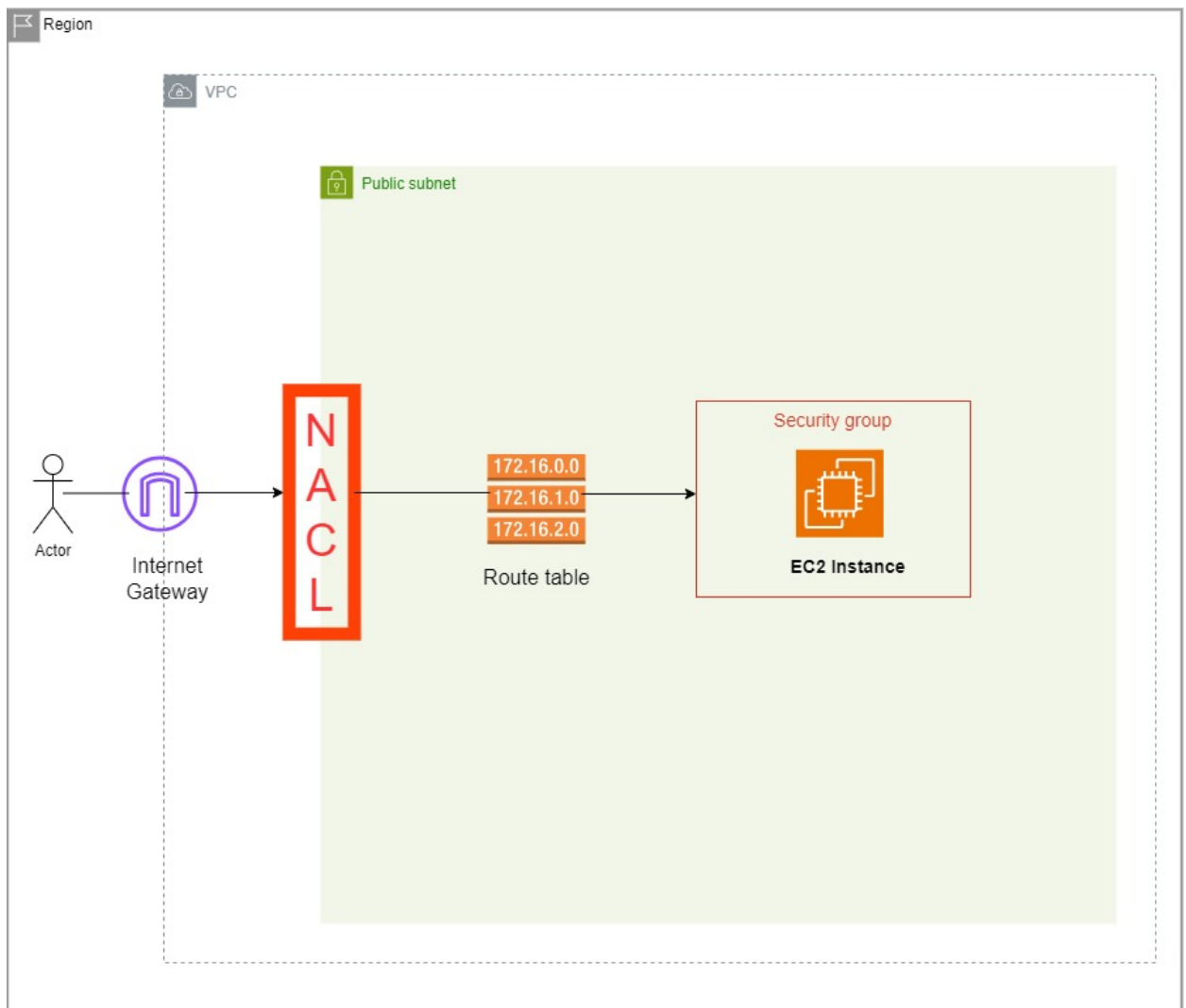


TECH GUARDIANS PROJECT

First, to start this project, we split ourselves into different teams so as to get everyone busy. These groups are:

- Architecture Design
- Creating of EC2 and VPC
- Python Deployment
- Write-up

The architecture design team came up with a road map for us and this became what we built on with respect to the login page project which was chosen by a majority in the group.



SECTION A

Detailed Explanation of Project Diagram and Process workflow

Explaining this diagram, the set-up is within an **AWS region** which is chosen and inside a **Virtual Private Network (VPC)**. The VPC helps to provide a network space that's isolated with a defined IP address range and subnets for the resources used.

A **Public subnet** which is within the VPC means that it has a direct access to the internet and other users can connect to connect to the resources in the subnet. The **Internet gateway** is a VPC component that allows communication access to the internet. In the diagram above, it allowed instances within the public subnet to communicate with the internet and vice versa.

The **Network Access Control List (NACL)** in this diagram helps to control inbound and outbound traffic based on rules set and NACL is a stateless, subnet-level security layer that controls inbound and outbound traffic based on rules. It acts as a firewall for the entire subnet, controlling access to and from the subnet based on IP addresses, ports, and protocols.

SECTION B

To set-up the EC2 and VPC

Login to the AWS console, go to VPC on the search bar and create a VPC. When creating the VPC, we created the resources by clicking VPC and more, we gave it a name "GameApp-VPC" input the IPv4 CIDR block of 10.0.0.0/16, with tenancy on default, availability zones and a public subnet is chosen as 1 according to the diagram and then click create VPC.

Note: The internet gateway automatically creates when public subnet is created.

After creating the VPC we proceed to creating an EC2 instance. To do this, in the console search bar, go to "EC2" and launch an instance. We renamed the instance "GameApp-server", chose the ubuntu AMI, used a t2.micro instance type, created a key pair, under network settings select your VPC and public subnet you created. We made sure to select the auto-assign public IP option which was not automatically enabled. We created a security group which would help to control

access into the instance we. For the storage, we chose a storage of 10 GiB gp3 so as to contain the resources that would be downloaded in the server after which we created the instance.

SECTION C

Connect the instance to the CLI terminal

To ssh into the terminal, we copy the public DNS of the instance "HERE" ec2-user@ec2-3-133-106-80.us-east-2.compute.amazonaws.com to the instance and it connects.

Update Virtual Server, Install required dependencies.



```
ubuntu@172-15-45-15:~$ ssh -i cloud-jay-key.pem ubuntu@34.238.156.120
ubuntu@34.238.156.120:~$ cd Downloads/
ubuntu@34.238.156.120:~/Downloads$ ssh -i cloud-jay-key.pem ubuntu@34.238.156.120
The authenticity of host '34.238.156.120 (34.238.156.120)' can't be established.
ED25519 key fingerprint is SHA256:4v4nIPfbcyN3K4ivIiifadwv/8ndqAmw1PjEtCaaZ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.238.156.120' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 = Documentation:  https://help.ubuntu.com
 = Management:    https://landscape.canonical.com
 = Support:       https://ubuntu.com/prn

System information as of Wed Sep 25 16:08:12 UTC 2024

System load:  0.48      Processes:    109
```

```
sudo apt update
```

```
sudo apt-get update -y
```

```
sudo apt-get upgrade -y
```

```
sudo apt-get install python3 python3-pip python3-dev build-essential libssl-dev
libffi-dev python3-setuptools -y
```

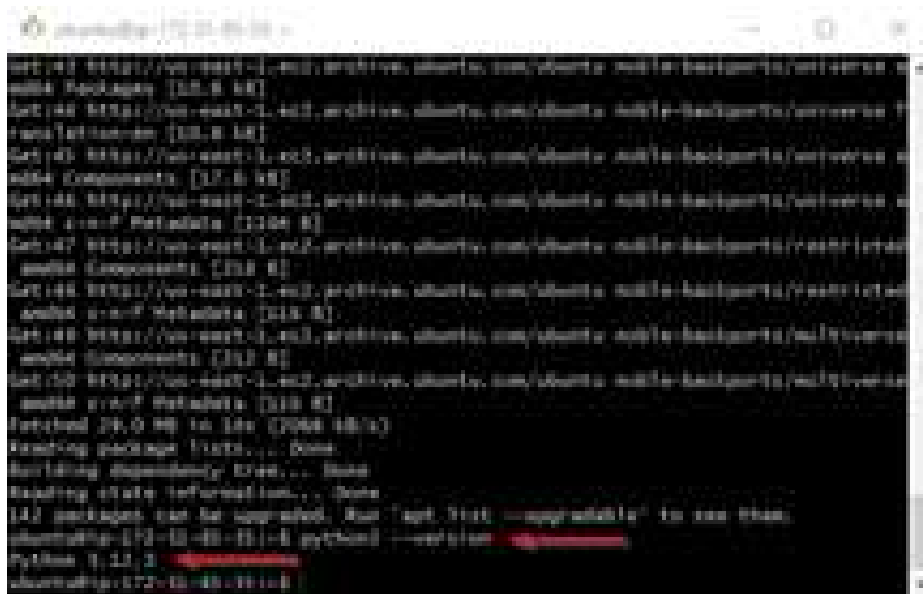
```
sudo apt-get install python3-venv -y
```

```
sudo apt-get install nginx -y
```

```
sudo systemctl start nginx
```

```
sudo systemctl status nginx
```

Step2: Verify Python Installation to Check Version, install other dependencies

A screenshot of a terminal window with a dark background. The terminal shows the output of several commands. The first command is 'python3 --version', which outputs 'Python 3.12.3'. The second command is 'pip install wheel flask', which outputs 'Requirement already satisfied: wheel in /usr/local/lib/python3.12/dist-packages (0.43.0)' and 'Requirement already satisfied: flask in /usr/local/lib/python3.12/dist-packages (3.0.3)'. The terminal also shows the output of 'pip install flask-cors', which outputs 'Requirement already satisfied: flask-cors in /usr/local/lib/python3.12/dist-packages (5.0.1)'. The terminal window has a title bar at the top with the text 'Terminal' and some icons on the left. The background of the terminal window shows a grid pattern.

```
mkdir GameApp
```

```
cd GameApp/
```

```
python3 --version
```

```
python3 -m venv Tech-Guardians
```

```
source Tech-Guardians/bin/activate
```

```
pip install wheel
```

```
pip install flask
```

```
git clone https://github.com/Jahmeeu-Cloud/PyBoardGame.git
```

```
cd PyBoardGame/
```

```
pip install -r requirements.txt
```

```
ls
```

```
pip install flask-cors
```

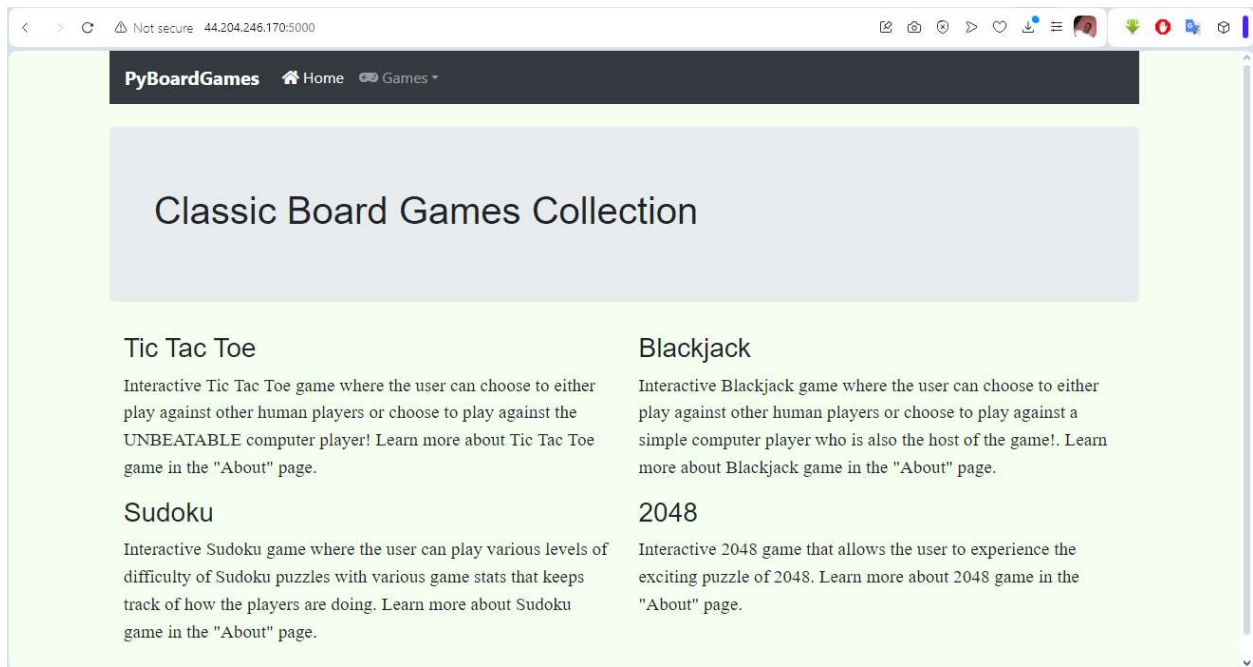
```
nano run.py app.run (host='0.0.0.0', port=5000)
```

```
ubuntu@ip-10-0-10-32: ~ - ssh
GNU nano 2.2.10 run.py
print('hello, I just wrote my first line of code')
^O
^X
```

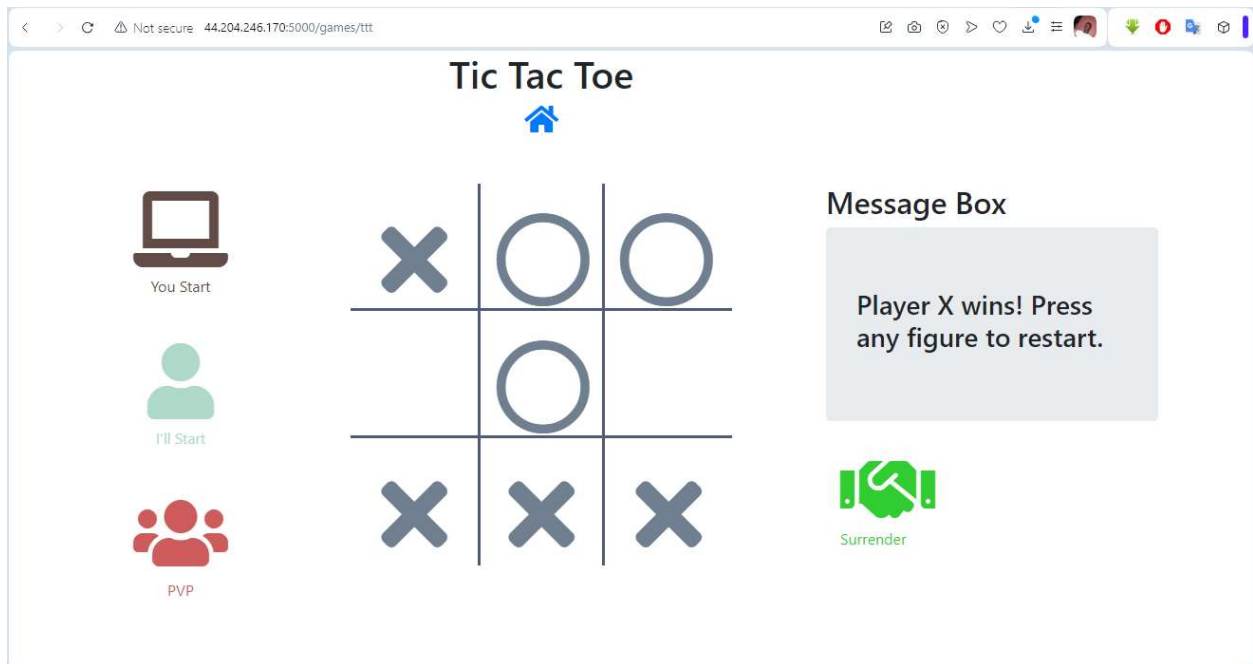
python3 run.py

```
ubuntu@ip-10-0-10-32: ~/GameApp/PyBoardGame
(Tech-Guardians) ubuntu@ip-10-0-10-32:~/GameApp/PyBoardGame$ nano run.py
(Tech-Guardians) ubuntu@ip-10-0-10-32:~/GameApp/PyBoardGame$ python3 run.py
* Serving Flask app 'board_games'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.10.32:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 290-993-747
102.215.57.99 - - [04/Nov/2024 13:39:34] "GET / HTTP/1.1" 200 -
102.215.57.99 - - [04/Nov/2024 13:39:34] "GET /static/css/auth.css HTTP/1.1" 200 -
102.215.57.99 - - [04/Nov/2024 13:39:52] "GET /static/images/favicon.ico HTTP/1.1" 200 -
102.215.57.99 - - [04/Nov/2024 13:42:41] "GET /games/ttt HTTP/1.1" 200 -
102.215.57.99 - - [04/Nov/2024 13:42:41] "GET /static/css/ttt.css HTTP/1.1" 200 -
102.215.57.99 - - [04/Nov/2024 13:42:41] "GET /static/js/ttt.js HTTP/1.1" 200 -
102.215.57.99 - - [04/Nov/2024 13:42:44] "GET /games/ttt/setup?human=X&pvp=false HTTP/1.1" 200 -
```

Congratulations! 🎉 You've Successfully Deployed Your First Flask Application in Python



This is the result of the deployment Python Flask Application.



You can imagine I win computer (I am player X).

44.204.246.170:5000/games/sudoku

1

2

3

4

5

6

7

8

9

<