# DEPLOYING A HYBRID INFRASTRUCTURE FOR RESEARCHERS IN AWS

Ihona Maria Correa de Cabo

HOL02

High-performance and distributed computing for big data

2023-2024

# INDEX

# 1. INTRODUCTION

The aim of this assignment is to create a hybrid cloud infrastructure within Amazon Web Services (AWS) to support a research team that uses Jupyter Notebooks for data analysis. This setup will involve a mix of public and private resources to allow secure analysis of data while providing a way to share findings with the public.

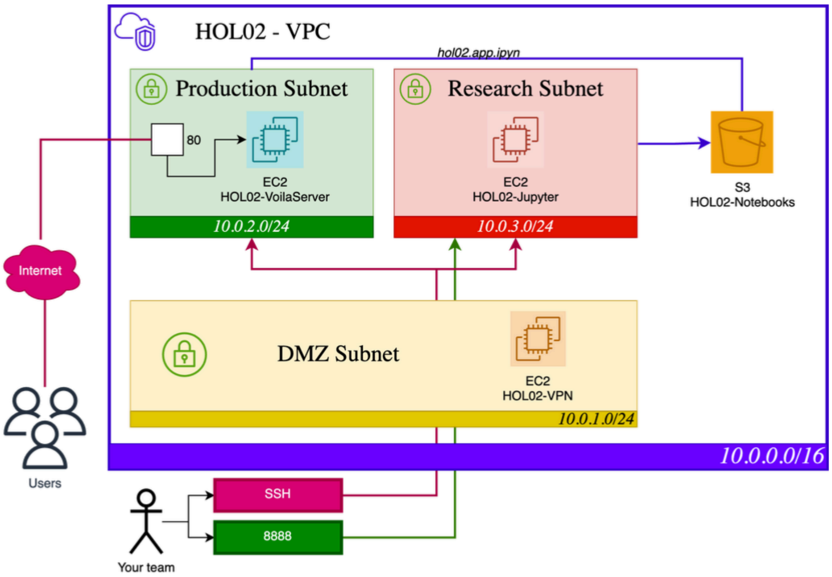


*Figure 1.- Architecture of the proposed hybrid infrastructure.*

This architecture contains a Virtual Private Cloud named HOL02 - VPC, which is an isolated cloud environment within AWS that holds all the resources. Inside this VPC, there are 3 subnets:

a) <u>Production Subnet:</u> Contains the HOL02-VoilaServer EC2 instance, which is exposed to the internet via port 80 (HTTP). Ideally, by using the Voila Server, which turns Jupyter Notebooks into web applications, the team can share their results publicly without exposing the data or analysis environment.

b) <u>Research Subnet:</u> Contains the HOL02-Jupyter EC2 instance, which hosts the Jupyter Notebook server. It is private and can only be accessible through the VPN.

c) <u>DMZ (De-Militarized Zone) Subnet:</u> Contains the HOL02-VPN EC2 instance. It is a semi-secure area that provides a buffer between the untrusted internet and the protected internal network. This subnet hosts the VPN server that the team can use to securely connect to the research subnet.

The VPC is connected to the internet via an internet gateway named HOL02-IGW, allowing the HOL02-VoilaServer to share content to the public internet.

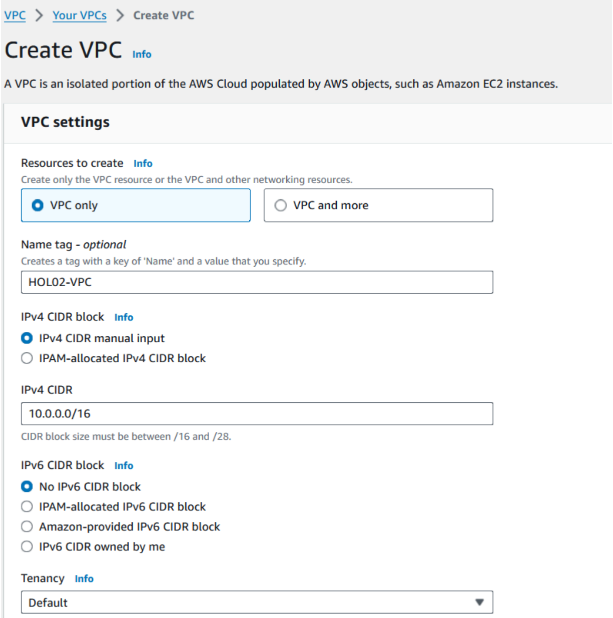
Each server (VPN, VoilaServer, Jupyter) is an EC2 instance, which is a virtual server in AWS's cloud.

The architecture is designed to provide a secure environment for the research team to analyze data using Jupyter Notebooks while at the same time ensuring that the results can be shared with the public through a web server, without compromising the research environment's security.

# 2. INFRAESTRUCTURE SET UP

## 2.1. VPC and subnets

First of all, the VPC is created. In order to achieve that, we must log into the AWS management console and navigate to the VPC dashboard. Then, we create the VPC named “HOL02-VPC”. The "IPv4 CIDR block" field is set to "10.0.0.0/16".



VPC > Your VPCs > Create VPC

Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create Info

Create only the VPC resource or the VPC and other networking resources.

● VPC only ○ VPC and more

Name tag - *optional*

Creates a tag with a key of 'Name' and a value that you specify.

HOL02-VPC

IPv4 CIDR block Info

● IPv4 CIDR manual input

○ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

CIDR block size must be between /16 and /28.

IPv6 CIDR block Info

● No IPv6 CIDR block

○ IPAM-allocated IPv6 CIDR block

○ Amazon-provided IPv6 CIDR block

○ IPv6 CIDR owned by me

Tenancy Info

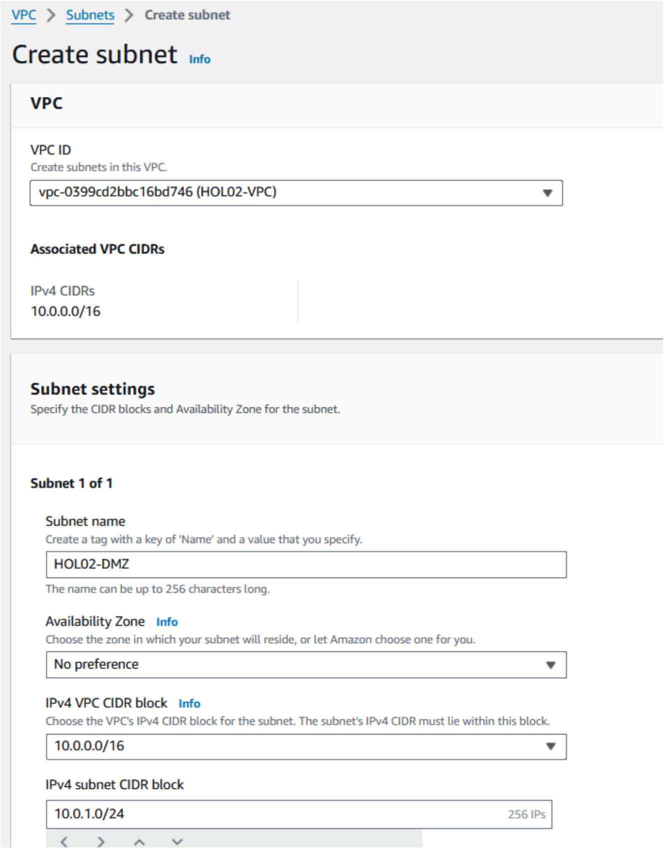
Default ▼

*Figure 2.- VPC creation.*

Next, from the VPC Dashboard, we select "Subnets" in the sidebar.

A subnet is a segmented portion of a VPC that allows to partition this VPC into smaller networks to better organize and secure the resources.

According to the assignment, we must create 3 subnets, one for the production, another for the research group, and finally the DMZ subnet.
When creating the subnets, we select the newly created HOL02-VPC from the drop menu and then create each of the subnets in the same way as *Figure 3*, each subnet with a different IPv4 subnet CIDR block.



VPC > Subnets > Create subnet

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.
vpc-0399cd2bbc16bd746 (HOL02-VPC)

Associated VPC CIDRs

IPv4 CIDRs
10.0.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

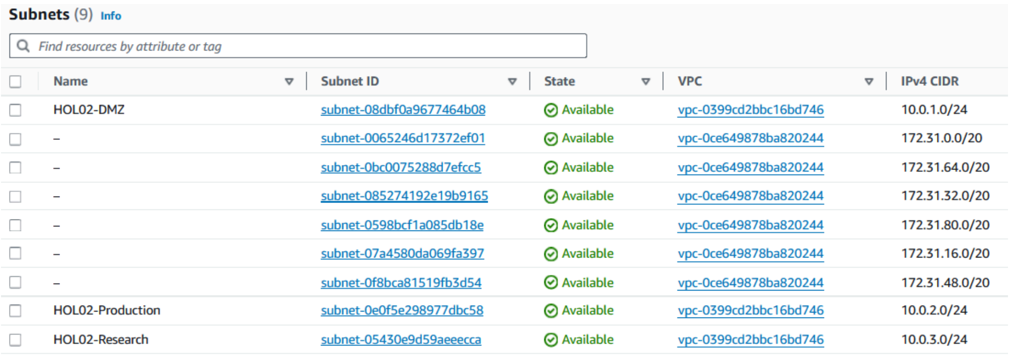
Subnet name
Create a tag with a key of 'Name' and a value that you specify.
HOL02-DMZ
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
No preference

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.
10.0.0.0/16

IPv4 subnet CIDR block
10.0.1.0/24 256 IPs

*Figure 3.-DMZ subnet creation.*



Subnets (9) Info

Find resources by attribute or tag

| □ | Name | Subnet ID | State | VPC | IPv4 CIDR |
|---|---|---|---|---|---|
| □ | HOL02-DMZ | subnet-08dbf0a9677464b08 | ✓ Available | vpc-0399cd2bbc16bd746 | 10.0.1.0/24 |
| □ | – | subnet-0065246d17372ef01 | ✓ Available | vpc-0ce649878ba820244 | 172.31.0.0/20 |
| □ | – | subnet-0bc0075288d7efcc5 | ✓ Available | vpc-0ce649878ba820244 | 172.31.64.0/20 |
| □ | – | subnet-085274192e19b9165 | ✓ Available | vpc-0ce649878ba820244 | 172.31.32.0/20 |
| □ | – | subnet-0598bcf1a085db18e | ✓ Available | vpc-0ce649878ba820244 | 172.31.80.0/20 |
| □ | – | subnet-07a4580da069fa397 | ✓ Available | vpc-0ce649878ba820244 | 172.31.16.0/20 |
| □ | – | subnet-0f8bca81519fb3d54 | ✓ Available | vpc-0ce649878ba820244 | 172.31.48.0/20 |
| □ | HOL02-Production | subnet-0e0f5e298977dbc58 | ✓ Available | vpc-0399cd2bbc16bd746 | 10.0.2.0/24 |
| □ | HOL02-Research | subnet-05430e9d59aeeecca | ✓ Available | vpc-0399cd2bbc16bd746 | 10.0.3.0/24 |

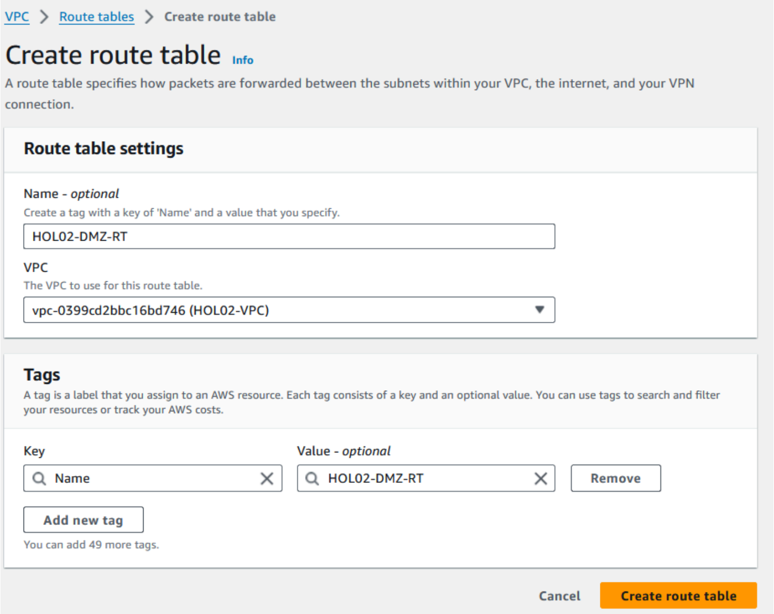
*Figure 4.- All the subnets created.*

Each CIDR block represents a range of IP addresses that can be assigned to resources within that subnet. By assigning different CIDR blocks to different subnets, we ensure that the IP addresses used in one subnet do not overlap with the ones used in another subnet within the same VPC.

## 2.2. Route tables

In AWS, a route table is a set of rules, called routes, that are used to determine where network traffic from the VPC should be directed. Each subnet in a VPC must be associated with a route table, and the route table controls the routing of traffic within that subnet.

To proceed, we go back to the VPC Dashboard and select "Route Tables" from the sidebar. We create one route table per each subnet:



VPC > Route tables > Create route table

### Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

**Route table settings**

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.
HOL02-DMZ-RT

VPC
The VPC to use for this route table.
vpc-0399cd2bbc16bd746 (HOL02-VPC)

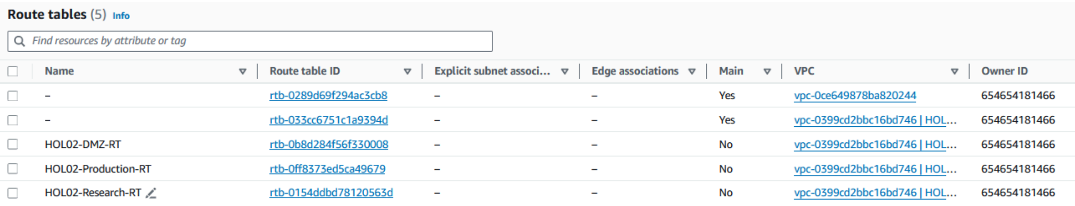
**Tags**

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

| Key | Value - *optional* |
|---|---|
| Name | HOL02-DMZ-RT |

You can add 49 more tags.

*Figure 5*

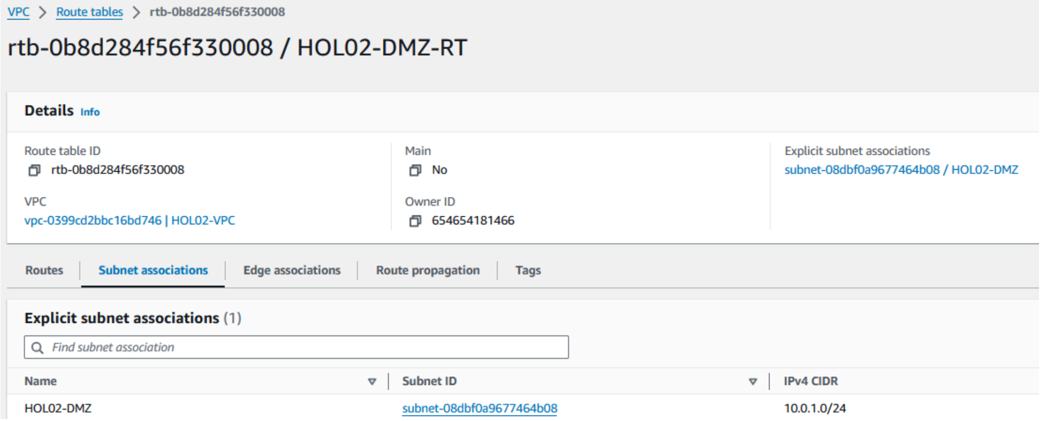


**Route tables** (5) Info

| ☐ | Name | Route table ID | Explicit subnet associ... | Edge associations | Main | VPC | Owner ID |
|---|---|---|---|---|---|---|---|
| ☐ | – | rtb-0289d69f294ac3cb8 | – | – | Yes | vpc-0ce649878ba820244 | 654654181466 |
| ☐ | – | rtb-033cc6751c1a9394d | – | – | Yes | vpc-0399cd2bbc16bd746 \| HOL... | 654654181466 |
| ☐ | HOL02-DMZ-RT | rtb-0b8d284f56f330008 | – | – | No | vpc-0399cd2bbc16bd746 \| HOL... | 654654181466 |
| ☐ | HOL02-Production-RT | rtb-0ff8373ed5ca49679 | – | – | No | vpc-0399cd2bbc16bd746 \| HOL... | 654654181466 |
| ☐ | HOL02-Research-RT | rtb-0154ddbd78120563d | – | – | No | vpc-0399cd2bbc16bd746 \| HOL... | 654654181466 |

*Figure 6*

Afterward, we associate each route table with their subnet:



VPC > Route tables > rtb-0b8d284f56f330008

rtb-0b8d284f56f330008 / HOL02-DMZ-RT

**Details** Info

Route table ID: rtb-0b8d284f56f330008

Main: No

Explicit subnet associations: subnet-08dbf0a9677464b08 / HOL02-DMZ

VPC: vpc-0399cd2bbc16bd746 | HOL02-VPC

Owner ID: 654654181466

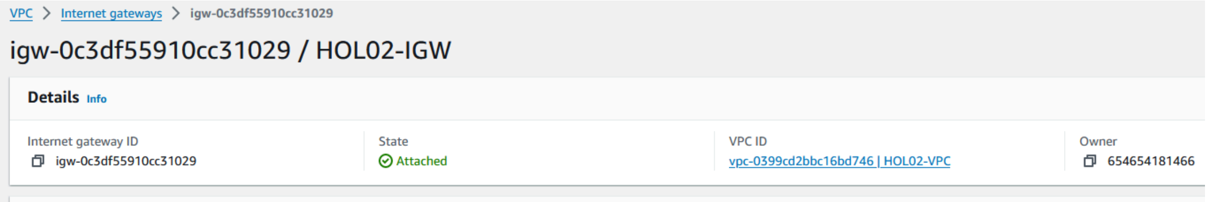
**Explicit subnet associations** (1)

| Name | Subnet ID | IPv4 CIDR |
|---|---|---|
| HOL02-DMZ | subnet-08dbf0a9677464b08 | 10.0.1.0/24 |

*Figure 7*

Moreover, the subnet that should be accessible from the internet is the one hosting the HOL02-VoilaServer, since this server is meant to share the team's findings with the public. Therefore, we must ensure that the subnet associated with the VoilaServer (the Production Subnet), has internet access.

To do that, first we create an internet gateway (*Figure 8*) and attach it to our VPC.



VPC > Internet gateways > igw-0c3df55910cc31029

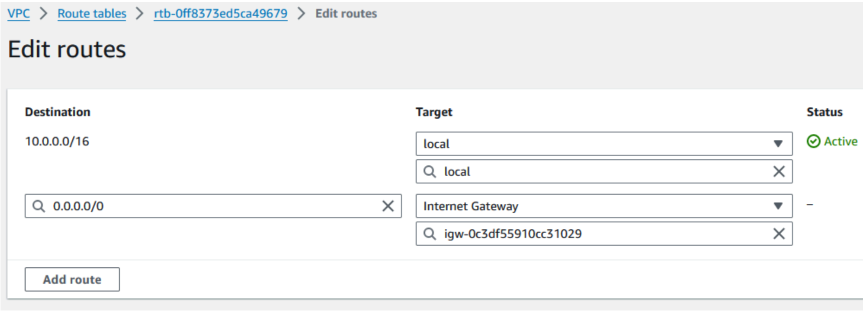
igw-0c3df55910cc31029 / HOL02-IGW

**Details** Info

| Internet gateway ID | State | VPC ID | Owner |
|---|---|---|---|
| igw-0c3df55910cc31029 | Attached | vpc-0399cd2bbc16bd746 \| HOL02-VPC | 654654181466 |

*Figure 8*- Internet gateway creation.

Then, we edit the route table associated with the Production Subnet (HOL02-Production) by adding a new route with 0.0.0.0/0 as the destination and the Internet Gateway as the target (*Figure 9*).



VPC > Route tables > rtb-0ff8373ed5ca49679 > Edit routes

Edit routes

| Destination | Target | Status |
|---|---|---|
| 10.0.0.0/16 | local<br>local | Active |
| 0.0.0.0/0 | Internet Gateway<br>igw-0c3df55910cc31029 | - |

*Figure 9*

We follow the same steps for the route table associated with the HOL02-DMZ subnet. At the moment, the final network configuration is as follows:

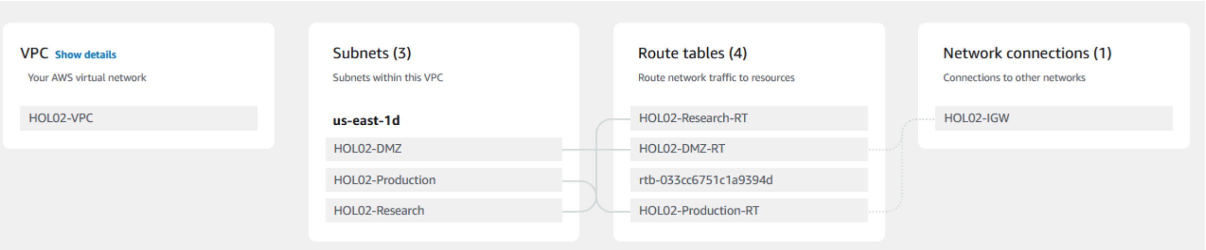


*Figure 10*

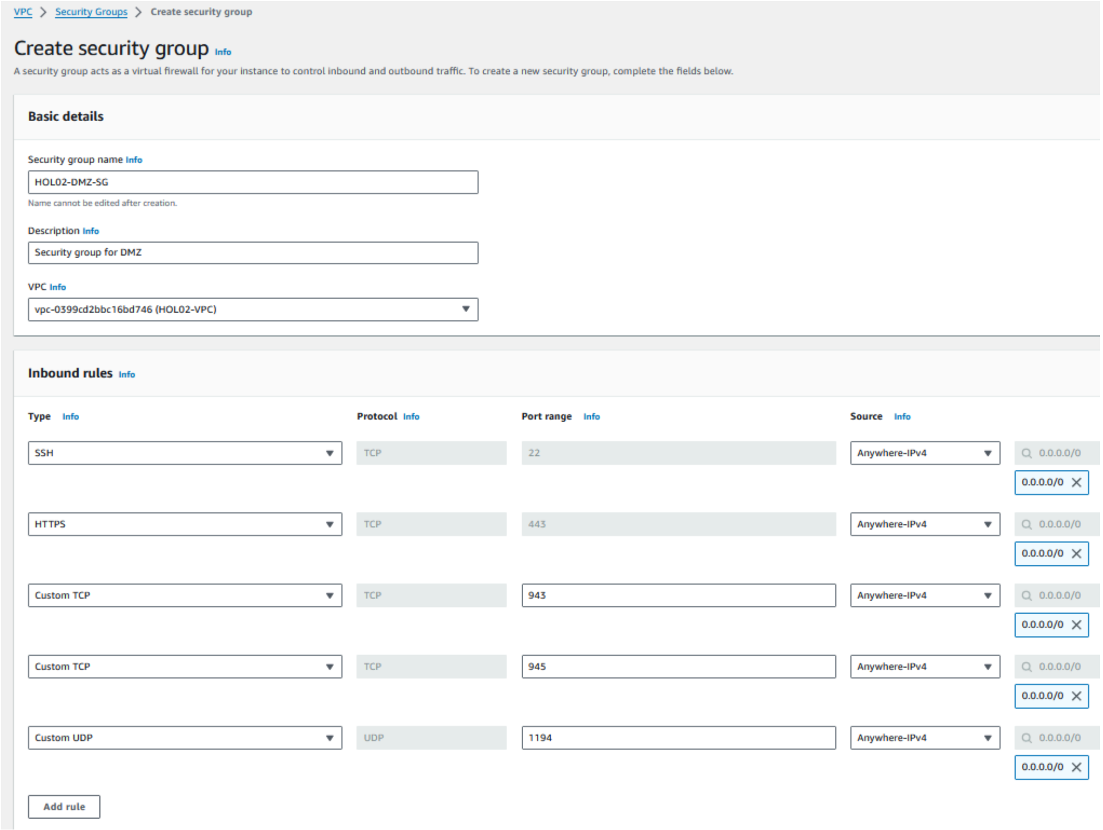
## 2.3. Security groups

In AWS, security groups act as virtual firewalls for EC2 instances and other resources within the VPC. They basically control inbound and outbound traffic at the instance level, and this enables to specify which traffic is allowed to reach our instances.

- I*nbound Rules*: These rules define the traffic that is allowed to reach the instances.

- *Outbound Rules*: These rules define the outbound traffic that is allowed to leave the instances. For instance, we can specify that all outbound traffic is allowed, or restrict it to specific protocols and ports.

Still within the VPC Dashboard, we click on "Security Groups" in the sidebar. We create 3 different security groups (HOL02-DMZ-SG, HOL02-Production-SG and HOL02-Research-SG) with the specified ports from the assignment instructions (*table 1*). The security groups are created in the same way as *Figure 8:*



VPC > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name Info: HOL02-DMZ-SG

Name cannot be edited after creation.

Description Info: Security group for DMZ

VPC Info: vpc-0399cd2bbc16bd746 (HOL02-VPC)

**Inbound rules** Info

| Type | Protocol | Port range | Source | |
|---|---|---|---|---|
| SSH | TCP | 22 | Anywhere-IPv4 | 0.0.0.0/0 |
| HTTPS | TCP | 443 | Anywhere-IPv4 | 0.0.0.0/0 |
| Custom TCP | TCP | 943 | Anywhere-IPv4 | 0.0.0.0/0 |
| Custom TCP | TCP | 945 | Anywhere-IPv4 | 0.0.0.0/0 |
| Custom UDP | UDP | 1194 | Anywhere-IPv4 | 0.0.0.0/0 |

*Figure 11*

| Security group | Inbound rules | Outbund rules |
|---|---|---|
| HOL02-DMZ-SG | 22(SSH)<br>443 (HTTPS)<br>943 (TCP)<br>945 (TCP)<br>1194 (UDP) | All traffic |
| HOL02-Production-SG | 80 (HTTP)<br>443 (HTTPS)<br>22 (SSH) - only from HOL02-DMZ-SG | All traffic |
| HOL02-Research-SG | 22 (SSH) - only from HOL02-DMZ-SG<br>8888 (TCP) - only from HOL02-DMZ-SG | All traffic |

*Table 1*

For the Production security group, we will be allowing HTTP and HTTPS traffic from anywhere, and SSH traffic only from the DMZ security group.

## 2.4. EC2 instances

An EC2 instance is a virtual server in Amazon's EC2 for running applications on the AWS infrastructure. When creating an instance, we are essentially renting a virtual server from AWS to run our own applications.

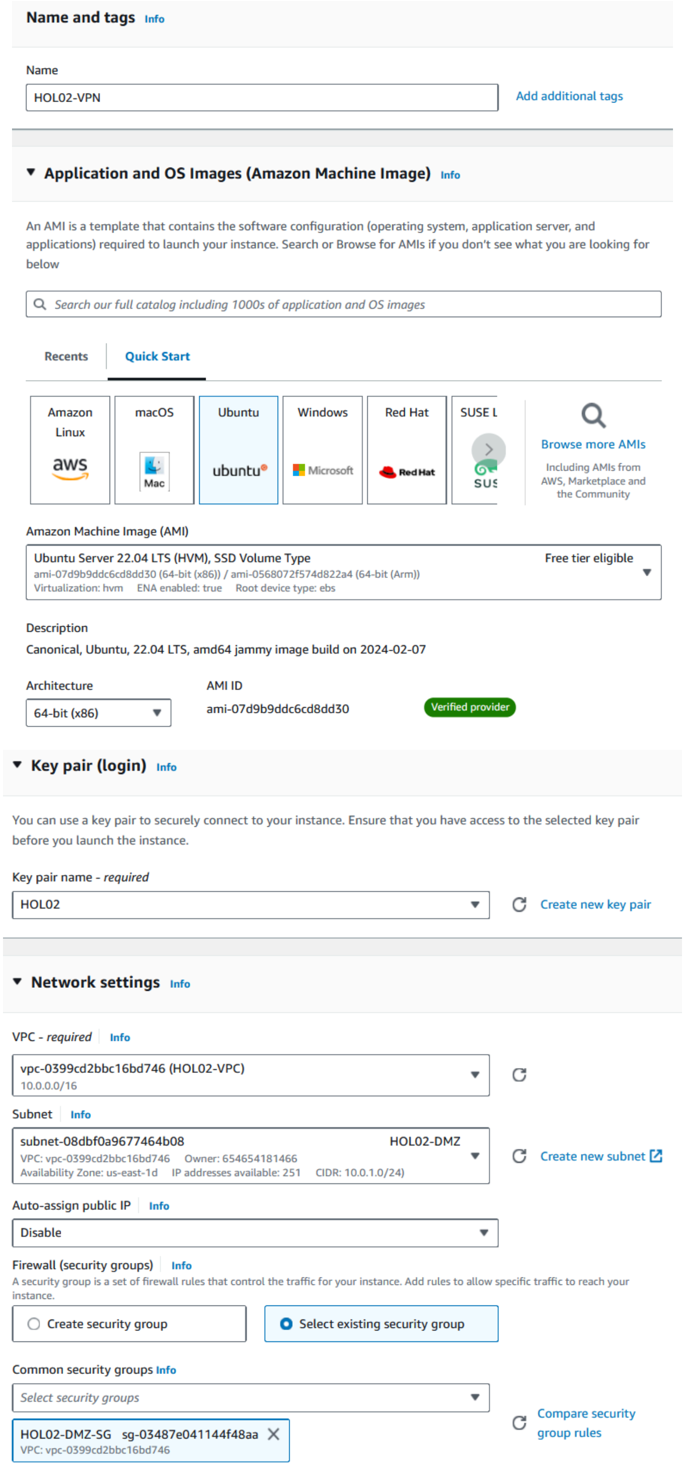
For this assignment, we have to create three EC2 instances: HOL02-VPN, HOL02-VoilaServer and HOL02-Jupyter.

We go to the EC2 dashboard and launch an instance. For each instance, we select the characteristics following *Table 2:*

| **EC2 instance name** | **Subnet associated** | **AMI** | **Instance type** |
|---|---|---|---|
| HOL02-VPN | HOL02-DMZ | Ubuntu 22.04 LTS | t2.micro |
| HOL02-VoilaServer | HOL02-Production | Amazon Linux 2 | t2.micro |
| HOL02-Jupyter | HOL02-Research | Amazon Linux 2 | t2.micro |

*Table 2*

Here is an example of the EC2 instance configuration for HOL02-VPN:



**Name and tags** Info

Name

HOL02-VPN

Add additional tags

▼ **Application and OS Images (Amazon Machine Image)** Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

*Search our full catalog including 1000s of application and OS images*

Recents | Quick Start

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-07d9b9ddc6cd8dd30 (64-bit (x86)) / ami-0568072f574d822a4 (64-bit (Arm))
Virtualization: hvm  ENA enabled: true  Root device type: ebs

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2024-02-07

Architecture: 64-bit (x86)

AMI ID: ami-07d9b9ddc6cd8dd30 — Verified provider

▼ **Key pair (login)** Info

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*

HOL02

Create new key pair

▼ **Network settings** Info

VPC - *required* Info

vpc-0399cd2bbc16bd746 (HOL02-VPC)
10.0.0.0/16

Subnet Info

subnet-08dbf0a9677464b08 — HOL02-DMZ
VPC: vpc-0399cd2bbc16bd746  Owner: 654654181466
Availability Zone: us-east-1d  IP addresses available: 251  CIDR: 10.0.1.0/24)

Create new subnet

Auto-assign public IP Info

Disable

Firewall (security groups) Info

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

Common security groups Info

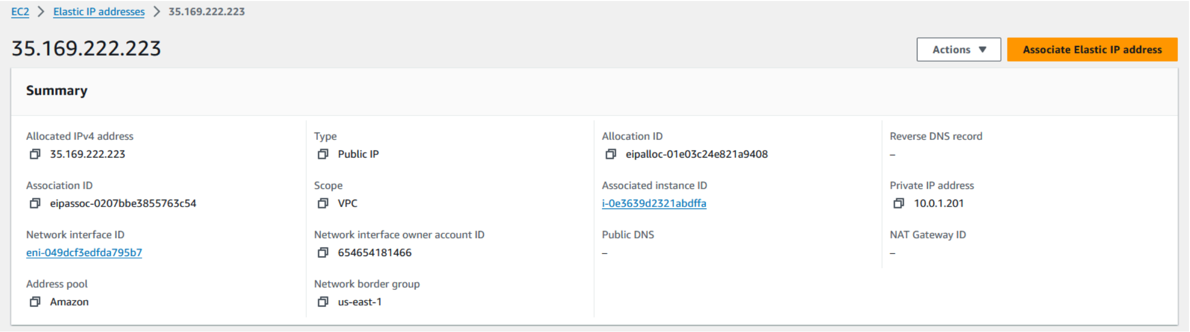
*Select security groups*

HOL02-DMZ-SG sg-03487e041144f48aa ✕
VPC: vpc-0399cd2bbc16bd746

Compare security group rules

*Figure 12*

HOL02-VPN requires an elastic IP, as specified in the assignment, so we have to go to the "Elastic IPs" section in the EC2 dashboard, create an Elastic IP and associate it with the EC2 instance HOL02-VPN:



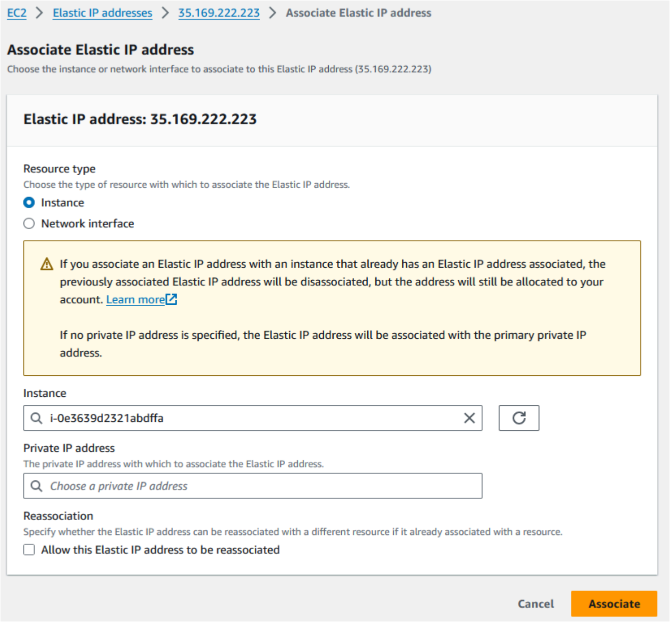
EC2 > Elastic IP addresses > 35.169.222.223

35.169.222.223

Summary

| | | | |
|---|---|---|---|
| Allocated IPv4 address<br>35.169.222.223 | Type<br>Public IP | Allocation ID<br>eipalloc-01e03c24e821a9408 | Reverse DNS record<br>– |
| Association ID<br>eipassoc-0207bbe3855763c54 | Scope<br>VPC | Associated instance ID<br>i-0e3639d2321abdffa | Private IP address<br>10.0.1.201 |
| Network interface ID<br>eni-049dcf3edfda795b7 | Network interface owner account ID<br>654654181466 | Public DNS<br>– | NAT Gateway ID<br>– |
| Address pool<br>Amazon | Network border group<br>us-east-1 | | |

*Figure 13.-Elastic IP creation*



EC2 > Elastic IP addresses > 35.169.222.223 > Associate Elastic IP address

Associate Elastic IP address

Choose the instance or network interface to associate to this Elastic IP address (35.169.222.223)

Elastic IP address: 35.169.222.223

Resource type

Choose the type of resource with which to associate the Elastic IP address.

● Instance

○ Network interface

⚠ If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. Learn more

If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.

Instance

i-0e3639d2321abdffa

Private IP address

The private IP address with which to associate the Elastic IP address.

*Choose a private IP address*

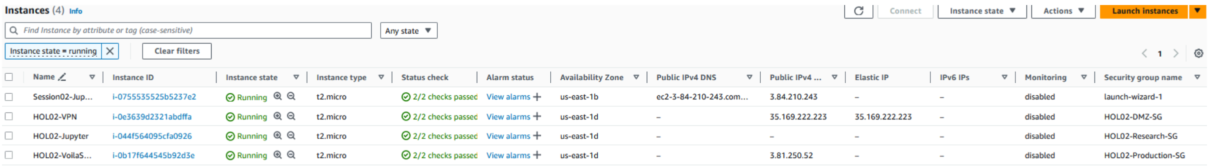
Reassociation

Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.

□ Allow this Elastic IP address to be reassociated

*Figure 14.- We associate the elastic IP to the instance.*

Here is an overview of the instances created:



Instances (4) Info

*Find Instance by attribute or tag (case-sensitive)*

Any state ▼

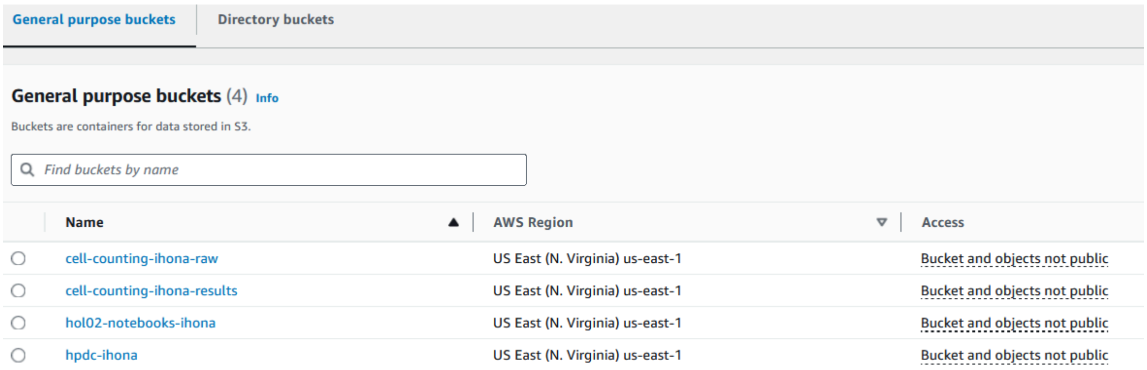
Instance state = running

| □ | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public IPv4 ... | Elastic IP | IPv6 IPs | Monitoring | Security group name |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| □ | Session02-Jup... | i-0755535525b5237e2 | Running | t2.micro | 2/2 checks passed | View alarms + | us-east-1b | ec2-3-84-210-243.com... | 3.84.210.243 | – | – | disabled | launch-wizard-1 |
| □ | HOL02-VPN | i-0e3639d2321abdffa | Running | t2.micro | 2/2 checks passed | View alarms + | us-east-1d | – | 35.169.222.223 | 35.169.222.223 | – | disabled | HOL02-DMZ-SG |
| □ | HOL02-Jupyter | i-044f564095cfa0926 | Running | t2.micro | 2/2 checks passed | View alarms + | us-east-1d | – | – | – | – | disabled | HOL02-Research-SG |
| □ | HOL02-VoilaS... | i-0b17f644545b92d3e | Running | t2.micro | 2/2 checks passed | View alarms + | us-east-1d | – | 3.81.250.52 | – | – | disabled | HOL02-Production-SG |

*Figure 15*

## 2.5. S3 buckets

Finally, we must create the S3 bucket that will include all the notebooks used by the research team. We navigate to the S3 dashboard, select S3 and click on "create bucket". The bucket is named "hol02-notebooks-ihona".



**General purpose buckets** (4)

Buckets are containers for data stored in S3.

| | Name | AWS Region | Access |
|---|---|---|---|
| ○ | cell-counting-ihona-raw | US East (N. Virginia) us-east-1 | Bucket and objects not public |
| ○ | cell-counting-ihona-results | US East (N. Virginia) us-east-1 | Bucket and objects not public |
| ○ | hol02-notebooks-ihona | US East (N. Virginia) us-east-1 | Bucket and objects not public |
| ○ | hpdc-ihona | US East (N. Virginia) us-east-1 | Bucket and objects not public |

*Figure 16*

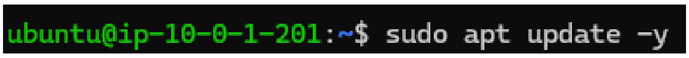
# 3. OPEN VPN CONFIGURATION

OpenVPN is a robust and configurable VPN solution that allows to securely connect devices to a remote network over the internet. In this context, OpenVPN allows us to securely access the Jupyter server, which is not directly exposed to the public internet, by routing the connection through the VPN.

Firstly, we must connect to the EC2 instance named HOL02-VPN. To do it, we open a terminal in the computer and go to the directory where there is our private key.Then, we execute this command,

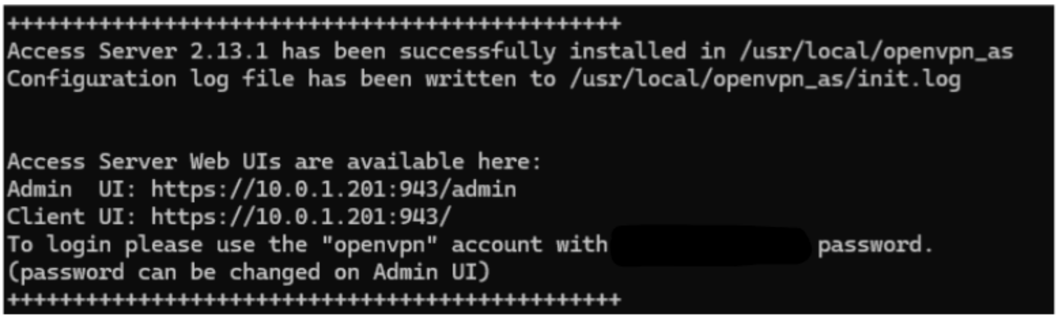
*ssh -i aws01 ubuntu@35.169.222.223*

where "aws01" is the name of the private key and 35.169.222.223 is the public IP of the instance. Once connected to the EC2 instance, we use the following commands to install OpenVPN on the EC2 instance:



```
ubuntu@ip-10-0-1-201:~$ sudo apt update -y
```

*sudo apt update -y*
*sudo apt install ca-certificates gnupg wget net-tools -y*
*sudo wget https://as-repository.openvpn.net/as-repo-public.asc -qO /etc/apt/trusted.gpg.d/as-repo-public.asc*
*sudo echo "deb [arch=amd64 signed-by=/etc/apt/trusted.gpg.d/as-repo-public.asc] http://as-repository.openvpn.net/as/debian jammy main" | sudo tee /etc/apt/sources.list.d/openvpn-as-repo.list*
*sudo apt update && sudo apt install openvpn-as -y*



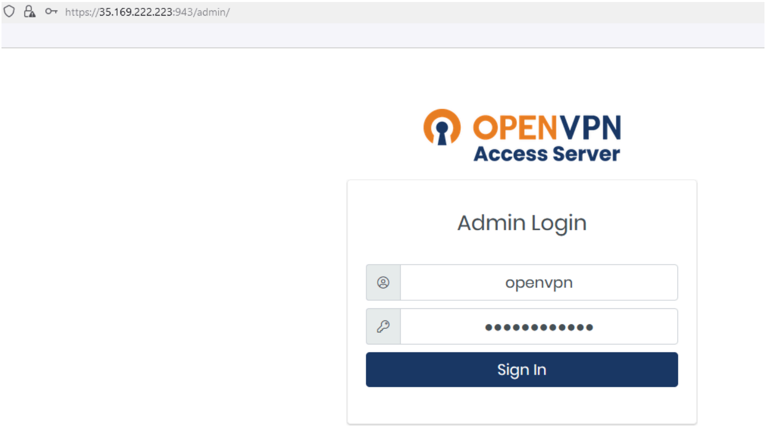
```
+++++++++++++++++++++++++++++++++++++++++++++++
Access Server 2.13.1 has been successfully installed in /usr/local/openvpn_as
Configuration log file has been written to /usr/local/openvpn_as/init.log


Access Server Web UIs are available here:
Admin  UI: https://10.0.1.201:943/admin
Client UI: https://10.0.1.201:943/
To login please use the "openvpn" account with [redacted] password.
(password can be changed on Admin UI)
+++++++++++++++++++++++++++++++++++++++++++++++
```

The previous commands install OpenVPN Access Server, which is a package that simplifies the setup and management of OpenVPN connections.

The next step is to configure the OpenVPN. For that, we open a browser and go to this direction:

*https://35.169.222.223:943/admin/*

This redirects us to this page and we log in with the default username and password obtained from the terminal:



https://35.169.222.223:943/admin/

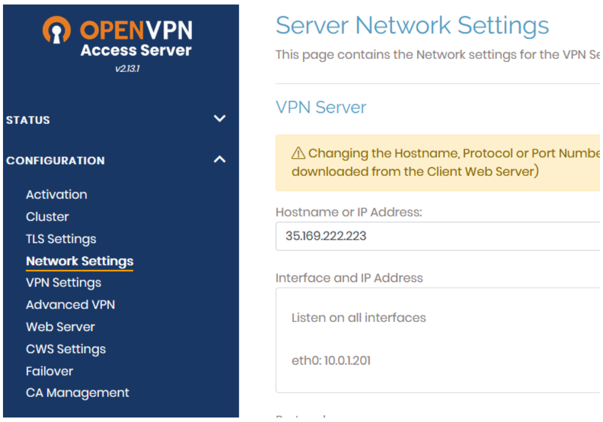
Admin Login

openvpn

●●●●●●●●●●●●

*Figure 17*

After the log in, we can access the OpenVPN web interface, so we navigate to Network Settings and change the hostname to the elastic IP as follows:



Server Network Settings

This page contains the Network settings for the VPN Se

VPN Server

⚠ Changing the Hostname, Protocol or Port Numbe downloaded from the Client Web Server)

Hostname or IP Address: 35.169.222.223

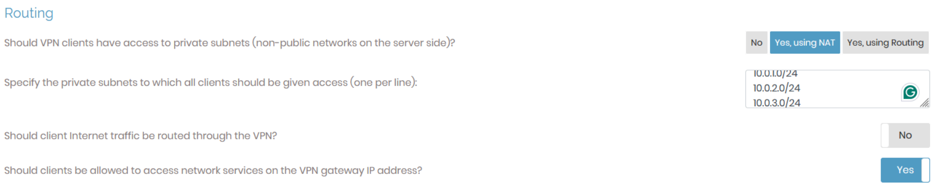
Interface and IP Address

Listen on all interfaces

eth0: 10.0.1.201

*Figure 18*

Then, we navigate to VPN Settings and add the subnets CIDR one per line:



Routing

Should VPN clients have access to private subnets (non-public networks on the server side)? No [✓ Yes, using NAT] Yes, using Routing

Specify the private subnets to which all clients should be given access (one per line):

```
10.0.1.0/24
10.0.2.0/24
10.0.3.0/24
```

Should client Internet traffic be routed through the VPN? No

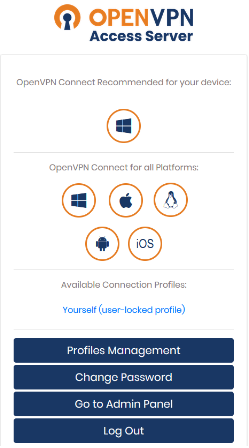
Should clients be allowed to access network services on the VPN gateway IP address? Yes

*Figure 19*

After configuring OpenVPN, we have to install the OpenVPN client. To achieve this, we open the browser and go to this direction:

*https://35.169.222.223:943*

We install the Yourself (user-locked) profile (*Figure 16*):



OpenVPN Connect Recommended for your device:

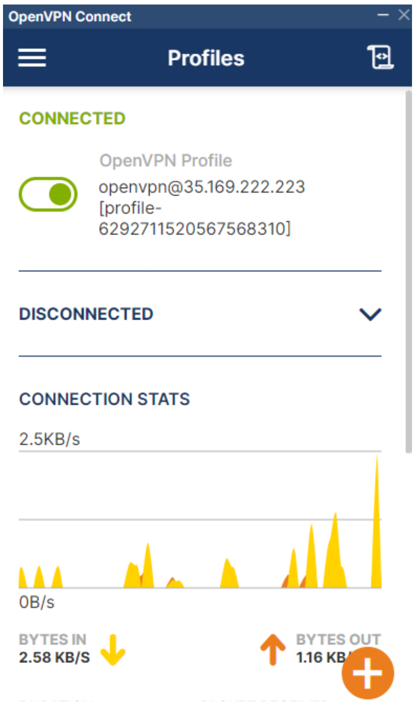
OpenVPN Connect for all Platforms:

Available Connection Profiles:

Yourself (user-locked profile)

*Figure  20*

Now, if we provide the password obtained from the terminal, we are able to connect to the VPN (*Figure 17*). It will be necessary to be connected to the VPN in order to access the Jupyter server (port 8888 and port 22).



CONNECTED

OpenVPN Profile
openvpn@35.169.222.223
[profile-
6292711520567568310]

DISCONNECTED

CONNECTION STATS

BYTES IN 2.58 KB/S

BYTES OUT 1.16 KB

*Figure 21- VPN connected*

## 4. JUPYTER NOTEBOOK SETUP

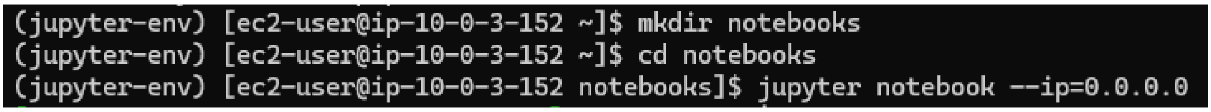
First, we connect to the HOL02-Jupyter EC2 instance. Since the connection is over the VPN, we have to put the private IP address (*10.0.3.152)*:

*ssh -i aws01 ec2-user@10.0.3.152*

Afterwards, we create a new python environment and we automatically activate it with this command:

*echo "source jupyter-env/bin/activate" >> ~/.bashrc*

We create a directory for the notebooks and run the command to initialize jupyter:

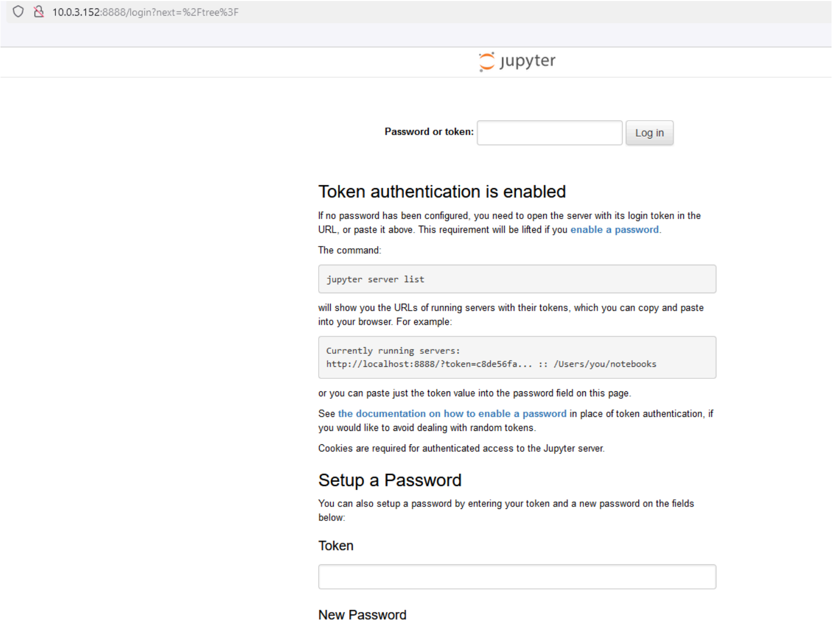


```
(jupyter-env) [ec2-user@ip-10-0-3-152 ~]$ mkdir notebooks
(jupyter-env) [ec2-user@ip-10-0-3-152 ~]$ cd notebooks
(jupyter-env) [ec2-user@ip-10-0-3-152 notebooks]$ jupyter notebook --ip=0.0.0.0
```

Finally, we go to the browser and paste this direction to be redirected to jupyter:

[http://10.0.3.152:8888/](http://10.0.3.152:8888/)

Once in the provided address, we must paste the token prompted in the terminal to access:



10.0.3.152:8888/login?next=%2Ftree%3F

**Password or token:** Log in

Token authentication is enabled

If no password has been configured, you need to open the server with its login token in the URL, or paste it above. This requirement will be lifted if you **enable a password**.

The command:

```
jupyter server list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

See **the documentation on how to enable a password** in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to the Jupyter server.

Setup a Password

You can also setup a password by entering your token and a new password on the fields below:

Token

New Password

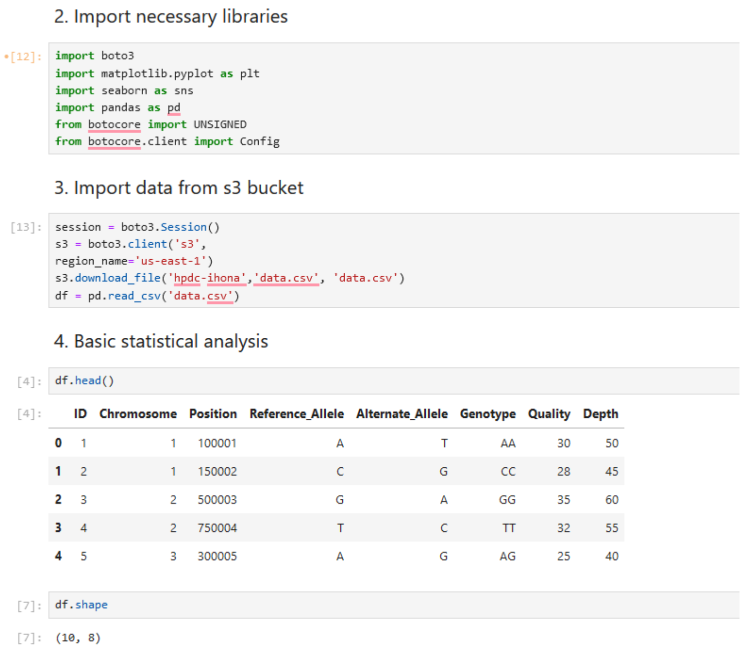
*Figure 22*

For convenience, we can run this command to to run Jupyter Notebook in the background and do not be asked for the token each time:

```
nohup jupyter notebook --ip=0.0.0.0 --notebook-dir=/home/ec2-user/notebooks --no-browser > /home/ec2-user/notebooks/jupyter.log  2>&1 &
```

We create a sample Jupyter Notebook named “hol02-app.ipynb”.

It has been decided to use the file data.csv, which contains genomic data. This dataset is in an s3 bucket named hpdc-ihona. We download the dataset from the bucket by doing the following commands:



2. Import necessary libraries

```
•[12]: import boto3
       import matplotlib.pyplot as plt
       import seaborn as sns
       import pandas as pd
       from botocore import UNSIGNED
       from botocore.client import Config
```

3. Import data from s3 bucket

```
[13]: session = boto3.Session()
      s3 = boto3.client('s3',
      region_name='us-east-1')
      s3.download_file('hpdc-ihona','data.csv', 'data.csv')
      df = pd.read_csv('data.csv')
```

4. Basic statistical analysis

```
[4]: df.head()
```

[4]:

| | ID | Chromosome | Position | Reference_Allele | Alternate_Allele | Genotype | Quality | Depth |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 100001 | A | T | AA | 30 | 50 |
| 1 | 2 | 1 | 150002 | C | G | CC | 28 | 45 |
| 2 | 3 | 2 | 500003 | G | A | GG | 35 | 60 |
| 3 | 4 | 2 | 750004 | T | C | TT | 32 | 55 |
| 4 | 5 | 3 | 300005 | A | G | AG | 25 | 40 |

```
[7]: df.shape

[7]: (10, 8)
```

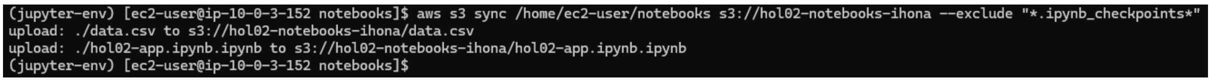
*Figure 23*

After doing the analysis of the data (which contains some basic statistical analysis and visualizations), we can synchronize the jupyter notebook we just created with the s3 bucket created in the 2.5 section. We can do this either manually or with a tool named cron.

a) To do it manually, we have to run this command:

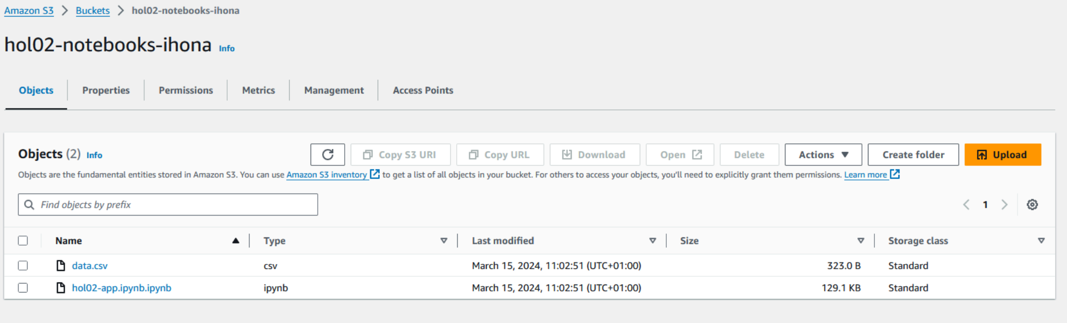
*aws s3 sync /home/ec2-user/notebooks s3://hol02-notebooks-ihona --exclude "*.ipynb_checkpoints*"*

Where /home/ec2-user/notebooks/, is the path where the notebook is stored, and hol02-notebooks-ihona the bucket we want to save the jupyter notebook in:



```
(jupyter-env) [ec2-user@ip-10-0-3-152 notebooks]$ aws s3 sync /home/ec2-user/notebooks s3://hol02-notebooks-ihona --exclude "*.ipynb_checkpoints*"
upload: ./data.csv to s3://hol02-notebooks-ihona/data.csv
upload: ./hol02-app.ipynb.ipynb to s3://hol02-notebooks-ihona/hol02-app.ipynb.ipynb
(jupyter-env) [ec2-user@ip-10-0-3-152 notebooks]$
```

We can check the bucket to ensure that everything has worked out fine:



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| □ | Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|---|
| □ | data.csv | csv | March 15, 2024, 11:02:51 (UTC+01:00) | 323.0 B | Standard |
| □ | hol02-app.ipynb.ipynb | ipynb | March 15, 2024, 11:02:51 (UTC+01:00) | 129.1 KB | Standard |

*Figure 24*

b) Cron jobs are a powerful tool for scheduling tasks automatically at specific times. In order to do the synchronization with cron, first we have to install it using this command:

*sudo yum install cronie -y*

Then we start the cron service:

*sudo systemctl start crond*

And enable Cron Service to Start on Boot:

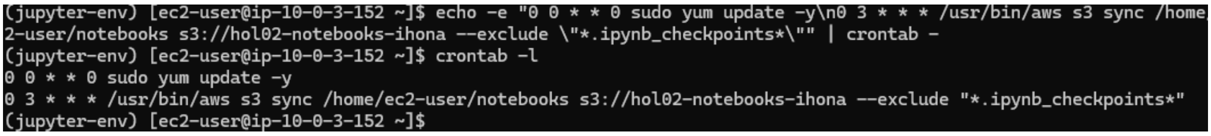
*sudo systemctl enable crond*

Then we execute this command:

*echo -e "0 0 * * 0 sudo yum update -y\n0 3 * * * /usr/bin/aws s3 sync /home/ec2-user/notebooks s3://hol02-notebooks-ihona --exclude \"*.ipynb_checkpoints*\"" | crontab -*

The first cron job *0 0 * * 0 sudo yum update -y* is scheduled to run at 00:00 (midnight) on Sunday (day 0 of the week). This job will update the system's packages automatically using yum.
The second cron job *0 3 * * * /usr/bin/aws s3 sync /home/ec2-user/notebooks s3://hol02-notebooks-ihona --exclude "*.ipynb_checkpoints*"* is scheduled to run at 03:00 every day. It synchronizes the Jupyter notebooks from the specified directory (/home/ec2-user/notebooks) to the S3 bucket (hol02-notebooks-ihona), excluding any Jupyter checkpoint files.

We can verify the job scheduling with the command *crontab -l*



```
(jupyter-env) [ec2-user@ip-10-0-3-152 ~]$ echo -e "0 0 * * 0 sudo yum update -y\n0 3 * * * /usr/bin/aws s3 sync /home
2-user/notebooks s3://hol02-notebooks-ihona --exclude \"*.ipynb_checkpoints*\"" | crontab -
(jupyter-env) [ec2-user@ip-10-0-3-152 ~]$ crontab -l
0 0 * * 0 sudo yum update -y
0 3 * * * /usr/bin/aws s3 sync /home/ec2-user/notebooks s3://hol02-notebooks-ihona --exclude "*.ipynb_checkpoints*"
(jupyter-env) [ec2-user@ip-10-0-3-152 ~]$
```

This command lists all cron jobs for the current user. The output confirmed that both jobs are correctly scheduled.

# 5. VOILA SERVER CONFIGURATION

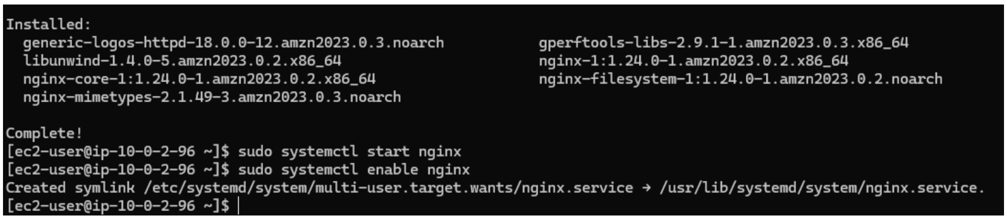
Voila turns Jupyter notebooks into standalone web applications, where the content of the Notebook's is displayed, with the option to hide the code. This is very useful for sharing data visualizations, reports, or interactive dashboards derived from Jupyter notebooks.

Nginx, on the other hand, is a high-performance web server software. In a real case, it could be used as the web server that hosts the Voila web application, making it accessible from the internet.
However, in order to simplify the activity, we will assume that the default page of nginx is the Voila server.

First of all, to configure the server, we connect through SSH to the EC2 instance named HOL02-VoilaServer, also accessible via its private IP.

Then, we use the following commands to install Nginx on the EC2 instance:

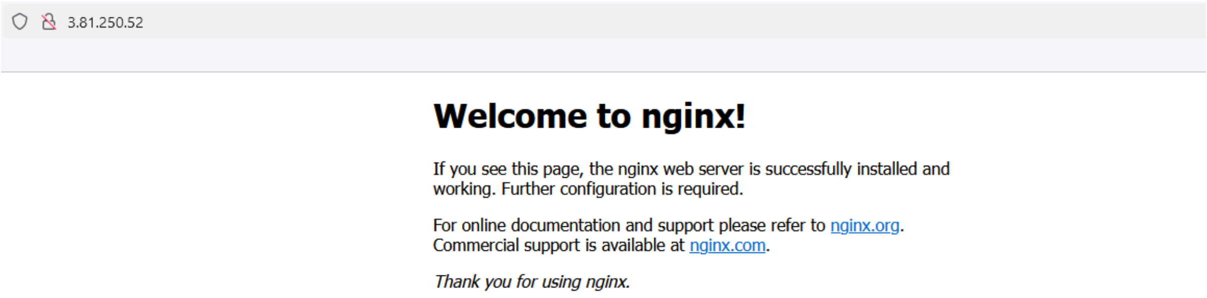
*sudo dnf install nginx*
*sudo systemctl start nginx*
*sudo systemctl enable nginx*



```
Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch          gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  libunwind-1.4.0-5.amzn2023.0.2.x86_64                      nginx-1:1.24.0-1.amzn2023.0.2.x86_64
  nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64                  nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch
  nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
[ec2-user@ip-10-0-2-96 ~]$ sudo systemctl start nginx
[ec2-user@ip-10-0-2-96 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-10-0-2-96 ~]$ |
```

We can check that we can access the Voila Server from the internet by going to the browser and using the public IP address of the HOL02-VoilaServer EC2 instance:

*http://3.81.250.52*



## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

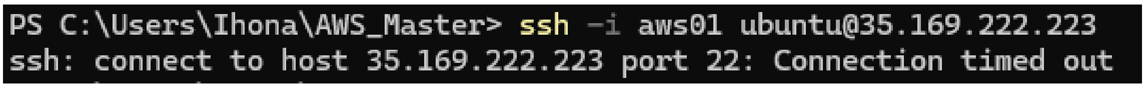
*Thank you for using nginx.*

*Figure 25*

# 6. CHALLENGES AND TROUBLESHOOTING

During this assignment, we had to face several challenges and perform some troubleshooting to overcome them.

The first issue was regarding the connection to the HOL02-VPN EC2 instance. When trying to connect, we would get a time out message:



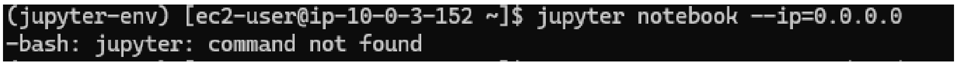
```
PS C:\Users\Ihona\AWS_Master> ssh -i aws01 ubuntu@35.169.222.223
ssh: connect to host 35.169.222.223 port 22: Connection timed out
```

To troubleshoot the problem, we had a look at the HOL02-DMZ subnet configuration and the correspondent route tables. We realized that the EC2 instance had an elastic IP attached but its route table was not linking to an internet gateway. If the subnet where the instance resides is not public or is not associated with a route table that has a route to the internet, the instance is not able to communicate to the internet or allow SSH access.

Therefore, we solved the issue by adding a new route with 0.0.0.0/0 as the destination and the Internet Gateway as the target.

Also, at the beginning, we were not able to connect to the VPN via OpenVPN client. The reason was that the connection was not taking the elastic IP so there was a connection time out. This issue was solved by downloading the user-locked profile.

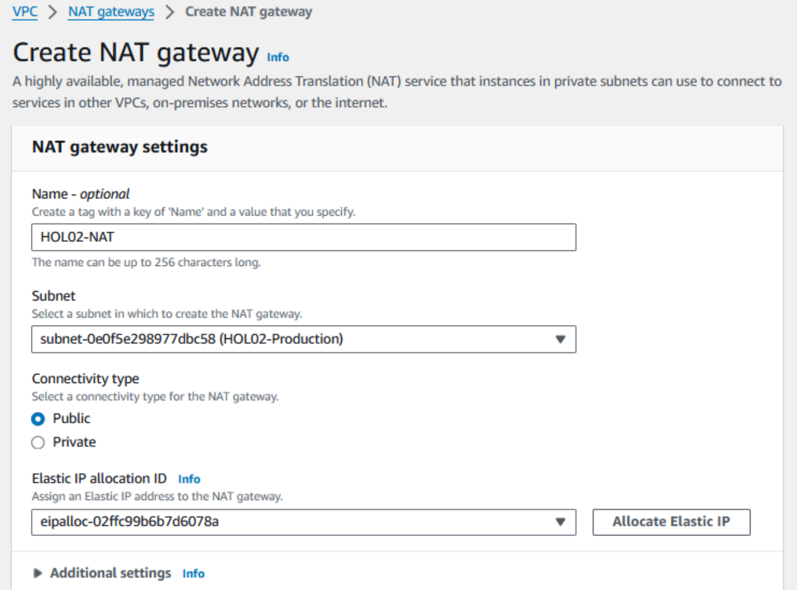
Moreover, after connecting to HOL02-Jupyter EC2 instance, and creating and activating a virtual environment, we could not create a jupyter notebook. There was an error because we had not installed jupyer:



```
(jupyter-env) [ec2-user@ip-10-0-3-152 ~]$ jupyter notebook --ip=0.0.0.0
-bash: jupyter: command not found
```

However, we were not able to install jupyter because the HOL02-Research subnet, where the Jupyter instance is located, was effectively acting as a private subnet and did not have a direct route to the Internet Gateway. Therefore, the instance could not reach the internet directly to install packages. After doing some research, we found that to enable instances in private subnets access the internet (for software installation, updates, etc.) without being directly exposed to the internet, a **NAT Gateway** was used. This gateway is placed in a subnet that has access to the internet and allows outbound internet access for instances in private subnets.

We decided to create a NAT Gateway in the HOL02-Production subnet (*Figure* 22). Then, the route table for HOL02-Research subnet was updated to route outbound internet traffic (0.0.0.0/0) through this NAT Gateway (*Figure* 23).



Create NAT gateway Info

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

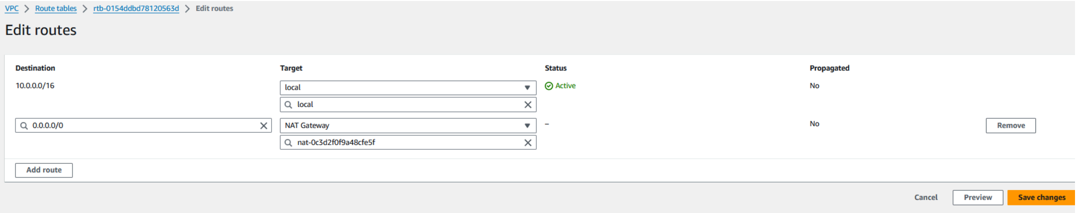
Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.
HOL02-NAT
The name can be up to 256 characters long.

Subnet
Select a subnet in which to create the NAT gateway.
subnet-0e0f5e298977dbc58 (HOL02-Production)

Connectivity type
Select a connectivity type for the NAT gateway.
● Public
○ Private

Elastic IP allocation ID Info
Assign an Elastic IP address to the NAT gateway.
eipalloc-02ffc99b6b7d6078a

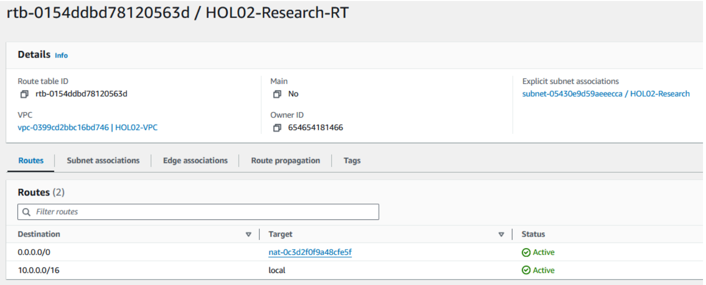
*Figure 26*



Edit routes

| Destination | Target | Status | Propagated |
|---|---|---|---|
| 10.0.0.0/16 | local<br>local | Active | No |
| 0.0.0.0/0 | NAT Gateway<br>nat-0c3d2f0f9a48cfe5f | - | No |

*Figure 27*


rtb-0154ddbd78120563d / HOL02-Research-RT

Details Info

Route table ID: rtb-0154ddbd78120563d
Main: No
Explicit subnet associations: subnet-05430e9d59aeeecca / HOL02-Research
VPC: vpc-0399cd2bbc16bd746 | HOL02-VPC
Owner ID: 654654181466

Routes (2)

| Destination | Target | Status |
|---|---|---|
| 0.0.0.0/0 | nat-0c3d2f0f9a48cfe5f | Active |
| 10.0.0.0/16 | local | Active |

*Figure 28*

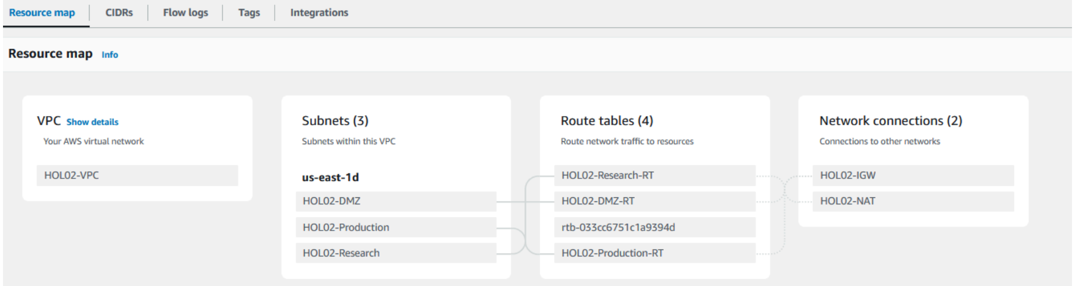


*Figure 29.- Network configuration.*

After doing these network configurations, we were able to successfully install jupyter in the research instance while maintaining a secure environment.

Lastly, there was an issue regarding the jupyter notebook used for the assignment. At first, it was thought that the notebook would get the data.csv file from the bucket hpdc-ihona. However, this made that for every time we wanted to run the notebook, we had to grant access to the virtual machine to access the bucket through the commands *aws configure* and *aws_session_token.* This was not very effective so it was decided to put the data directly in the bucket (hol02-notebooks-ihona) where the notebook is saved. To read the file, we only need to execute: *df = pd.read_csv('data.csv').*

# 7. CONCLUSIONS

During this laboratory session, the requested infrastructure (VPC, subnets, route tables, security groups, EC2 instances and s3 buckets) was successfully set up.
The OpenVPN was also successfully configured and was necessary in order to connect to the VoilaServer instance via SSH and the Jupyter instance. Moreover, a Jupyter notebook was set up in a safe environment only accessible through the VPN and also synchronized with the s3 bucket, both manually and through cron jobs.

On the other hand, the Volia Server was properly configured and the default Nginx page is accessible from the internet via the public IP, enabling the “production team” to share results with the public in a safe way.
Throughout the report, screenshots providing evidence, tests and troubleshooting have been provided.

In conclusion, AWS services were used to provide a solution and built an hybrid infrastructure for researchersin the cloud. This practical session has been very useful to learn new concepts regarding cloud computing with AWS and has undoubtedly improved my solvem-problem abilities.