High Performance and Distributed Computing for Big Data

Unit 3: Unit 3: Cloud Systems and Big Data Management

Session 3 - Introduction to AWS + EC2

Francesc Solsona Tehas francesc.solsona@udl.cat

Universitat Rovira i Virgili and Universitat de Lleida

We will focus on AWS in this session. AWS offers a variety of cloud services.

Figure 5: AWS services overview

Today, we will be using **EC2** (Elastic Compute Cloud) to create and configure a virtual machine.

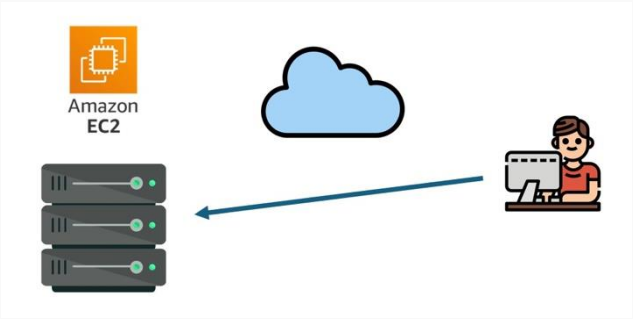


Figure 6: AWS EC2

Connecting to EC2

A remote terminal through SSH

What is a Terminal?

A **terminal** is an interface that allows users to interact with a computer using text commands. Unlike graphical user interfaces (GUIs), terminals allow direct communication with the operating system.

What is SSH?

SSH (Secure Shell) is a protocol that allows secure remote access to machines over the internet. It enables users to execute commands on remote servers as if they were physically present.

When we connect to a remote machine through SSH, we are essentially opening a **remote terminal** on that machine.

How does SSH work?

SSH works with **public and private keys** to establish a secure connection. This eliminates the need for password-based authentication, enhancing security.

Public and Private Keys

- **Public key**: We are going to share this with the server.
- **Private key**: This one is kept secret on our local machine.

Key-Based Authentication

When we connect to a server using SSH, the server (the EC2 machine) checks if the public key matches the private key. If they match, access is granted.

What do you mean by client and server?

The Client-Server Model

The **client-server model** describes how computers communicate over a network.

- The **client** (your local machine) accesses things or **services** on the server.
- The **server** (the remote machine) is the one responsible for providing those services.

Example: Watching a YouTube Video

When you watch a YouTube video, your computer (the client) sends a request to YouTube's servers (the server) to stream the video.

What do you mean by client and server?

Figure 7: Client-server model when watching Netflix

What do you mean by client and server?

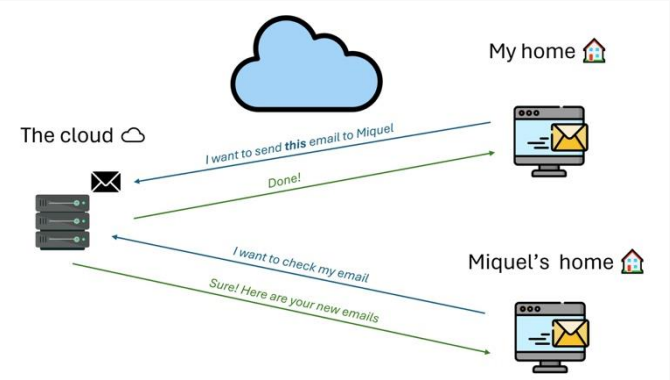


Figure 8: Client-server model when sending emails

Why do I need to *open ports*?

Ports

What Are Ports?

Computers use **ports** to differentiate between different types of network services.

For example:

- **Port 22** → SSH (Remote Login)
- **Port 80** → HTTP (Web Traffic)
- **Port 443** → HTTPS (Secure Web Traffic)

Hotel Analogy

Think of a **hotel**, where different rooms provide different services:

- **Room 22** is the reception (SSH access).
- **Room 80** is a public restaurant (HTTP web service).
- **Room 443** is a private lounge (HTTPS secure access).

Security groups

Configuring Security Groups

Security groups in AWS define **which ports** are open to the internet for which services.

1. By default, AWS **blocks all incoming traffic**.
2. You need to **explicitly allow** access to certain ports (e.g., SSH, HTTP).

What to configure on a Security Group

A security group is just a set of **firewall rules**. You can configure:

- **Inbound rules**: Traffic coming into the server.
- **Outbound rules**: Traffic going out of the server.

Example: Opening Port 22 (SSH)

- **Type:** SSH
- **Port Range:** 22
- **Source:** Your IP address or `0.0.0.0/0` (not recommended in production environments for security reasons).

Today's Work on AWS

Goals for today

Today, we will:

- Create an EC2 instance.
- Connect to it through SSH.
- Set up a Python environment.
- Configure a Jupyter Notebook server to access it from our local machine.

AWS Academy allows educators to provide students with access to real AWS services with **a limited budget and restricted service access**.

Figure 9: AWS Academy

Activating your account

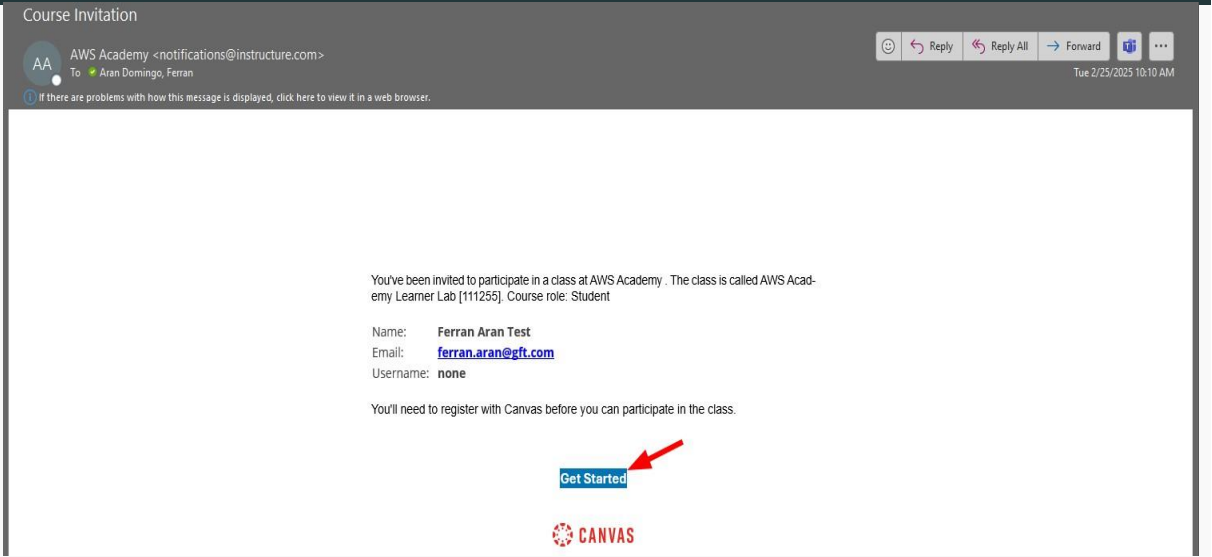


Figure 10: Accepting the invite

Activating your account

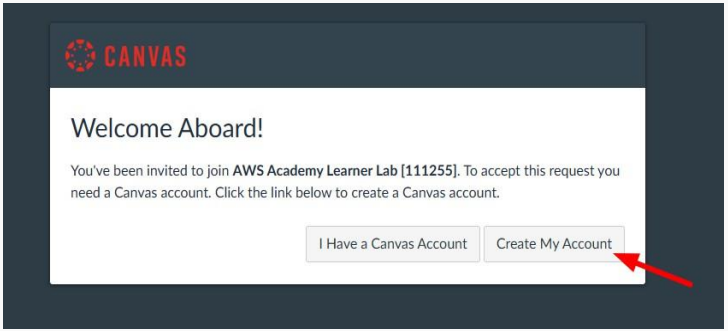


Figure 11: Creating your account

Logging in from now on

You'll have to visit https://awsacademy.instructure.com/ and log in with your credentials as a student.

Figure 12: Login with your credentials

Logging in from now on

You will then receive an email with a verification code to complete the login process.

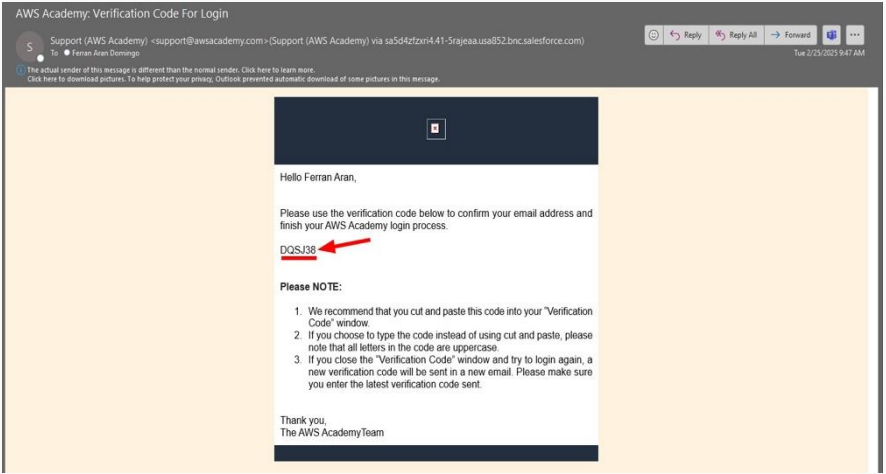


Figure 13: Enter verification code

Available courses

You'll have access to:

1. AWS Cloud Foundation Course (Theory-based learning).

2. AWS Learner Lab (Hands-on experience with AWS services).

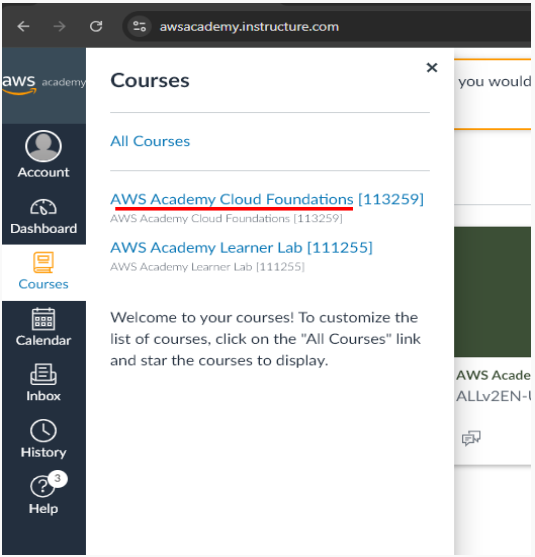


Figure 14: Available courses

Available courses: AWS Cloud Foundation Course

- All the students are invited to complete the **AWS Cloud Foundations** course. This course is available in the AWS Educate platform and it is a great introduction to the AWS services. The course is not mandatory, but it is highly recommended.

- This introductory course is intended for students who seek an overall understanding of cloud computing concepts, independent of specific technical roles. It provides a detailed overview of cloud concepts, AWS core services, security, architecture, pricing, and support.

Available courses: AWS Learner Lab

- We will be using **Learner Lab** to complete today's tasks and future deliverables.
- The Learner Lab is a hands-on environment where you can practice with real AWS services. You will have access to a **limited set of services and a budget** to use them.
- It is a great opportunity to learn how to use AWS services in a real-world scenario since the dashboard is the same as the one used by professionals.
- Getting into the Learner Lab Dashboard can be a bit tricky, so we will see which are the steps to follow.

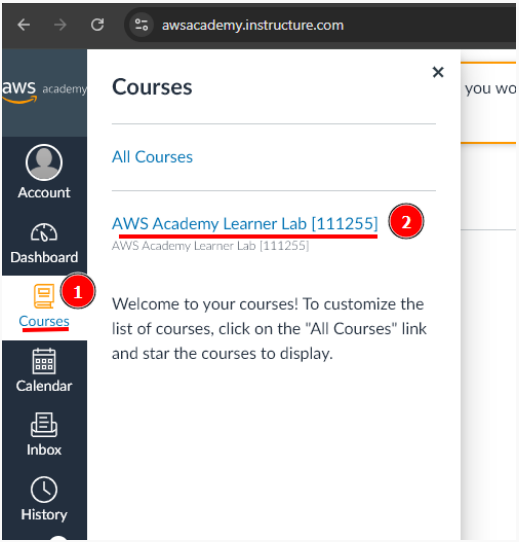


Figure 15: Choosing the learner lab course

Accessing the Learner Lab

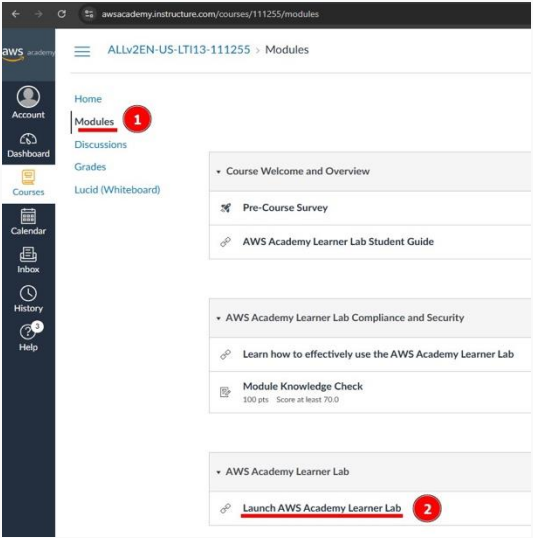


Figure 16: Choosing the learner lab module

Accessing the Learner Lab

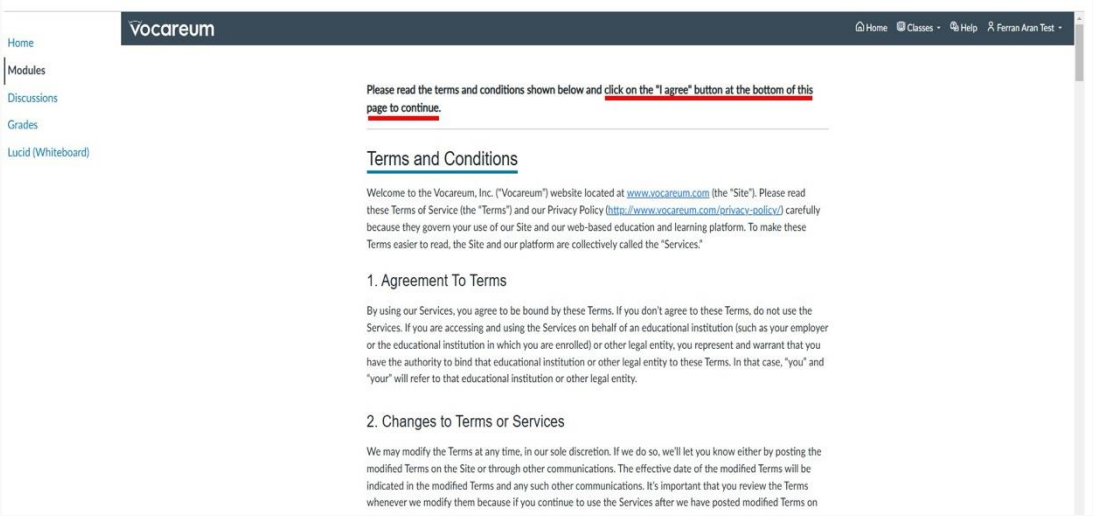


Figure 17: Scroll down and agree the terms and conditions

Accessing the Learner Lab

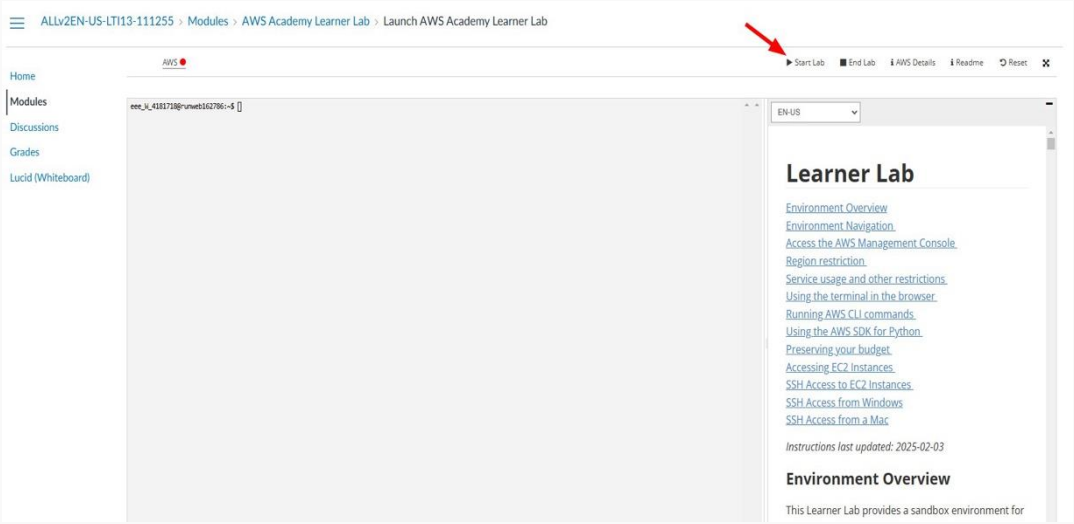


Figure 18: Click on ‘Start’

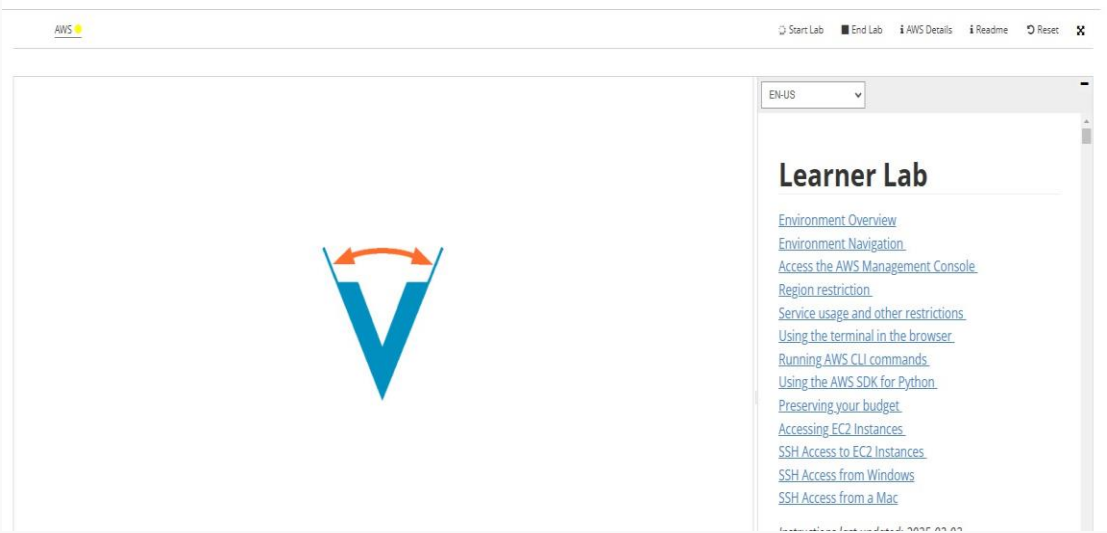


Figure 19: Wait for the environment to set up

Accessing the Learner Lab

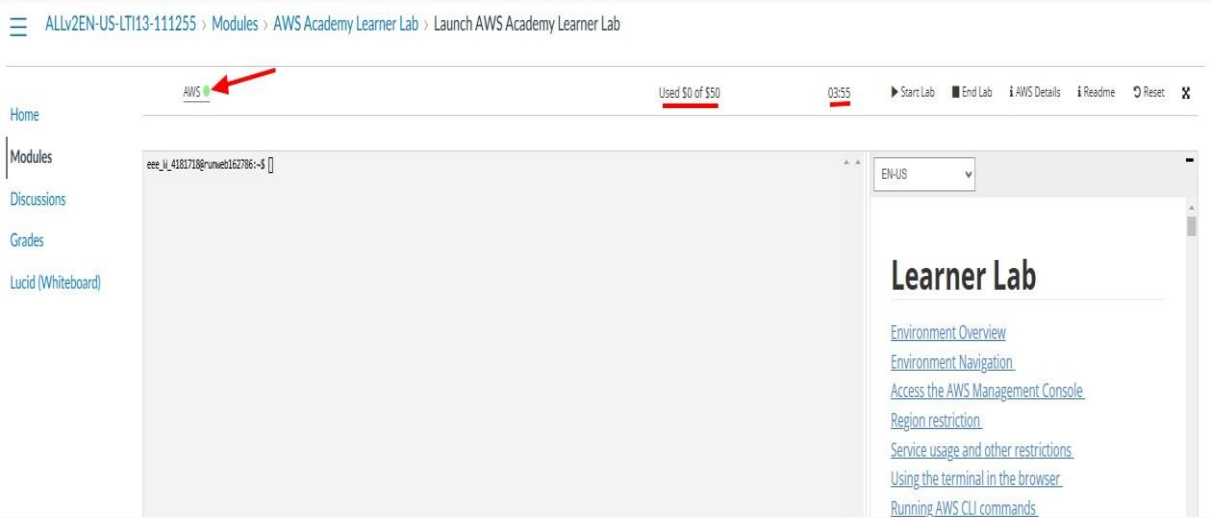


Figure 20: Once it is ready (the AWS buttom becomes green) click on 'AWS'.
Current Budget usage: $0 spent out of a $50 limit.

Accessing the Learner Lab

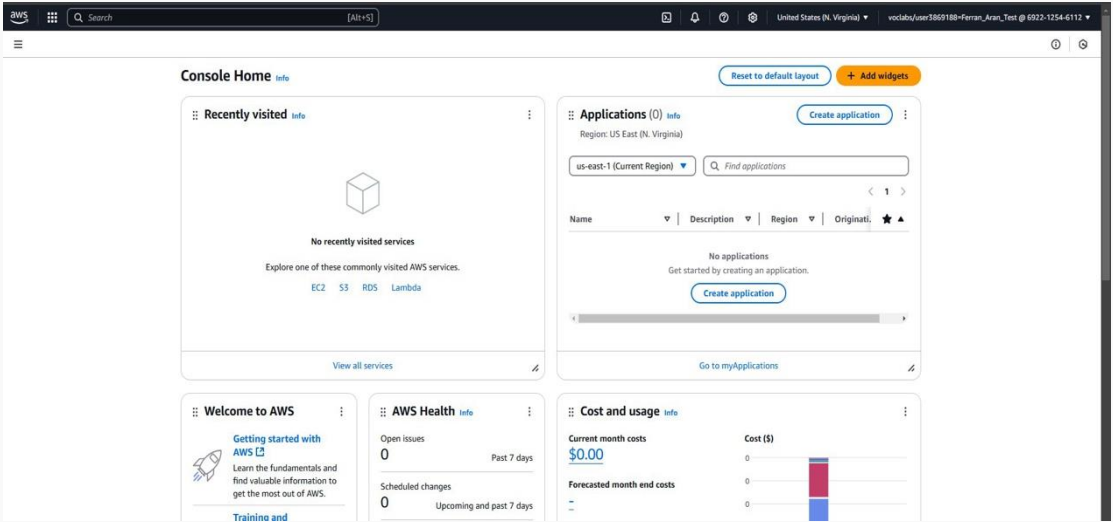


Figure 21: We are now presented with the dashboard

EC2 - Deploying a Jupyter Notebook

Creating an SSH Key Pair

Open a terminal or a powershell and type the following command:

```
mkdir .ssh
ssh-keygen -t rsa -f .ssh/aws-keypair
```

The first command might throw an error if the `.ssh` directory already exists. You can ignore it.

The `-t` option specifies the type of key to create:

- `rsa`
- `dsa`
- `ecdsa`
- `ed25519`

The `-f` option specifies the filename of the key file.

Creating an SSH Key Pair

The command will prompt you to enter a file in which to save the key. The command will also prompt you to enter a passphrase. You can enter a passphrase or leave the passphrase empty. This command will create a public and a private key in the default location:

- Public key: `.ssh/aws-keypair.pub`
- Private key: `.ssh/aws-keypair`

Importing our generated key pair

AWS provides a key pair to connect to the EC2 instance. However, we can use our key pair.

1. Go to search and write Key Pairs.
2. Click on Key Pairs.
3. Click on Actions and then on Import Key Pair.
4. Fill the form with the following settings:
   - Name: aws-keypair
   - Browse and select the public key file we created before. (.ssh/aws-keypair.pub)
   - Another option is to paste the public key in the Public key contents field. Use command `cat .ssh/aws-keypair.pub` to get the public key.
5. Import the key pair.

Creating an EC2 instance

1. Click on the Services and then on EC2.
2. Launch an instance.
3. Fill the form with the following settings:
    - Name: Jupyter Notebook
    - Image: Amazon Linux 2 AMI (HVM) - SSD Volume Type
    - Architecture: 64-bit (x86)
    - Type: t2.micro
    - Key pair: use the key pair created before. (aws-keypair)
    - Network: default VPC

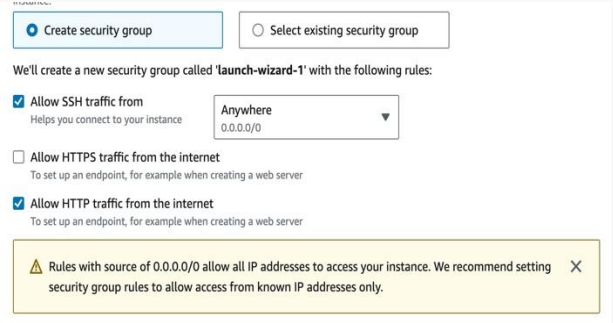
Security Group

This instance requires a security group that allows traffic on port 22 (SSH) and port 80 (HTTP).

- SSH is used to connect and manage the instance.
- HTTP is used to access the Jupyter Notebook from the browser.

Mark the checkbox to create a new security group and fill the form with the following settings:

- Allow SSH from anywhere
- Allow HTTP traffic from the internet



Connecting to the instance

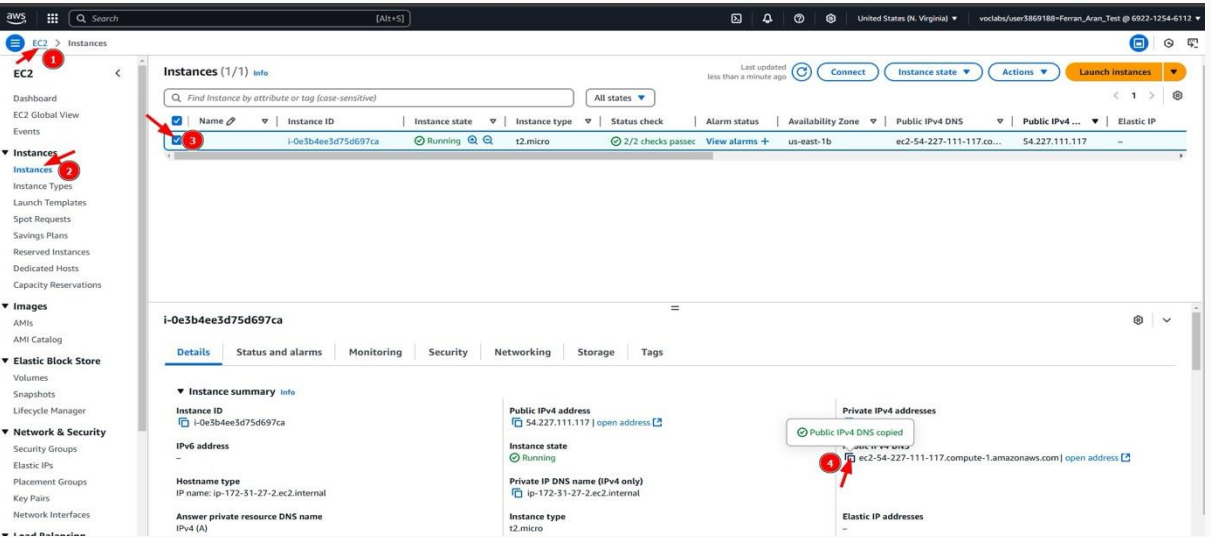


Figure 22: EC2 instance created. Getting the public DNS of your EC2 instance

Connecting to the instance

1. Open a terminal (mac,linux) or a powershell (windows).
2. Use the following command to connect to the instance. Replace the public DNS with the public DNS of your instance.

- **aws-keypair** is the name of to the private key file. We need to introduce the full path to the file (for example `.ssh/aws-keypair`).
- **ec2-user** is the default user for the Amazon Linux EC2 machine. Don't worry about it.
- **ec2-3-87-76-117.compute-1.amazonaws.com** is the public DNS of the instance. This is going to change everytime you restart your lab and each EC2 will have its own. So everytime you want to connect to the instance you will have to get the public DNS from the AWS console.

```
ssh -i .ssh/aws-keypair ec2-user@ec2-3-87-76-117.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Managing Python environments with uv

Why use uv?

- Python version management: Easily switch between different versions.
- Dependency isolation: Keep projects independent.
- Reproducibility: Ensures that dependencies remain consistent.

Installing uv

For more information about uv I encourage you to read the official documentation.

To install it, run the following on your EC2 instance:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

If it worked, you should see the following message on your terminal:

```
everything's installed!
```

Creating two example uv Projects

We will create two environments with different Python versions and dependencies.

Project 1: Python 3.8 + Jupyter

```
mkdir project1
cd project1
uv venv --seed --python 3.8 .project1-venv
source .project1-venv/bin/activate
pip install jupyter
deactivate
cd ..
```

Creating two example uv Projects

We will create two environments with different Python versions and dependencies.

Project 1: Python 3.8 + Jupyter

```
mkdir project1
cd project1
uv venv --seed --python 3.8 .project1-venv
source .project1-venv/bin/activate
pip install jupyter
deactivate
cd ..
```

Project 2: Python 3.10 + Jupyter + Pandas

```
mkdir project2
cd project2
uv venv --seed --python 3.10 .project2-venv
source .project2-venv/bin/activate
pip install jupyter pandas
deactivate
cd ..
```

Working with the environments

When I want to work with project1:

```
cd project1
source .project1-venv/bin/activate
```

Now when I run `python`, it will use the Python 3.8 version and when using `pip`, it will install packages in the project1 environment.

Installed packages will be stored and the next time I activate the environment, they will be available.

```
python --version
# Output
Python 3.8.20
```

When I am finished working with project1:

```
deactivate
cd ..
```

Working with the environments

When I want to work with project2:

```
cd project2
source .project2-venv/bin/activate
```

Now when I run `python`, it will use the Python 3.10 version and when using `pip`, it will install packages in the project2 environment.

When I am finished working with project2:

```
deactivate
cd ..
```

Running Jupyter Notebook (I)

We will first need one of the environments activated. For example, project1.

```
cd project1
source .project1-venv/bin/activate
```

Running Jupyter Notebook (I)

We can now run the Jupyter Notebook server. There are two ways to run the server:

- Running on localhost (default): Not accessible from the internet.

```
jupyter notebook
```

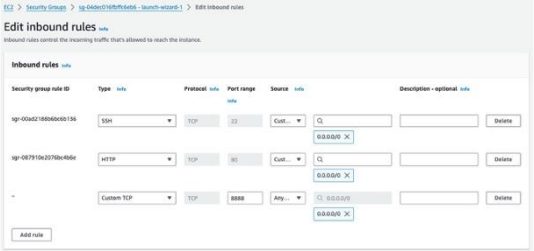
- Running on the public IP: Accessible from the internet.

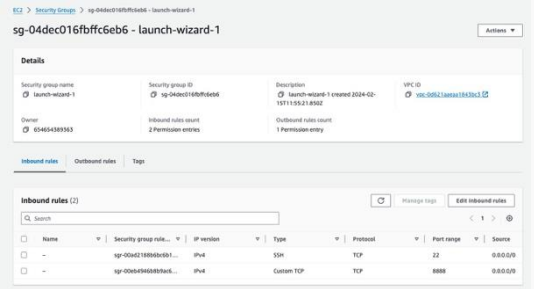
```
jupyter notebook --ip=? --port=?
```

- **ip**:
  - 0.0.0.0 (default): Listen on all IP addresses.
  - Public IP: Obtain the public IP from the instance and use it.
- **port**:
  - 8888 (default): The default port for the Jupyter Notebook. **This port is not opened by the security group.**

Opening the port 8888

1. Search for Security Groups (EC2). Click on **launch-wizard-1**.
2. Edit the inbound rules and add a new rule with the following settings:
   - Type: Custom TCP
   - Port Range: 8888
   - Source: Anywhere (0.0.0.0/0)
3. Delete the old rule for HTTP (port 80).
4. Save the changes.

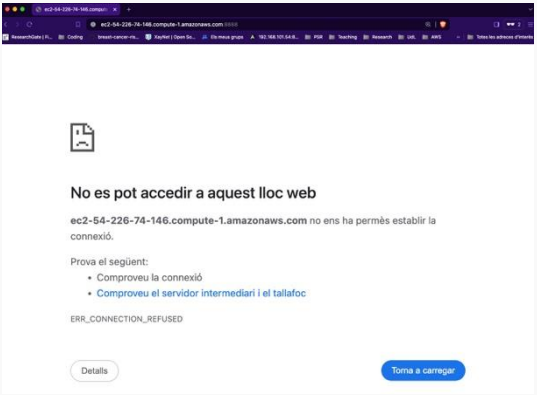




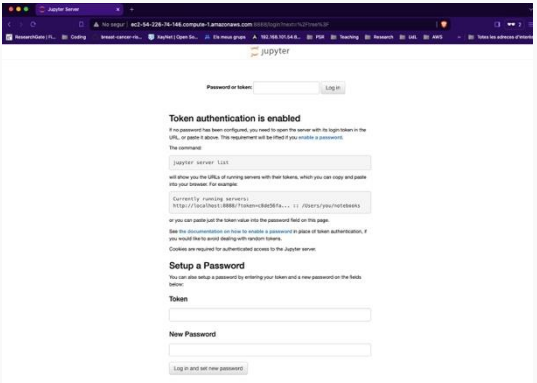
Running Jupyter Notebook (II)

```
jupyter notebook --ip=0.0.0.0 --port=8888
# In the browser
http://ec2-3-87-76-117.compute-1.amazonaws.com:8888
```

Before running the command

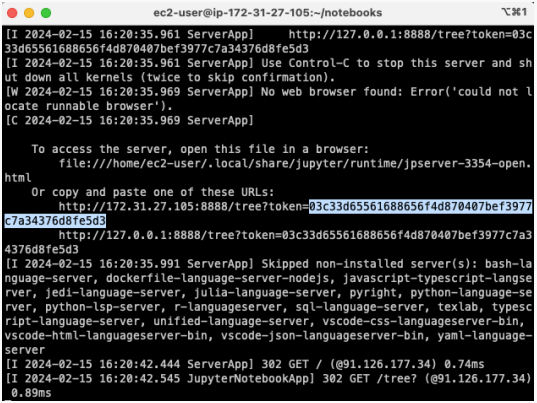


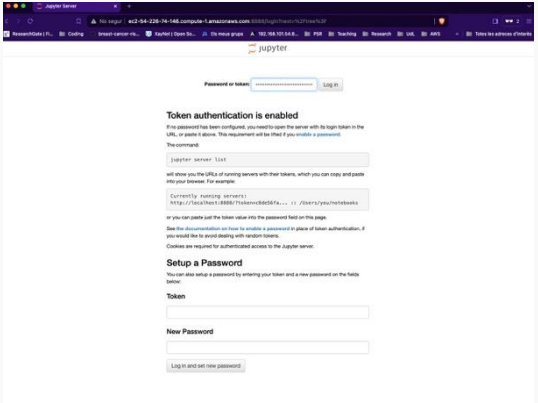
After running the command



Accessing the Jupyter Notebook

Copy the token from the terminal. Then, log in to the Jupyter Notebook from the browser. (If you have trouble copying the token, instead of `Ctrl+C`, use `Ctrl+Shift+C` on the terminal).





Accessing the Jupyter Notebook

Once on the Jupyter Notebook, to create a new notebook do the following:

1. Click on File → New → Notebook.
2. You will be prompted to choose the kernel. **Be careful**, the default one is going to fail, you have to change it to `Python 3` as shown below. If the only available option is `Python 3` that is fine, leave it as it is (we saw during class that in my demonstration there was only one option).

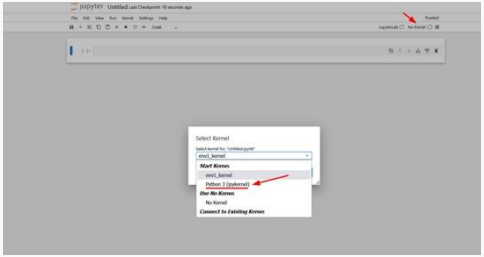


Figure 23: How to choose the right kernel

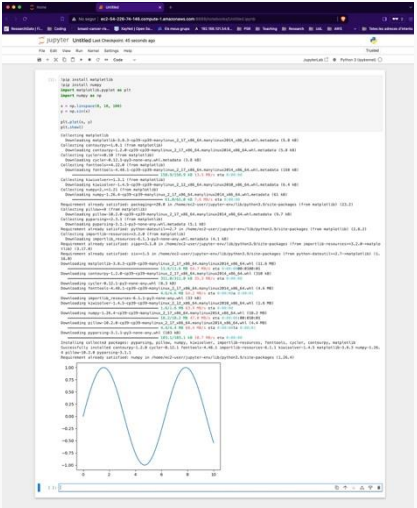
Running python code in the notebook

Click on New and then on Python 3. Write the following code and run it.

```
!pip install matplotlib
!pip install numpy
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)

plt.plot(x, y)
plt.show()
```



Commands to remember

- `uv venv --seed --python 3.8 .project1-venv`: Create a Python 3.8 environment named `.project1-venv`.
- `source .project1-venv/bin/activate`: Activate the `.project1-venv` environment. (You have to be on the same directory as the environment).
- `pip install jupyter`: Install Jupyter Notebook in the current environment. (You have to activate the environment first).
- `pip install pandas`: Install Pandas in the current environment.
- `pip install numpy==1.21.0`: Install a specific version of a package. (In this case, Numpy 1.21.0).
- `jupyter notebook --port=8888 --ip=0.0.0.0`: Run Jupyter Notebook on port 8888 and listen on all IP addresses.
- `deactivate`: Deactivate the current environment.

Commands to remember

- `mkdir project1`: Create a directory named `project1`.
- `cd project1`: Move to the `project1` directory.
- `cd ..`: Move to the parent directory.
- `cat .ssh/aws-keypair.pub`: Show the contents of file located at `.ssh/aws-keypair.pub`.

Conclusion

RECAP: Summary of the session

1. We have learned the about services in AWS, specifically EC2.
2. We have learned how to deploy a Jupyter Notebook in EC2 and manage Python environments with uv.
3. We have learned about security groups, key pairs, and how to connect to the instance using SSH.

Why use Jupyter on the Cloud?

Example Use Cases

- Large Datasets: If a dataset is too big for a local machine (e.g., 1000GB), cloud storage and compute power are essential.
- High Computational Requirements: Some models require GPUs or high-memory machines, which can be rented via AWS.
- Collaboration: I can give access to others to my Jupyter Notebook with the same environment and data.