

# Configuration Manual

MSc Research Project  
Cloud Computing

**Srija Perugu**  
Student ID: X21168105

School of Computing  
National College of Ireland

Supervisor: Vikas Sahni

**National College of Ireland**  
**MSc Project Submission Sheet**



**School of Computing**

**Student Name:** Srija Perugu

**Student ID:** X21168105

**Programme:** Cloud Computing **Year:** 2022

**Module:** Msc Research Project

**Lecturer:** Vikas Sahni

**Submission Due Date:** 15-12-2022

**Project Title:** A novel model for data storage using LZW compression technique for Cloud based Electronic Healthcare Systems

**Word Count:** 1404 **Page Count:** 12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** P.Srija

**Date:** 15-12-2022

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

|                                                                                                                                                                                           |                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| Attach a completed copy of this sheet to each project (including multiple copies)                                                                                                         | <input type="checkbox"/> |
| <b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).                                                                  | <input type="checkbox"/> |
| <b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | <input type="checkbox"/> |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

|                                  |  |
|----------------------------------|--|
| <b>Office Use Only</b>           |  |
| Signature:                       |  |
| Date:                            |  |
| Penalty Applied (if applicable): |  |

# Configuration Manual

Srija Perugu  
Student ID: X21168105

## 1 Introduction

This document can be used as a configuration manual for reference purpose, while attempting to replicate this work as it details the software and hardware setup used to execute the codes from the initial data collection stage all the way through to the final implementation.

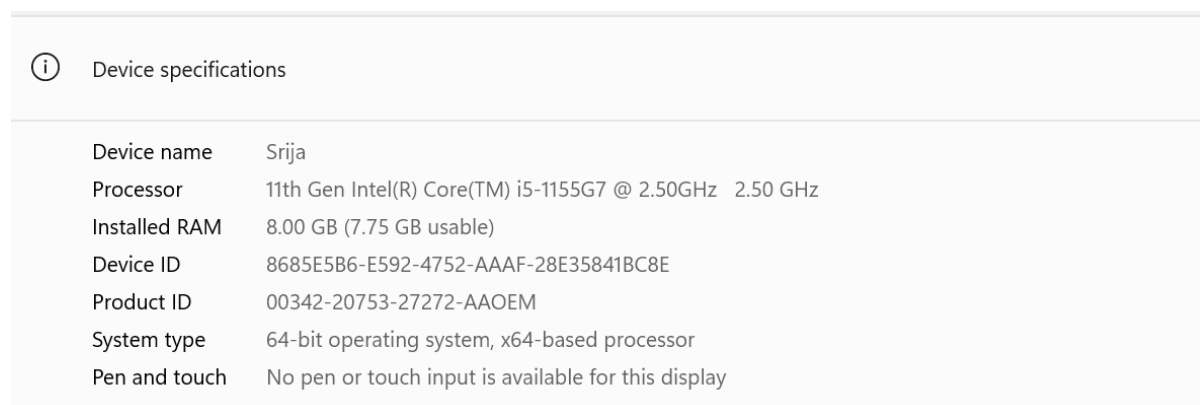
**Research Work:** A novel model for data storage using LZW compression technique for Cloud based Electronic Healthcare Systems. The main aim of this work is data storage using LZW data compression technique and storing it inside IPFS server and also providing security utilising blockchain technology.

## 2 System Configuration

As the data is large, the below requirements need to be met for a smooth as well as time efficient approach.

### 2.1 Hardware Configuration

All code are executed within the parameters of the system hardware listed below.



| Device specifications |                                                         |
|-----------------------|---------------------------------------------------------|
| Device name           | Srija                                                   |
| Processor             | 11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz 2.50 GHz |
| Installed RAM         | 8.00 GB (7.75 GB usable)                                |
| Device ID             | 8685E5B6-E592-4752-AAAF-28E35841BC8E                    |
| Product ID            | 00342-20753-27272-AAOEM                                 |
| System type           | 64-bit operating system, x64-based processor            |
| Pen and touch         | No pen or touch input is available for this display     |

Figure 1: Hardware Requirements

## 2.2 Software Configuration

Listed below are the various software and their respective versions used.

Table 1: Software Requirements

| Software          | Version |
|-------------------|---------|
| Visual Studio(VS) | 2019    |
| Python            | 3.7.0   |
| Numpy             | 1.18.5  |
| JDK               | 8       |
| Go Ethereum(geth) | 1.8.22  |
| Ipfs-api          | 0.2.3   |

## 3 System Configuration

### 3.1 Ethereum Wallet

An ethereum wallet needs to be created. To manage wallet files, WalletUtils is used, which provides a few useful Utility operations. The wallet file's storage location must be specified, and the password and file path must be provided to the below method in order to gain access to the mnemonic phrases and the UTC Json file used to store the user's credentials in an encrypted manner. Tokens as well as smart contracts are most commonly found on Ethereum, which is also the most popular platform overall (Di Angelo, 2020). The account's private and public keys (credentials) are stored in Wallet.

```
web3j = Web3j.build(new HttpService());  
credentials = WalletUtils.loadCredentials("erum", "C:/ETH/data-private/keystore/UTC--2020-07-02
```

Fig 2: Ethereum Wallet

### 3.2 Smart Contract

Smart contracts have been embedded into the popular blockchain-based development environments, such as Ethereum as well as Hyperledger, and have a wide range of potential application areas in the digital economy as well as intelligent industries, such as financial services, healthcare, the Internet of Things and management (Wang, 2019).A smart Contract will be loaded using the below code snippet:

```
SmartContract sc = SmartContract.load(address, web3j, credentials, ManagedTransaction.GAS_PRICE, C  
String access = sc.getAccessAccount().send();
```

Fig 3: Smart Contract

### 3.3 Importing Libraries

The below mentioned libraries are used for storage, data compression, performing mathematical calculations and socket functionality.

```
1 import matplotlib.pyplot as plt
2 import os
3 from flask import Flask, render_template, request, redirect, Response
4 import ipfsApi
5 import socket
6 import json
7 import zlib
8 import sys
9 import numpy as np
a
```

Fig 4: Importing Libraries

### 3.4 Data Compression

Data compression is performed utilising LZW technique and then the data is encrypted.

```
data = str(pid)+" "+name+" "+date+" "+address+" "+phone+" "+condition
original_size = sys.getsizeof(data)
compressed = zlib.compress(data.encode())
compress_size = sys.getsizeof(compressed)
f = open(str(pid)+".txt", "w")
f.write(data)
f.close()
```

Fig 5: Data Compression

### 3.5 Data Encryption and Storage to IPFS

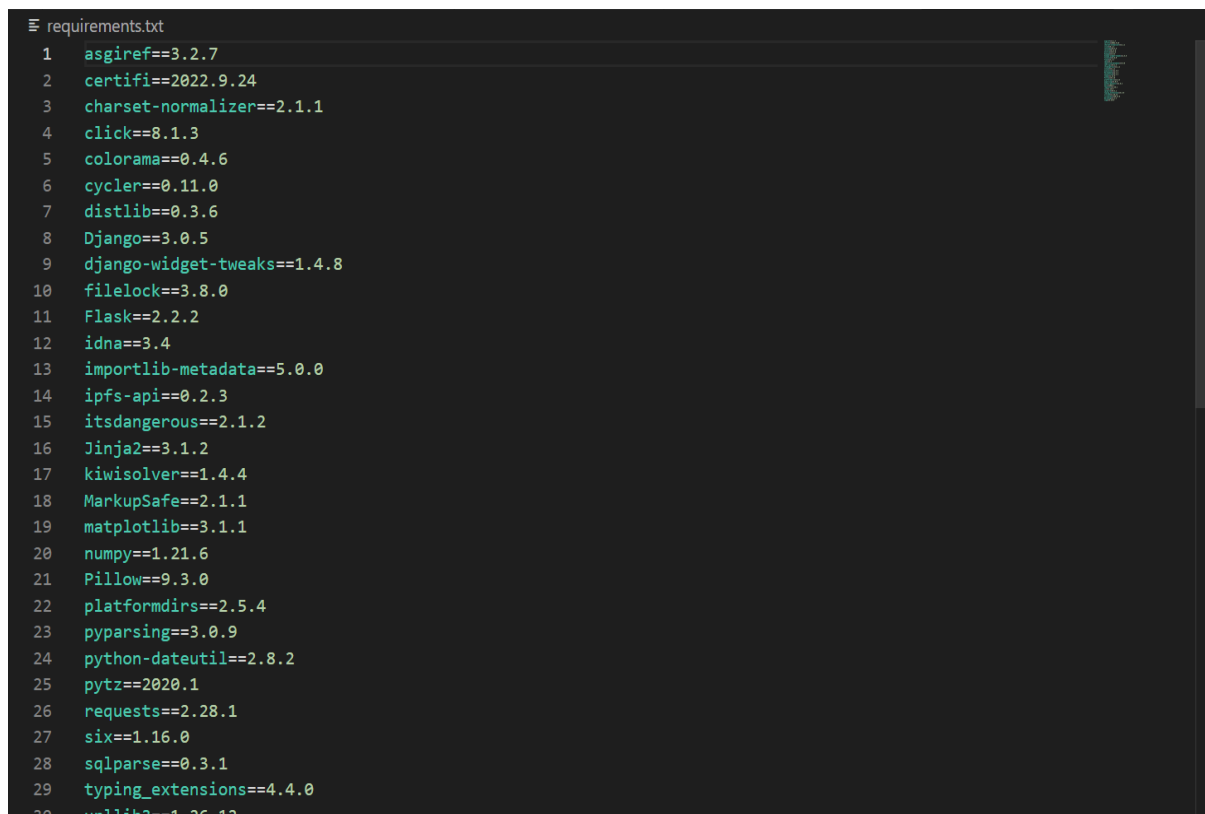
The compressed data is encrypted using encryption algorithm and stored to IPFS Server.

```
new_file = api.add(str(pid)+".txt") #adding encrypted model to IPFS
hashcode = new_file['Hash']
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('localhost', 3333))
hashdata = 'createrecord,'+str(pid)+" "+hashcode
```

Fig 6: Data Encryption and Storage

### 3.6 Requirements File

A requirements.txt file is a common file format in Python that contains a list of all the dependencies for a given project, such as libraries, modules, and packages. It also keeps any packages or additional files the project needs to function. This "requirements.txt" document is typically found in the main project folder. The command `pip freeze` is executed, which records the current package list of an environment to the file requirements.txt.



```
requirements.txt
1  asgiref==3.2.7
2  certifi==2022.9.24
3  charset-normalizer==2.1.1
4  click==8.1.3
5  colorama==0.4.6
6  cyclr==0.11.0
7  distlib==0.3.6
8  Django==3.0.5
9  django-widget-tweaks==1.4.8
10 filelock==3.8.0
11 Flask==2.2.2
12 idna==3.4
13 importlib-metadata==5.0.0
14 ipfs-api==0.2.3
15 itsdangerous==2.1.2
16 Jinja2==3.1.2
17 kiwisolver==1.4.4
18 MarkupSafe==2.1.1
19 matplotlib==3.1.1
20 numpy==1.21.6
21 Pillow==9.3.0
22 platformdirs==2.5.4
23 pyparsing==3.0.9
24 python-dateutil==2.8.2
25 pytz==2020.1
26 requests==2.28.1
27 six==1.16.0
28 sqlparse==0.3.1
29 typing_extensions==4.4.0
30 urllib3==1.26.12
```

Fig 7: Requirements.txt

## 4 Execution Steps

**Step-1:** Start the project by double-clicking the file named "start eth.bat." This will launch the Ethereum development tool, and once that's done, below screen will be appeared.

```

C:\windows\system32\cmd.exe
INFO [12-14 10:44:21.158]   block reached canonical chain      number=15498 hash=f4f45b_2f4438
INFO [12-14 10:44:21.243]   mined potential block      number=15497 hash=1188c9_2c08c3
INFO [12-14 10:44:21.331]   Commit new mining work     number=15498 sealhash=e9a9b5_c5cd05 uncles=0 txs=0 gas=0 fees=0 elapsed=180.852ms
INFO [12-14 10:44:52.844]   Successfully sealed new block number=15498 sealhash=e9a9b5_c5cd05 hash=b6b190_d1caf3 elapsed=31.680s
INFO [12-14 10:44:52.973]   block reached canonical chain number=15491 hash=eb5e7a_da8240
INFO [12-14 10:44:53.051]   mined potential block      number=15498 hash=b6b190_d1caf3
INFO [12-14 10:44:53.148]   Commit new mining work     number=15499 sealhash=91b39d_f36548 uncles=0 txs=0 gas=0 fees=0 elapsed=175.331ms
INFO [12-14 10:44:54.209]   Successfully sealed new block number=15499 sealhash=91b39d_f36548 hash=0d148b_618129 elapsed=1.236s
INFO [12-14 10:44:54.334]   block reached canonical chain number=15492 hash=0bc7d_3f8a2c
INFO [12-14 10:44:54.445]   mined potential block      number=15499 hash=0d148b_618129
INFO [12-14 10:44:54.532]   Commit new mining work     number=15500 sealhash=7724c2_d9926a uncles=0 txs=0 gas=0 fees=0 elapsed=198.007ms
INFO [12-14 10:45:05.008]   Successfully sealed new block number=15500 sealhash=7724c2_d9926a hash=1743fc_b4cc34 elapsed=10.673s
INFO [12-14 10:45:05.132]   block reached canonical chain number=15493 hash=c608d9_9d377a
INFO [12-14 10:45:05.232]   mined potential block      number=15500 hash=1743fc_b4cc34
INFO [12-14 10:45:05.313]   Commit new mining work     number=15501 sealhash=a6e435_e73638 uncles=0 txs=0 gas=0 fees=0 elapsed=177.874ms
INFO [12-14 10:45:06.667]   Successfully sealed new block number=15501 sealhash=a6e435_e73638 hash=9e3042_dff698 elapsed=1.532s
INFO [12-14 10:45:06.896]   block reached canonical chain number=15494 hash=9e3042_dff698
INFO [12-14 10:45:06.986]   mined potential block      number=15501 sealhash=9e3042_dff698
INFO [12-14 10:45:06.986]   Commit new mining work     number=15502 sealhash=adf469_8cd853 uncles=0 txs=0 gas=0 fees=0 elapsed=174.899ms
INFO [12-14 10:45:19.757]   Successfully sealed new block number=15502 sealhash=adf469_8cd853 hash=25a683_da8d62 elapsed=12.946s
INFO [12-14 10:45:19.917]   block reached canonical chain number=15495 hash=af320b_570f9f
INFO [12-14 10:45:20.011]   mined potential block      number=15502 sealhash=25a683_da8d62
INFO [12-14 10:45:20.112]   Commit new mining work     number=15503 sealhash=a884a1_7bcd00 uncles=0 txs=0 gas=0 fees=0 elapsed=195.436ms
INFO [12-14 10:45:22.655]   Successfully sealed new block number=15503 sealhash=a884a1_7bcd00 hash=cfb0de_dd1b2e elapsed=2.738s
INFO [12-14 10:45:22.784]   block reached canonical chain number=15496 hash=af320b_570f9f
INFO [12-14 10:45:22.880]   mined potential block      number=15503 sealhash=cfb0de_dd1b2e
INFO [12-14 10:45:22.952]   Commit new mining work     number=15504 sealhash=921daf_0a6825 uncles=0 txs=0 gas=0 fees=0 elapsed=168.110ms
INFO [12-14 10:45:23.598]   Successfully sealed new block number=15504 sealhash=921daf_0a6825 hash=cece67_2bc5c elapsed=814.416ms
INFO [12-14 10:45:23.713]   block reached canonical chain number=15497 hash=eb5e7a_da8240
INFO [12-14 10:45:23.854]   mined potential block      number=15504 sealhash=cece67_2bc5c
INFO [12-14 10:45:23.934]   Commit new mining work     number=15505 sealhash=9d39d6_13c51e uncles=0 txs=0 gas=0 fees=0 elapsed=193.018ms
INFO [12-14 10:45:28.355]   Successfully sealed new block number=15505 sealhash=9d39d6_13c51e hash=ca641d_c13f0e elapsed=4.514s
INFO [12-14 10:45:28.477]   block reached canonical chain number=15499 hash=0d148b_618129
INFO [12-14 10:45:28.577]   mined potential block      number=15505 sealhash=ca641d_c13f0e
INFO [12-14 10:45:28.673]   Commit new mining work     number=15506 sealhash=9315e8_8dfc5b uncles=0 txs=0 gas=0 fees=0 elapsed=196.613ms
INFO [12-14 10:45:51.343]   Successfully sealed new block number=15506 sealhash=9315e8_8dfc5b hash=be4938_8ba248 elapsed=22.866s
INFO [12-14 10:45:51.472]   block reached canonical chain number=15499 hash=0d148b_618129
INFO [12-14 10:45:51.577]   mined potential block      number=15506 sealhash=be4938_8ba248
INFO [12-14 10:45:51.678]   Commit new mining work     number=15507 sealhash=c45fdc_75b76f uncles=0 txs=0 gas=0 fees=0 elapsed=205.577ms

```

Fig 8: Ethereum Tool

**Step-2:** To deploy a smart contract to the Ethereum tool, run the 'initialize eth.bat' file and proceed when you see the message "Smart Contract Ready to store data".

```

C:\windows\system32\cmd.exe
C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>javac -cp ";lib/*" -d . *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>java -cp ";lib/*" -Xmx1000M com.deploy
Transaction complete : 0x59594f9bec43eaf0fc15cfb602ed5cb848ab840500cac897d14c0e13c50c4cf
Deploying smart contract
Smart contract deployed to address 0x3bc32cd84c22b094424711d8029f240d898736e4
Initial value of counter in Smart contract: temp,temp,temp
Smart Contract Ready to store data

```

Fig 9: Smart Contract

**Step-3:** When the above screen displays an error, wait a few minutes and repeat the process again. When that message appears, continue to the next procedure. Running the 'Start IPFS.bat' file will launch the IPFS server, and the subsequent screen will look like the one shown below.

```

C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>ipfs init
initializing ipfs node at C:\Users\35389\AppData\Local\ipfs
error: ipfs configuration file already exists!
Reinitializing would overwrite your keys.

C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>ipfs daemon
Initializing daemon...
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/192.168.43.245/tcp/4001
Swarm listening on /ip6/2401:4900:328d:15f3:4c33:2125:58eb:d990/tcp/4001
Swarm listening on /ip6/2401:4900:328d:15f3:4c33:2125:58eb:d990/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmPRVvY17R3HqKQ3q1n34S5ABHYN0vSVVRpTCh0eFKEA
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/192.168.43.245/tcp/4001
Swarm announcing /ip6/2401:4900:328d:15f3:4c33:2125:58eb:d990/tcp/4001
Swarm announcing /ip6/2401:4900:328d:15f3:4c33:2125:58eb:d990/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
IPFS server listening on /ip4/127.0.0.1/tcp/8080
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080

Daemon is ready.
[0]37m00:59.607 - [31]ERROR - [0]34m dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
put record to routing error: failed to find any peer in table dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
[0]37m00:59.623 - [31]ERROR - [0]34mrepub: +[en]Republisher failed to republish: failed to find any peer in table - [0]37mrepub.go:66-[0]
[0]37m00:59.632 - [31]ERROR - [0]34m dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
put record to routing error: failed to find any peer in table dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
[0]37m00:59.645 - [31]ERROR - [0]34mrepub: +[en]Republisher failed to republish: failed to find any peer in table - [0]37mrepub.go:66-[0]
[0]37m00:59.659 - [31]ERROR - [0]34m dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
put record to routing error: failed to find any peer in table dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
[0]37m00:59.662 - [31]ERROR - [0]34mrepub: +[en]Republisher failed to republish: failed to find any peer in table - [0]37mrepub.go:66-[0]
[0]37m00:59.670 - [31]ERROR - [0]34m dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
put record to routing error: failed to find any peer in table dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]
[0]37m00:59.681 - [31]ERROR - [0]34m dht: +[en]loggableKey could not cast key; invalid cid version number: 47 - [0]37mlookup.go:35-[0]

```

Fig 10: Start IPFS Server

**Step-4:** After the IPFS server starts up in the previous screen, start 'run.bat' file to launch the python FLASK server, as shown in the subsequent screen.

```

c:\windows\system32\cmd.exe
C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>python SecureEHR.py
* Serving Flask app 'SecureEHR'
* 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 http://127.0.0.1:9999
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 205-228-628
127.0.0.1 - - [09/Dec/2022 13:10:39] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [09/Dec/2022 13:10:39] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [09/Dec/2022 13:10:49] "GET /index HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 13:10:49] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 13:10:49] "GET /static/images/img01.gif HTTP/1.1" 404 -
127.0.0.1 - - [09/Dec/2022 13:10:49] "GET /static/images/img03.jpg HTTP/1.1" 404 -
127.0.0.1 - - [09/Dec/2022 13:10:49] "GET /static/images/img02.gif HTTP/1.1" 404 -
127.0.0.1 - - [09/Dec/2022 13:10:53] "GET /Login HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 13:10:53] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [09/Dec/2022 13:10:56] "GET /Patients HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 13:10:56] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [09/Dec/2022 13:10:56] "GET /static/datetimpicker.js HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 13:10:56] "GET /cal.gif HTTP/1.1" 404 -
Record saved in Ethereum
127.0.0.1 - - [09/Dec/2022 13:12:08] "POST /PatientData HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 13:12:08] "GET /static/style.css HTTP/1.1" 304 -

```

Fig 11: Python Server Started

In above screen python server started and now open browser and enter URL as 'http://127.0.0.1:9999/index' and press enter key to index page. Once the necessary details such as name of patient, data of birth, address, phone number and health issue have been entered by the patient, then it displays the HASHCODE returned by IPFS and Blockchain. Also it shows that record is saved to ethereum, then data will be compressed and is represented with a graphical representation comparing the original data and compressed data in the UI. The data is then encrypted and after stored into IPFS server will get below output along with the address.

```

c:\windows\system32\cmd.exe
C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>java -cp "lib/*" -d . *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
C:\Users\35389\Desktop\NCI\Extension Project\SecureEHR>java -cp "lib/*" *.xxx1000M.com.deploy
Transaction complete : 0x33c32cd84c22b094424711d8029f24e898736e4
Deploying smart contract
Smart contract deployed to address 0x3bc32cd84c22b094424711d8029f24e898736e4
Initial value of counter in Smart contract: temp,temp,temp
Smart contract ready to store data
Transaction complete : 0xe4025e0165d04bc3a3e352c1c5d9b3b5458e8240c17e3935bcf89ec6d77eb81 0x3bc32cd84c22b094424711d8029f24e898736e4
Address 0x3bc32cd84c22b094424711d8029f24e898736e4
1.QmVKT5SaG5QkAZVvN49wHFP1rcH2TP2ntLk8m95TmwpF5m Record saved in Ethereum
Address 0x3bc32cd84c22b094424711d8029f24e898736e4
2.QmP4XT4kakeHUS7wDEYRpvJ6E7X16hs5AhgXmREVBKYNHE Record saved in Ethereum
1,2
3.QmP4XT4kakeHUS7wDEYRpvJ6E7X16hs5AhgXmREVBKYNHE
1,2
1.QmVKT5SaG5QkAZVvN49wHFP1rcH2TP2ntLk8m95TmwpF5m

```

Fig 12: Hashcode returned by IPFS



## 5 AWS EC2 Instance Creation for Deployment

Below are steps for deploying an application in AWS using EC2 instance

- Go to the AWS management console and click on the EC2 instance tab.
- Select the instance to run and click the "Launch" button.
- Choose the machine image for Ubuntu Server 18.04 LTS (HVM), SSD Volume Type.
- Choose a suitable type; for this example, I've gone with t2.micro. Choose the option to "Configure Instance Details" next.
- Determine the amount of storage space needed; in this case, I'll go with 15 GB. Follow this by clicking "Add Tags," followed by "Select Configure Security Group."
- Click the Review and Start button.
- It will be prompted to choose a key pair. You can generate a new key pair, name it, and save the resulting Key Pair file by clicking the corresponding button.
- Launch instances and check out the instance.

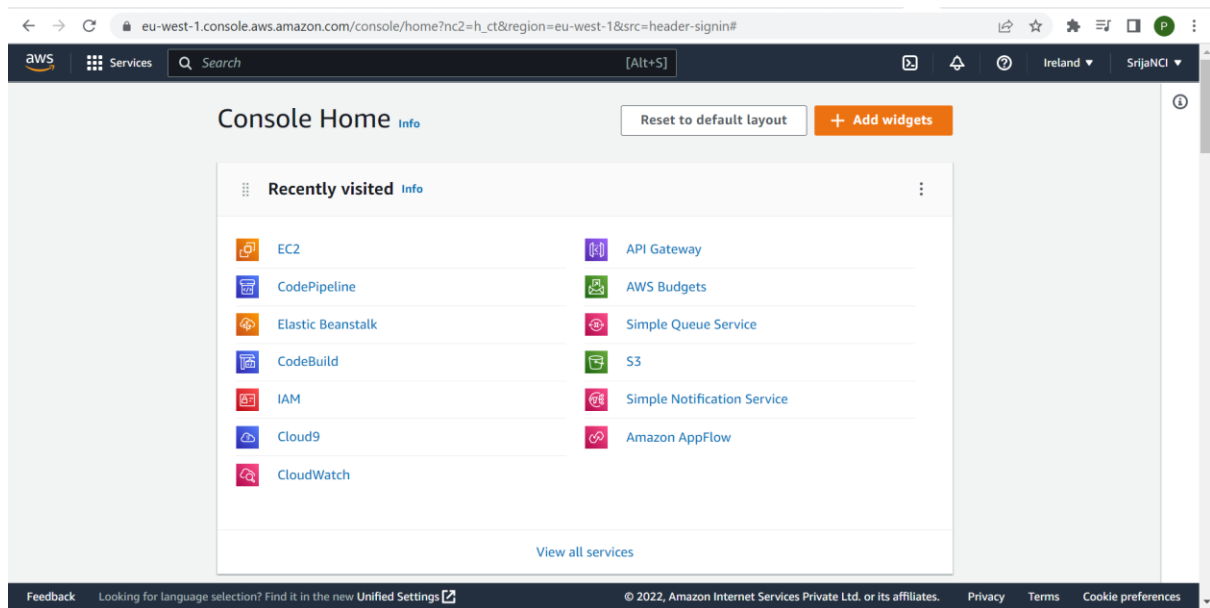


Fig 13: AWS management Console

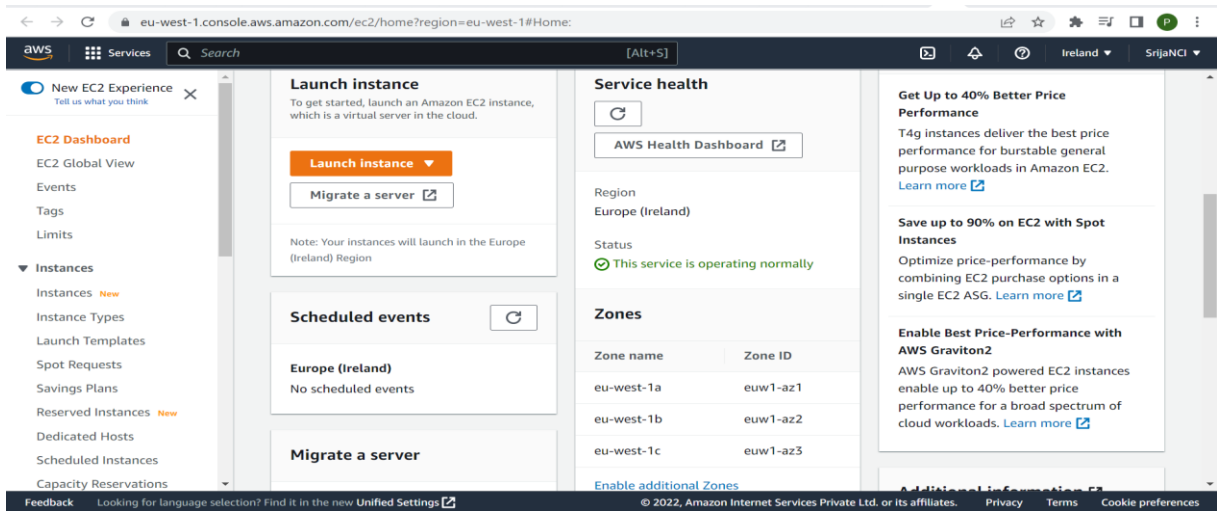


Fig 14: Launch EC2 instance

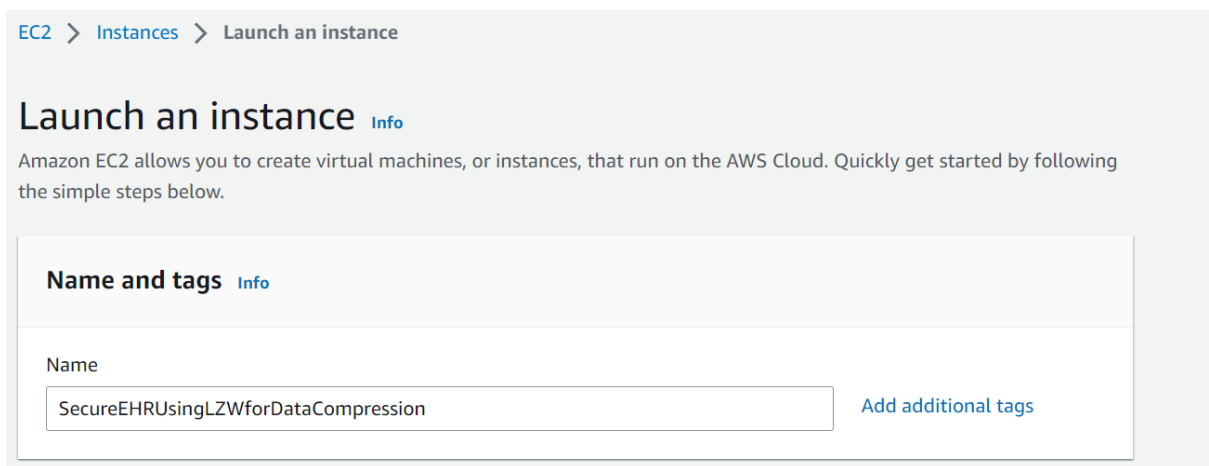


Fig 15: Name for instance

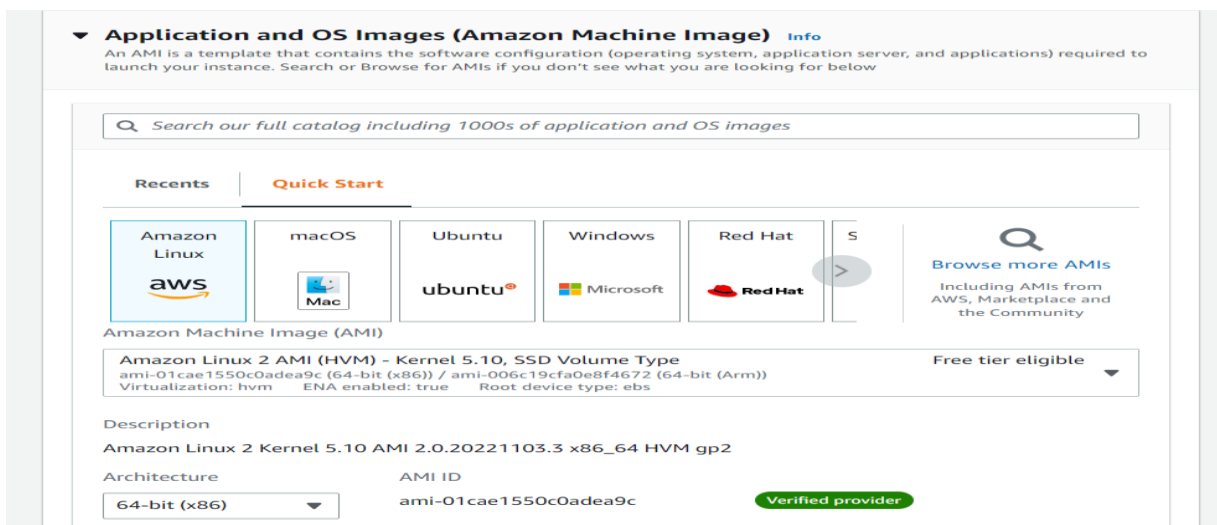


Fig 16: Amazon Linux AMI

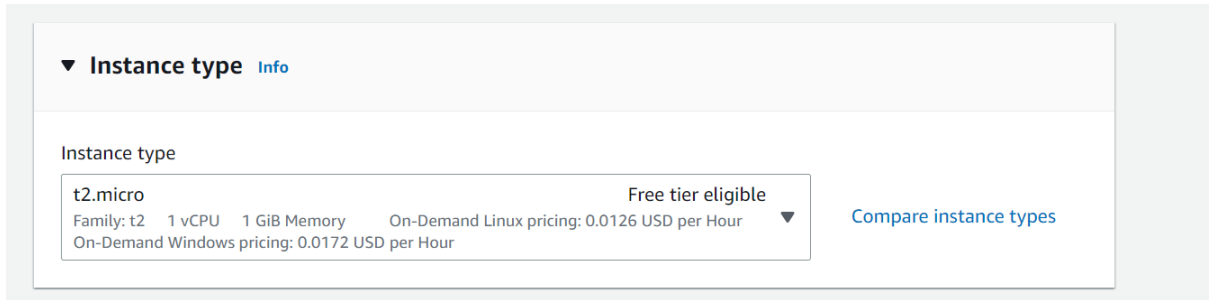


Fig 17: Instance Type

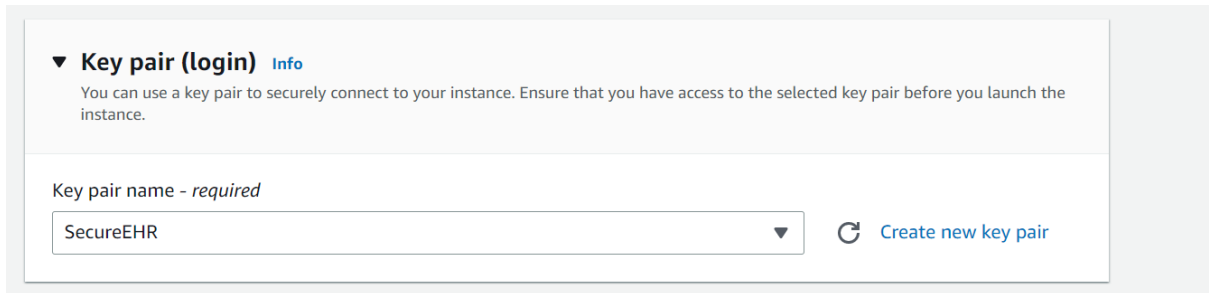


Fig 18: Key Pair Name

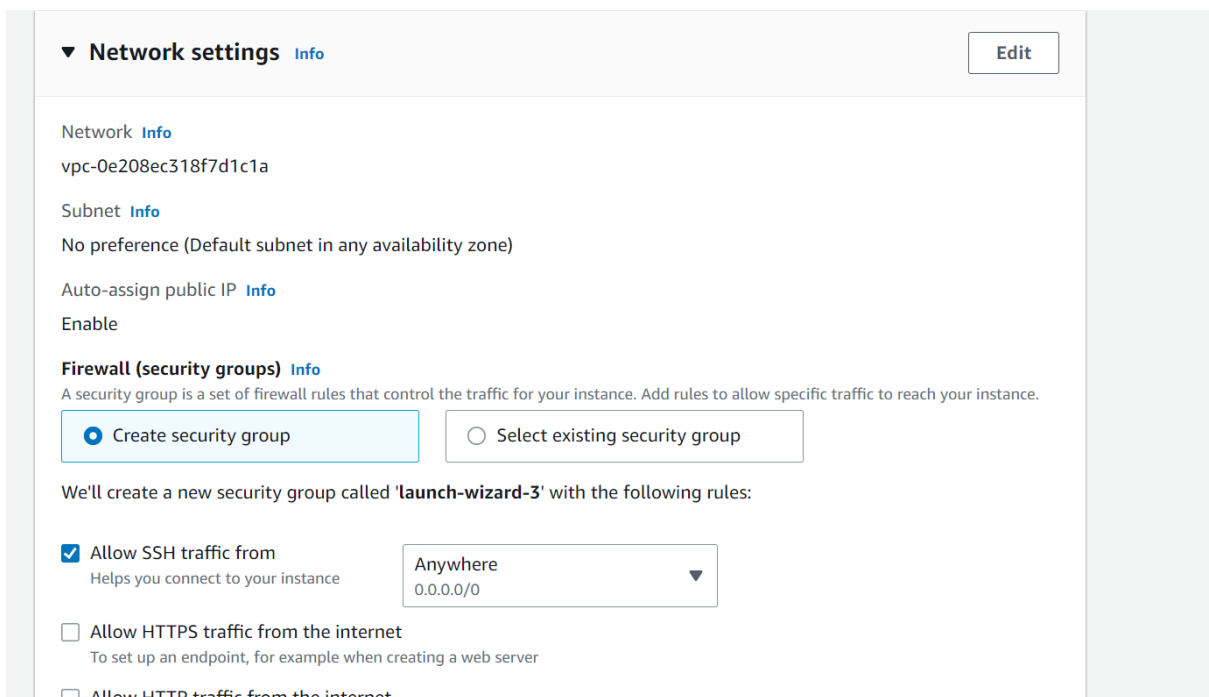


Fig 19: Network Settings

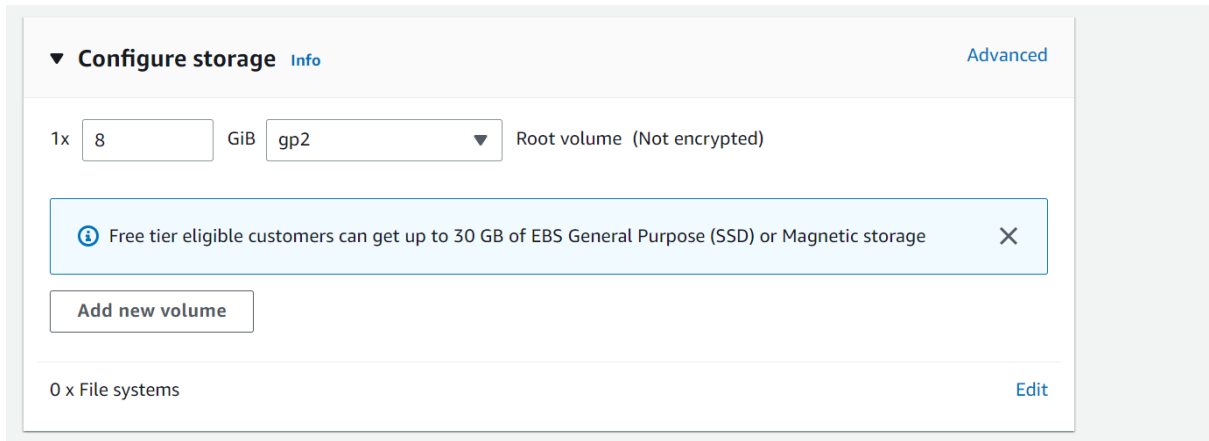


Fig 20: Configure Storage

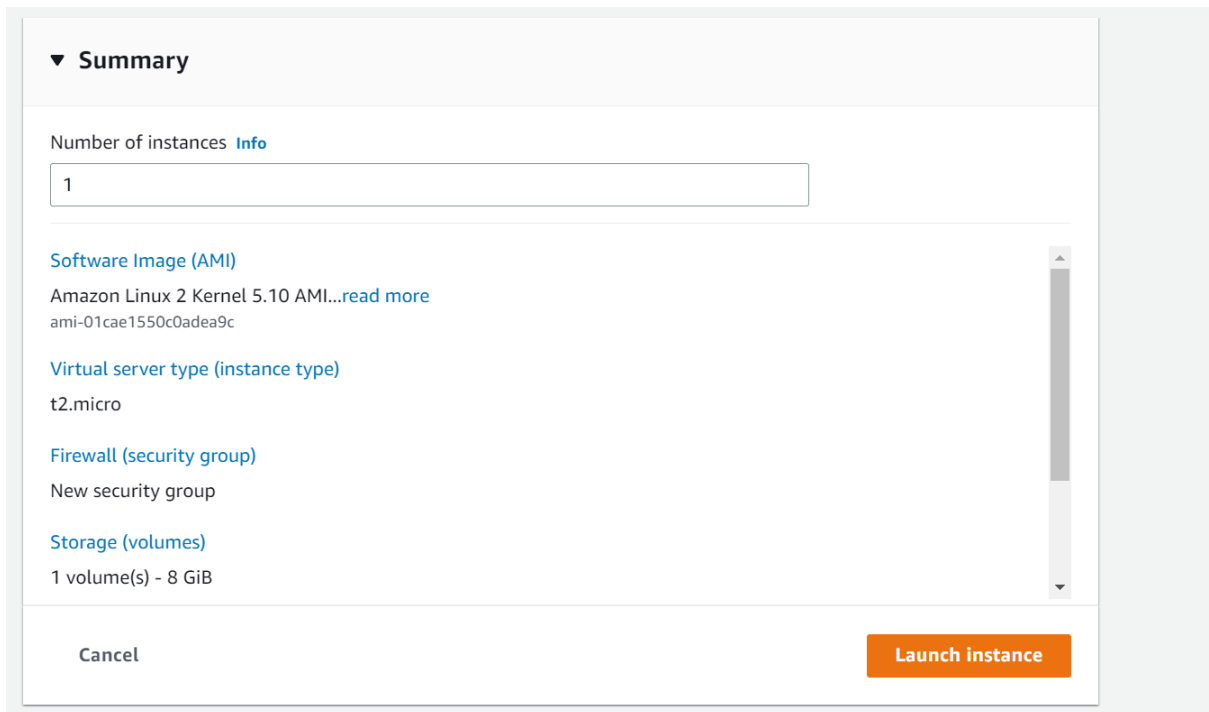


Fig 21: Summary

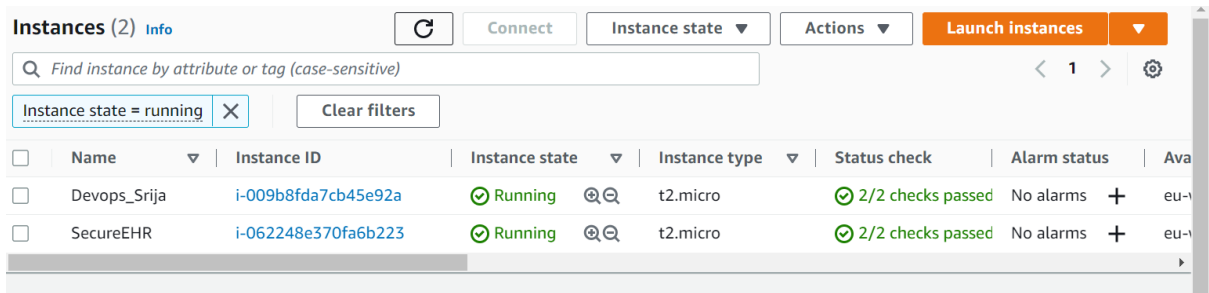


Fig 22: Instances Running

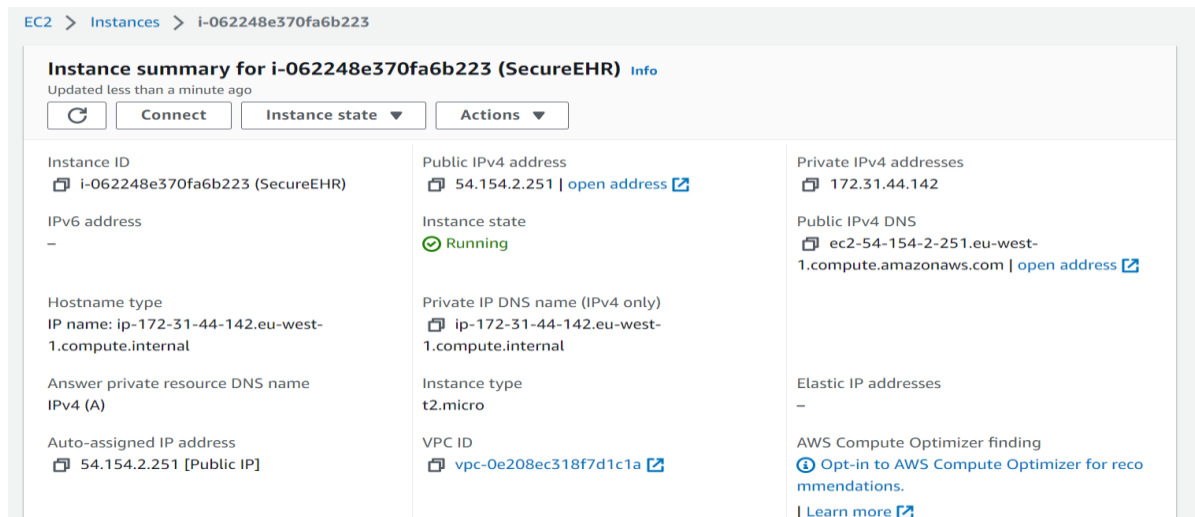


Fig 23: Instance Summary

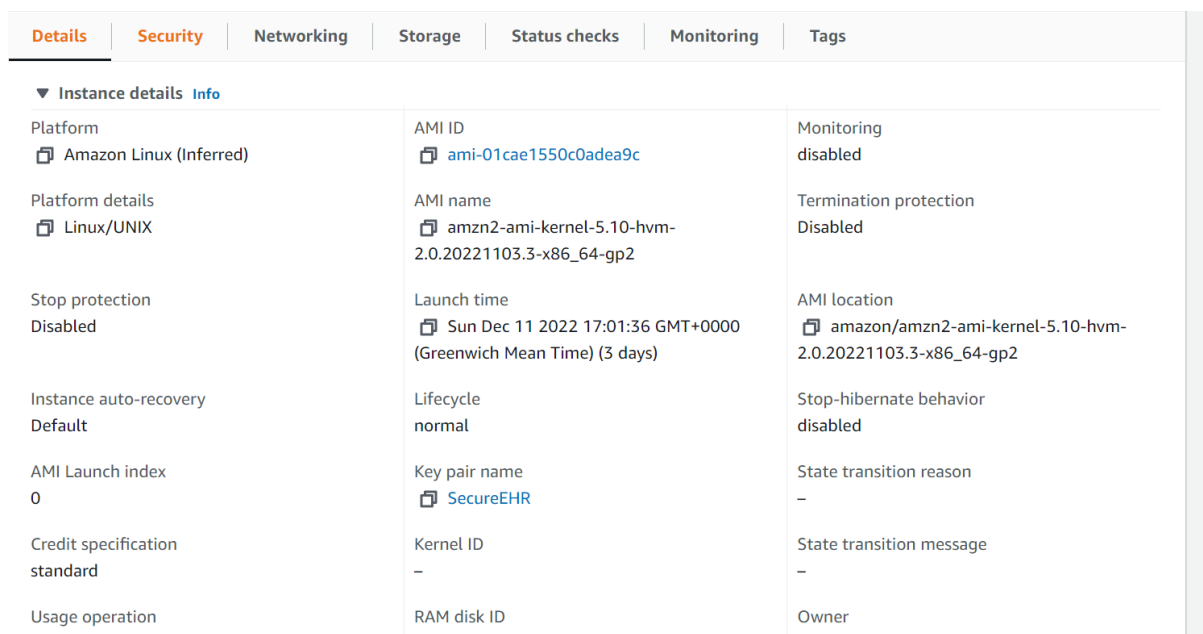


Fig 24: Instance Details

Then the instance needs to be connected to the putty by clicking the connect. It will be then redirect to SSH. The next step is to install Java and upgrade the AWS EC2 Linux Server's software packages so that the Apache Tomcat Server can be run and deploy application. Then the application will be started using following commands.

```
sudo systemctl start apache2
```

```
sudo systemctl enable apache2
```

Fig 25: Start Service

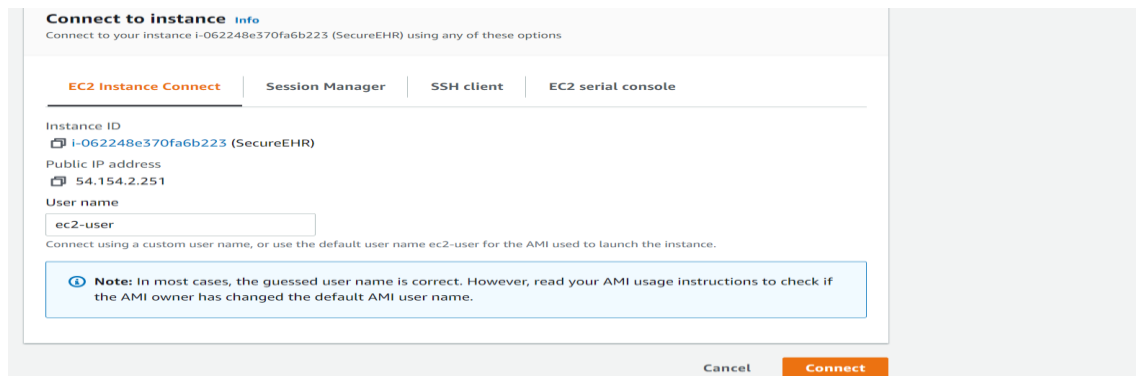


Fig 26: Connect to SSH

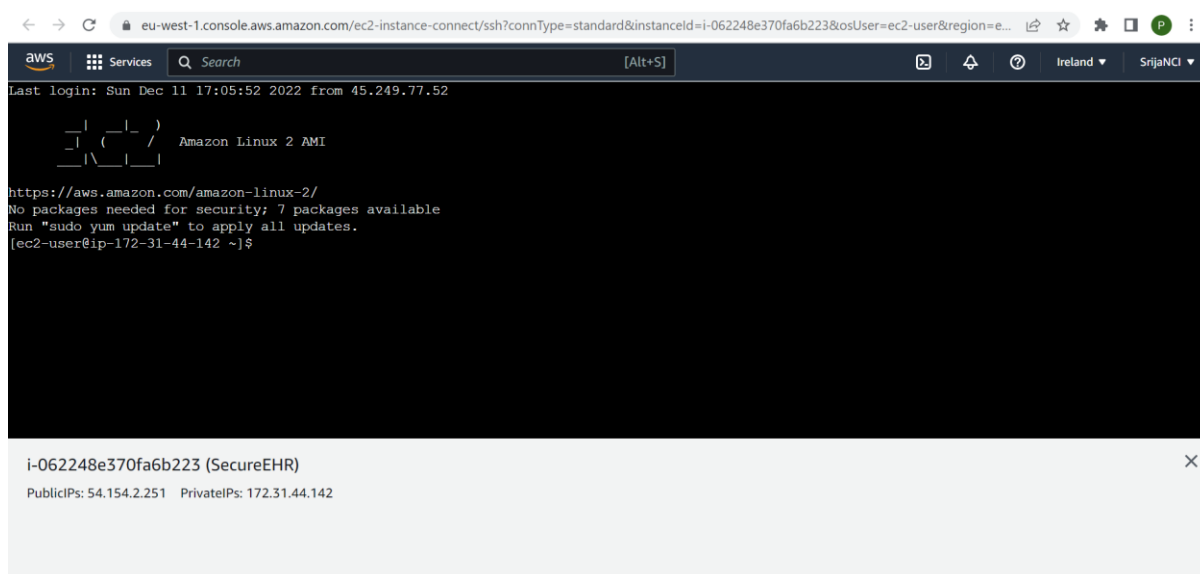


Fig 27: Application Deployment

## References

References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at <https://libguides.ncirl.ie/referencing>

You can use a reference management system such as Zotero or Mendeley to cite in MS Word.

Di Angelo, M. and Slazer, G., 2020, May. Wallet contracts on Ethereum. In 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-2). IEEE.

Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X. and Wang, F.Y., 2019. Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(11), pp.2266-2277.