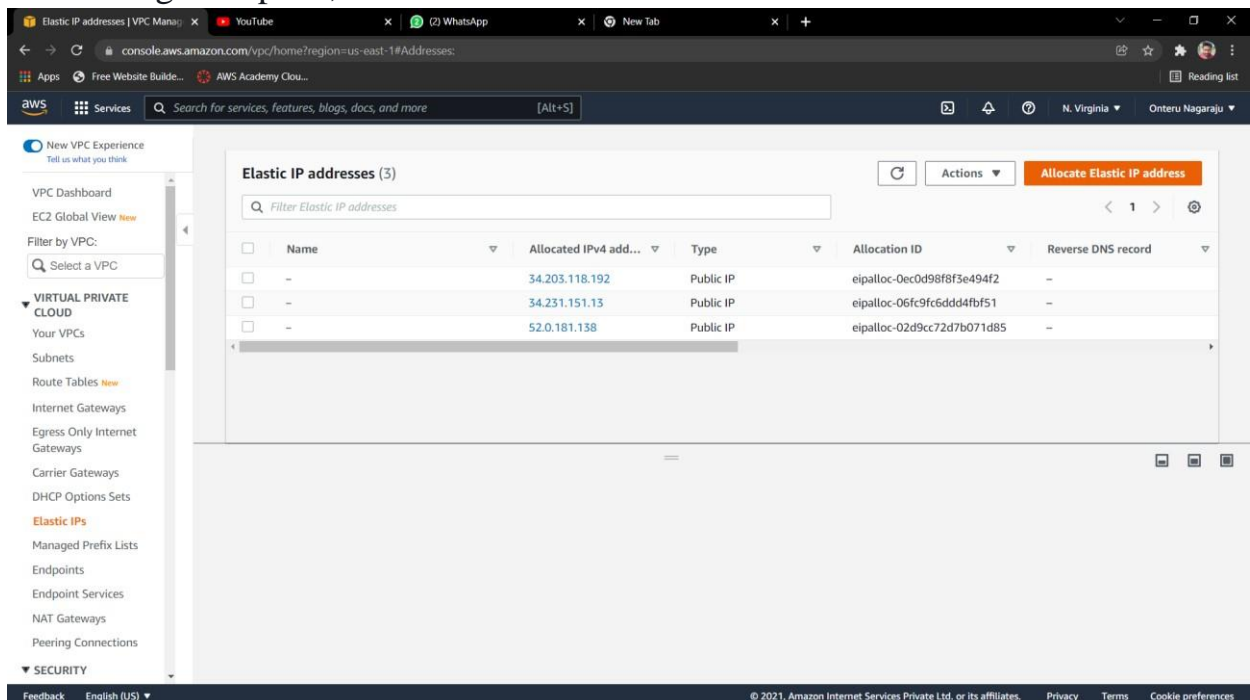


Fig : 4.12 : Allocating Elastic IP Address

3. Choose **Allocate new address**, and then **Allocate**.
4. Select the Elastic IP address from the list, choose **Actions**, and then choose **Associate Address**.

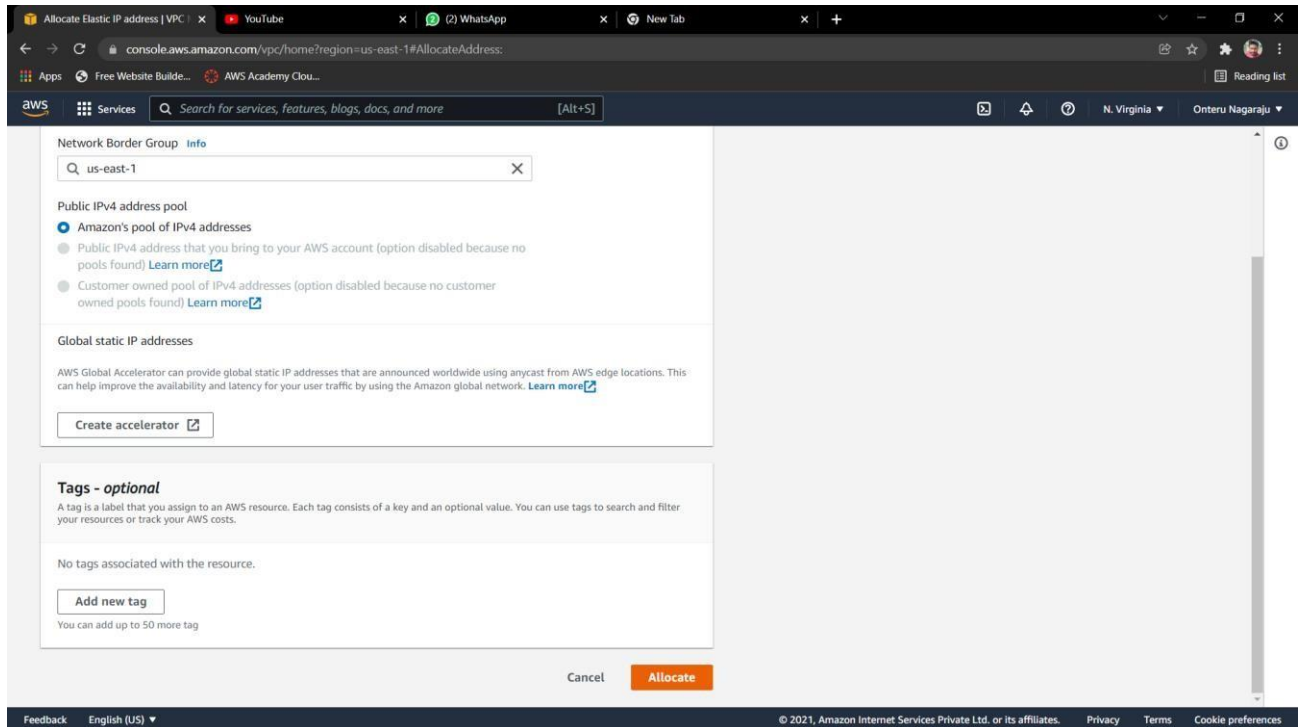


Fig 4.13 : Selecting IPV4 address

5. For **Resource type**, ensure that **Instance** is selected. Choose your instance from the **Instance** list. Choose **Associate** when you're done.

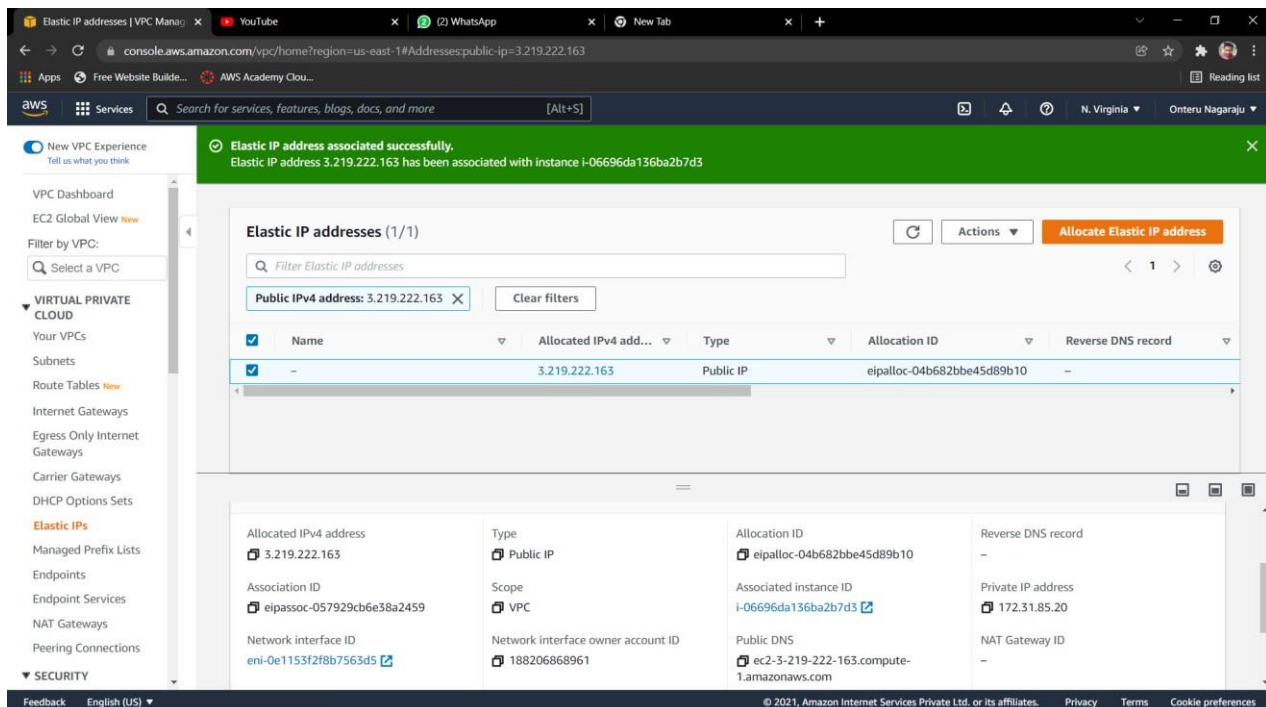
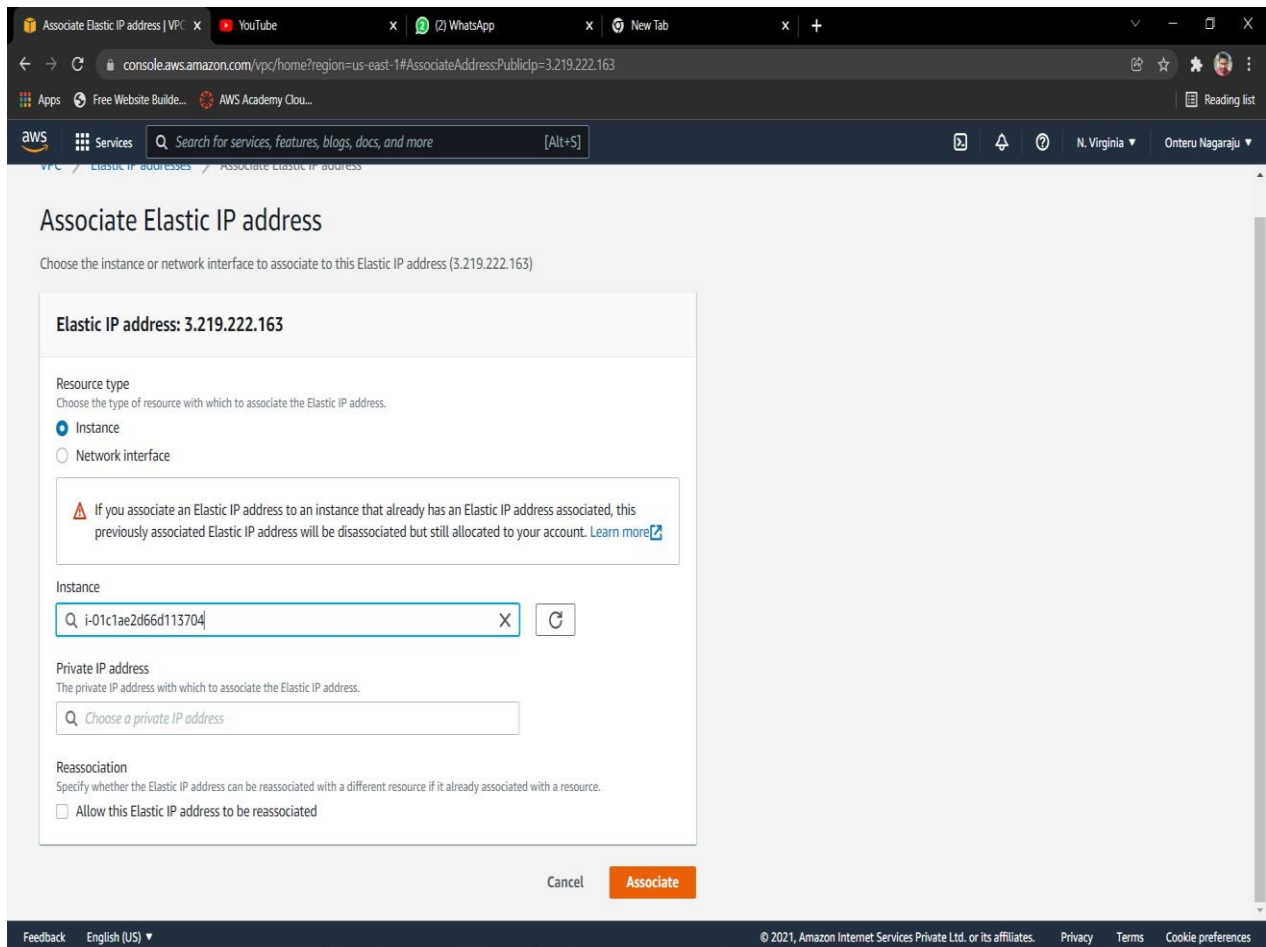


Fig 4.14 : Selecting And Allocating IP address

STEP 8 : Clean up

You can choose to continue using your instance in your VPC, or if you do not need the instance, you can terminate it and release its Elastic IP address to avoid incurring charges for them. You can also delete your VPC — note that you are not charged for the VPC and VPC components created in this exercise (such as the subnets and route tables).

To terminate your instance, release your Elastic IP address, and delete your VPC

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances**.
3. Select your instance, choose **Actions**, then **Instance State**, and then select **Terminate**.
4. In the dialog box, expand the **Release attached Elastic IPs** section, and select the check box next to the Elastic IP address. Choose **Yes, Terminate**.
5. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
6. In the navigation pane, choose **Your VPCs**.
7. Select the VPC, choose **Actions**, and then choose **Delete VPC**.
8. When prompted for confirmation, choose **Delete VPC**.

4.5 Connecting to Linux instance from Windows using Putty

After you launch your instance, you can connect to it and use it the way that you'd use a computer sitting in front of you.

The following instructions explain how to connect to your instance using PuTTY, a free SSH client for Windows.

4.5.1 Prerequisites

Before you connect to your Linux instance using PuTTY, complete the following prerequisites.

1. Verify that the instance is ready

After you launch an instance, it can take a few minutes for the instance to be ready so that you can connect to it. Check that your instance has passed its status checks. You can view this information in the Status check column on the Instances page.

2. Verify the general prerequisites for connecting to your instance

To find the public DNS name or IP address of your instance and the user name that you should use to connect to your instance, see [General prerequisites for connecting to your instance](#). Install PuTTY on your local computer

Download and install PuTTY from the [PuTTY download page](#). If you already have an older version of PuTTY installed, we recommend that you download the latest version. Be sure to install the entire suite. Convert your private key using PuTTYgen

Locate the private key (.pem file) for the key pair that you specified when you launched the instance. Convert the .pem file to a .ppk file for use with PuTTY. For more information, follow the steps in the next section.

3.Convert your private key using PuTTYgen

PuTTY does not natively support the private key format for SSH keys. PuTTY provides a tool named PuTTYgen, which converts keys to the required format for PuTTY. You must convert your private key (.pem file) into this format (.ppk file) as follows in order to connect to your instance using PuTTY.

To convert your private key

1. From the **Start** menu, choose **All Programs, PuTTY, PuTTYgen**.
2. Under **Type of key to generate**, choose **RSA**. If your version of PuTTYgen does not include this option, choose **SSH-2 RSA**.

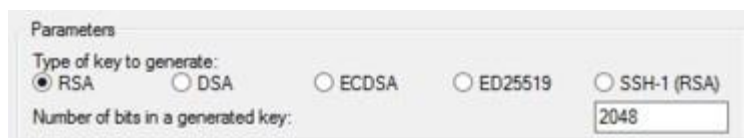


Fig 4.15 : Converting private key

3. Choose **Load**. By default, PuTTYgen displays only files with the extension .ppk. To locate your .pem file, choose the option to display files of all types.

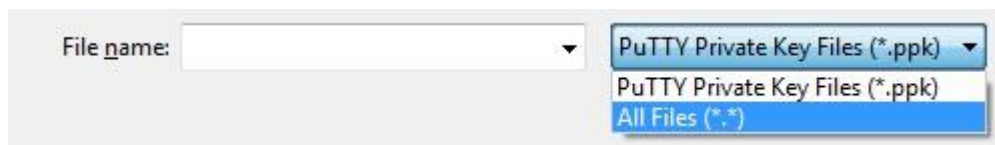


Fig 4.16 : Selecting extension key file

1. Select your .pem file for the key pair that you specified when you launched your instance and choose **Open**. PuTTYgen displays a notice that the .pem file was successfully imported. Choose **OK**.
2. To save the key in the format that PuTTY can use, choose **Save private key**. PuTTYgen displays a warning about saving the key without a passphrase. Choose **Yes**.

Note

A passphrase on a private key is an extra layer of protection. Even if your private key is discovered, it can't be used without the passphrase. The downside to using a passphrase is that it makes automation harder because human intervention is needed to log on to an instance, or to copy files to an instance.

5. Specify the same name for the key that you used for the key pair (for example, my-key-pair) and choose **Save**. PuTTY automatically adds the .ppk file extension.

6. Your private key is now in the correct format for use with PuTTY. You can now connect to your instance using PuTTY's SSH client.
-

4.5.2 Connect to your Linux instance

Use the following procedure to connect to your Linux instance using PuTTY. You need the `.ppk` file that you created for your private key. For more information, see [Convert your private key using PuTTYgen](#) in the preceding section. If you receive an error while attempting to connect to your instance, see [Troubleshoot connecting to your instance](#).

To connect to your instance using PuTTY:

1. Start PuTTY (from the **Start** menu, choose **All Programs, PuTTY, PuTTY**).
2. In the **Category** pane, choose **Session** and complete the following fields:
 - a. In the **Host Name** box, do one of the following:
 - (Public DNS) To connect using your instance's public DNS name, enter `my-instance-user-name@my-instance-public-dns-name`.
 - (IPv6) Alternatively, if your instance has an IPv6 address, to connect using your instance's IPv6 address, enter `my-instance-user-name@my-instance-IPv6-address`.

For information about how to get the user name for your instance, and the public DNS name or IPv6 address of your instance, see [Get information about your instance](#).

- b. Ensure that the **Port** value is 22.

Under **Connection type**, select **SSH**.

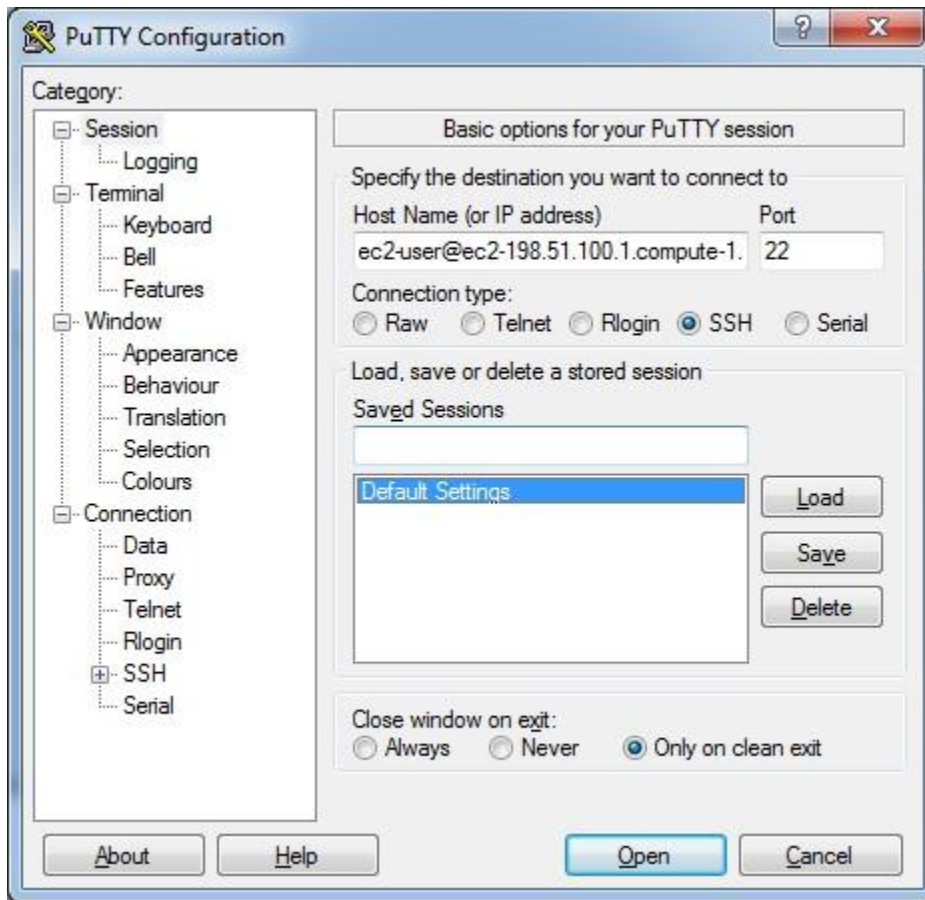


Fig 4.17 : Connecting using IPV4 Address

3. (Optional) You can configure PuTTY to automatically send 'keepalive' data at regular intervals to keep the session active. This is useful to avoid disconnecting from your instance due to session inactivity. In the **Category** pane, choose **Connection**, and then enter the required interval in the **Seconds between keepalives** field. For example, if your session disconnects after 10 minutes of inactivity, enter 180 to configure PuTTY to send keepalive data every 3 minutes. 4. In the **Category** pane, expand **Connection**, expand **SSH**, and then choose **Auth**.

Complete the following:

- Choose **Browse**.
- Select the **.ppk** file that you generated for your key pair and choose **Open**.
- (Optional) If you plan to start this session again later, you can save the session information for future use. Under **Category**, choose **Session**, enter a name for the session in **Saved Sessions**, and then choose **Save**.

- d. Choose **Open**.
- 5. If this is the first time you have connected to this instance, PuTTY displays a security alert dialog box that asks whether you trust the host to which you are connecting.
 - a. (Optional) Verify that the fingerprint in the security alert dialog box matches the fingerprint that you previously obtained in [\(Optional\) Get the instance fingerprint](#). If these fingerprints don't match, someone might be attempting a "man-in-the-middle" attack. If they match, continue to the next step.
 - b. Choose **Yes**. A window opens and you are connected to your instance.

Note

If you specified a passphrase when you converted your private key to PuTTY's format, you must provide that passphrase when you log in to the instance.

4.5.3 Transfer files to your Linux instance using the PuTTY Secure Copy client

The PuTTY Secure Copy client (PSCP) is a command line tool that you can use to transfer files between your Windows computer and your Linux instance. If you prefer a graphical user interface (GUI), you can use an open source GUI tool named WinSCP. For more information, see [Transfer files to your Linux instance using WinSCP](#).

To use PSCP, you need the private key you generated in [Convert your private key using PuTTYgen](#). You also need the public DNS name of your Linux instance, or the IPv6 address if your instance has one.

Open PuTTY:

In the Category pan select session session and enter host name or IP address to which you want to connect. Make sure the port number be 22.

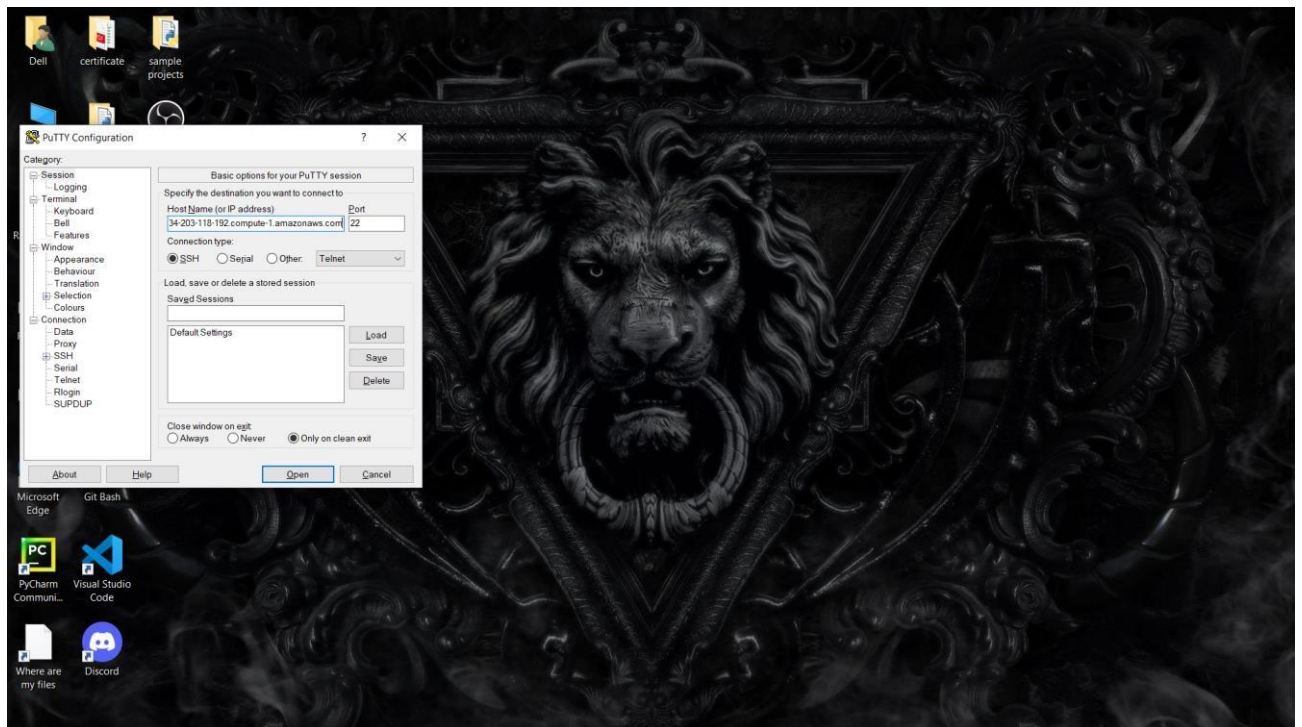


Fig 4.18 : Connecting using IPV4 Address

After That Expand connection and Expand SSH and select Auth and browse the key pair (.ppk) file from your PC and click open.

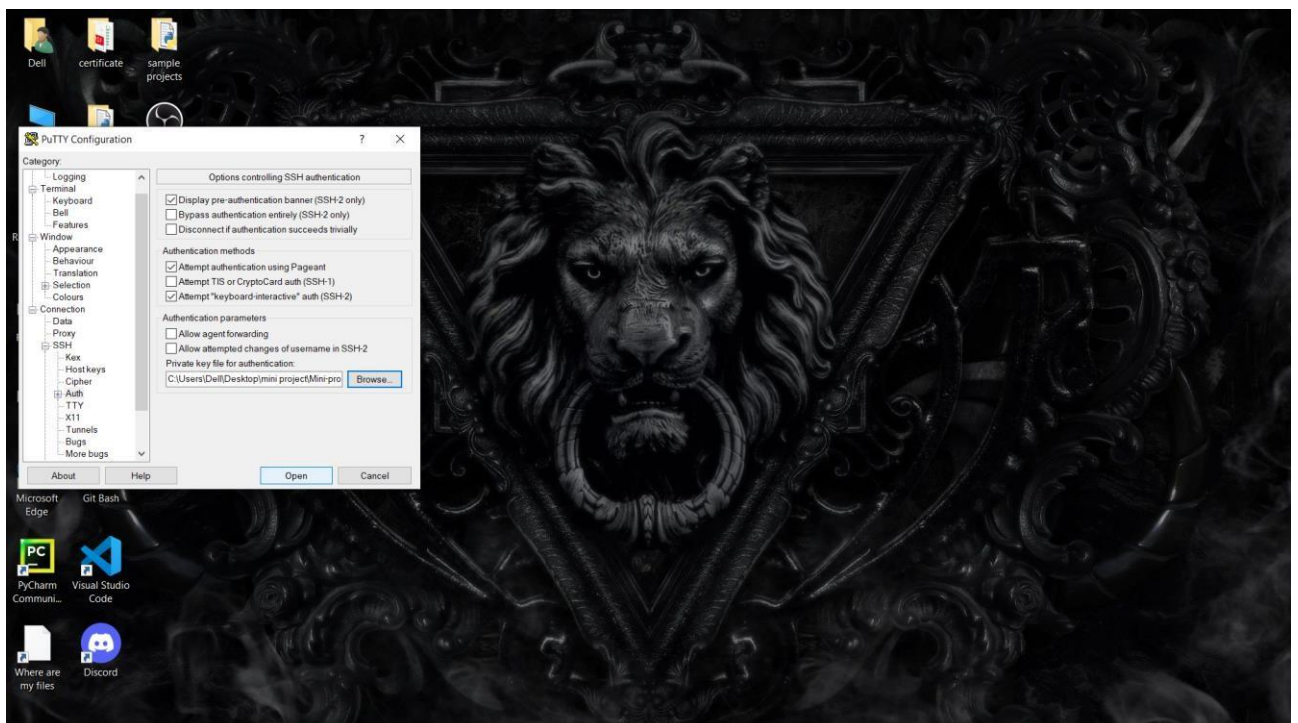


Fig 4.19 : Opening our Private Key

After Opening clone the project into virtual machine using the following command

```
git clone https://github.com/PALLI-VAMSI-KRISHNA/miniproject.git
```

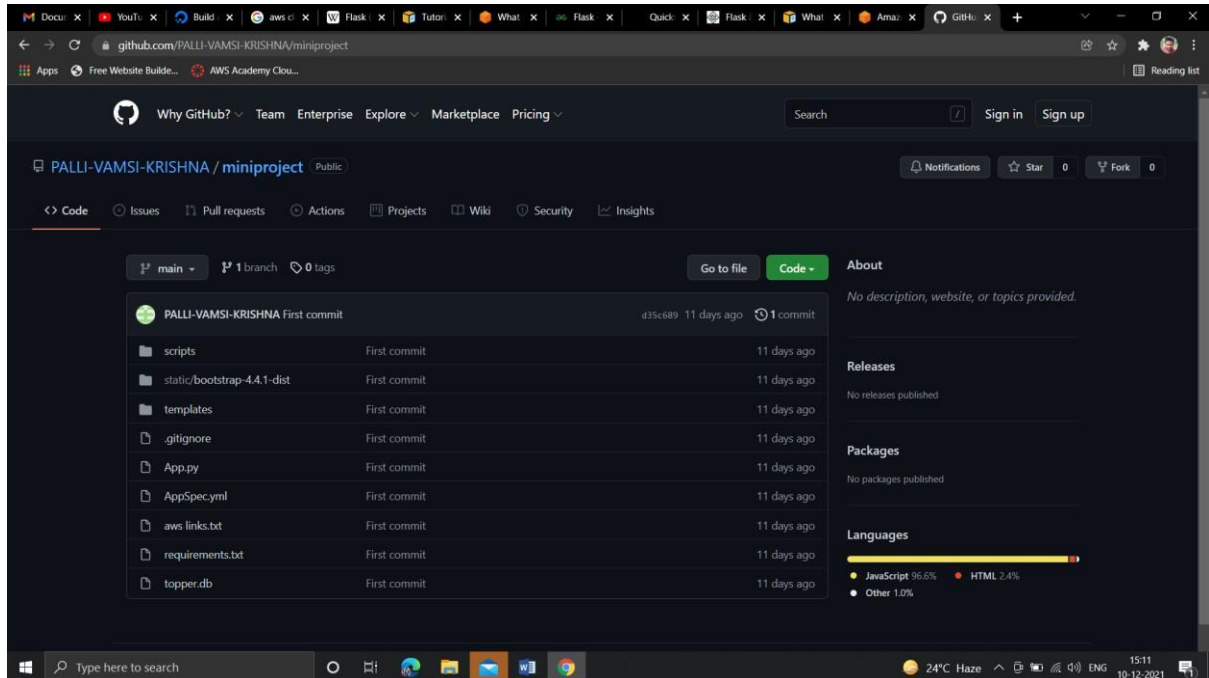


Fig 4.20 : Cloning git

Now go to project folder and run the application using following commands

```
export FLASK_APP=App.py flask  
run -h 0.0.0.0 -p 8000
```

```
ubuntu@ip-10-0-0-142: ~/miniproject
ubuntu@ip-10-0-0-142:~$ cd miniproject
ubuntu@ip-10-0-0-142:~/miniproject$ export FLASK_APP=App.py
ubuntu@ip-10-0-0-142:~/miniproject$ flask run -h 0.0.0.0 -p 8000
* Serving Flask app "App.py"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
```

Fig 4.21 : Running Flask Application

Now our flask application is running.

Copy the Public IPV4 address of our ec2 instance

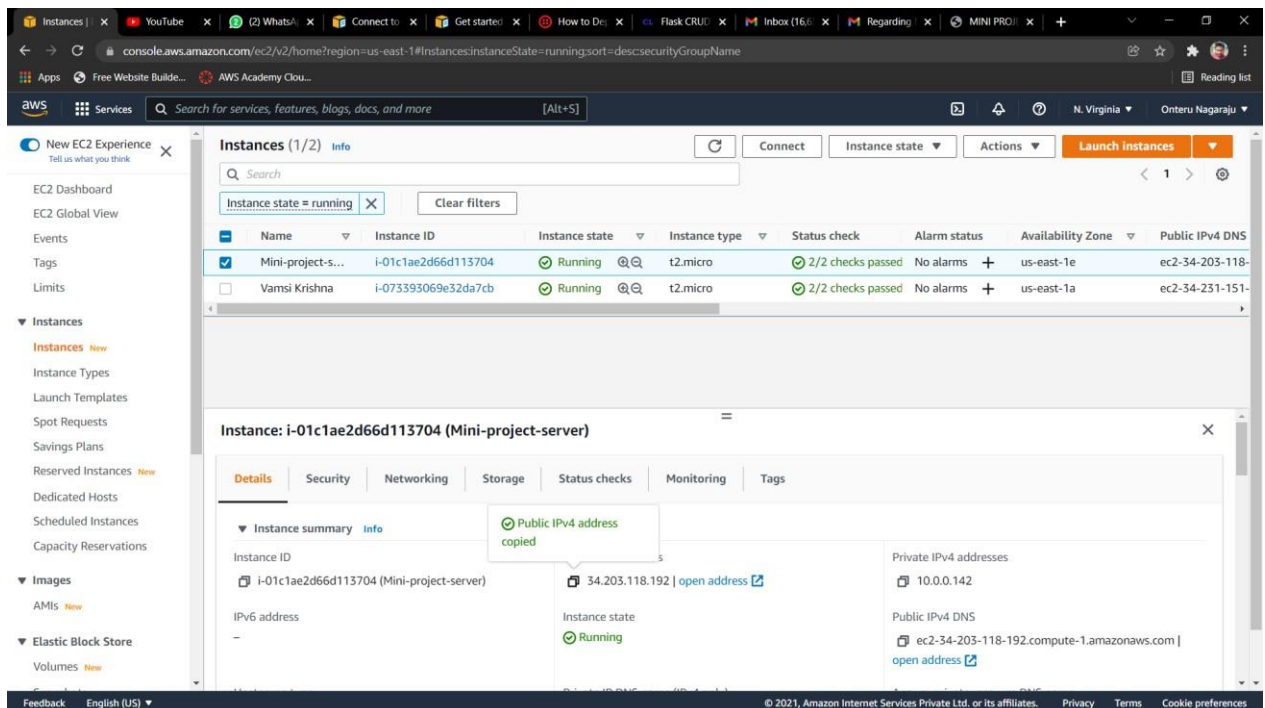


Fig 4.22 : Choose EC2 Instance

Type the IPV4 address in address bar along with port number and enter Now our website is ready.

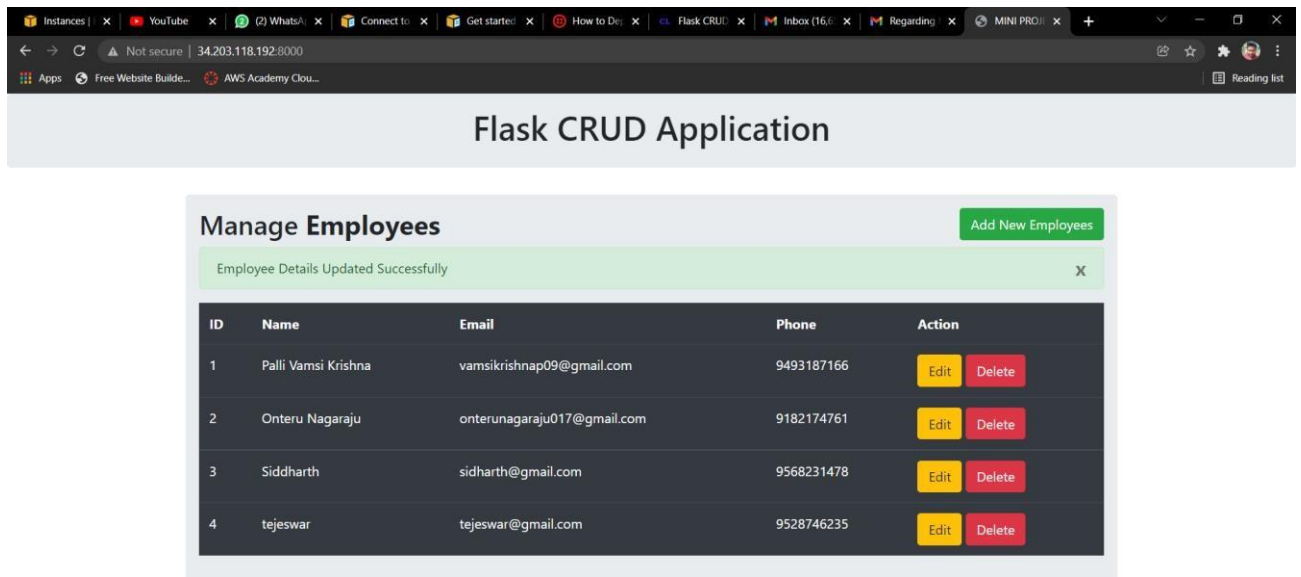


Fig 4.23 : Running website Globally

CHAPTER FIVE

ADVANTAGES and DISADVANTAGES

5.1 Advantages

- Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers
- With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure.
- You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments.

5.2 Disadvantages

- Importing the project files from windows to Linux ec2 instance using putty SSH client is a complicated task.
- General CRUD Database Connectivity is done through XAMPP Server and MYSQL.
- Manually installing commands every time we launch a new instance in the Linux terminal is a tiresome process.

5.3 Overcoming the drawbacks

- We can create a git hub repository to push all our project files and later use git hub cloning later in the Linux terminal.
- SQL Lite can be used instead of using xampp and mysql.
- We can create a requirents.txt that captures the software functional requirements and can be used later for newly created instances.

CHAPTER SIX

RESULTS AND CONCLUSION

- The flask application is designed and tested against the crud operations.
- The database connectivity has been given to the flask application.
- The Virtual Private Cloud of an EC2 Instance has successfully been set up.
- The flask application has been launched successfully through PUTTY SSH Client.
- We can now perform create, read, update, delete operations on the employee data.
- The database connectivity has been given to the flask application.
- The Virtual Private Cloud of an EC2 Instance has successfully been set up.
- The flask application has been launched successfully through PUTTY SSH Client.
- We can now perform create, read, update, delete operations on the employee data.
- We can create your own address
- We can create as many subnets as required irrespective of public or private
- We can restrict access to the users
- We can launch any number of instances and attain security to our web application

CHAPTER SEVEN

REFERENCES

<https://flask.palletsprojects.com/en/2.0.x/>

Cloud Services – Amazon Web Services

<https://aws.amazon.com>

Stack Overflow – Where Developers Learn, Share, & Build...

<https://stackoverflow.com>
