

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

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

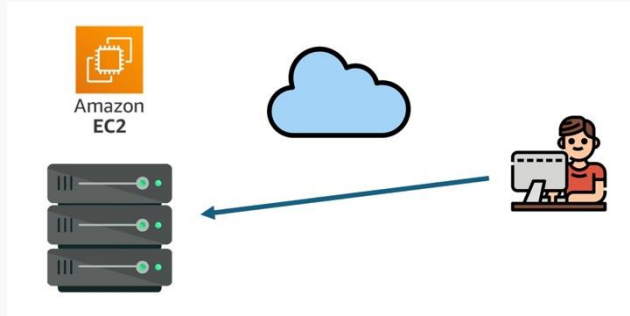


Figure 6: AWS EC2

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?

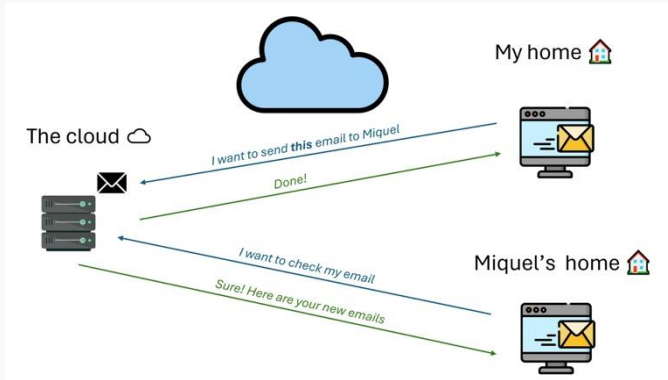


Figure 8: Client-server model when sending emails

Why do I need to *open ports*?

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).

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.

Introduction to AWS Academy

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

Course Invitation

AA AWS Academy <notifications@instructure.com>
To Aran Domingo, Ferran

☺ Reply Reply All Forward 📧 ⋮

Tue 2/25/2025 10:10 AM

🔗 If there are problems with how this message is displayed, click here to view it in a web browser.

You've been invited to participate in a class at AWS Academy . The class is called AWS Academy Learner Lab [111255]. Course role: Student

Name: **Ferran Aran Test**
Email: ferran.aran@gft.com
Username: **none**

You'll need to register with Canvas before you can participate in the class.

[Get Started](#)



Figure 10: Accepting the invite

Activating your account

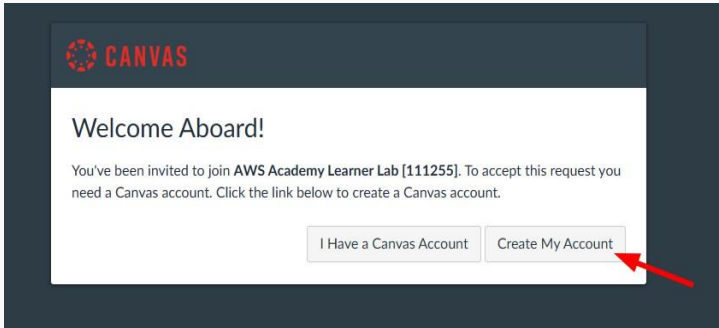


Figure 11: Creating your account

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.

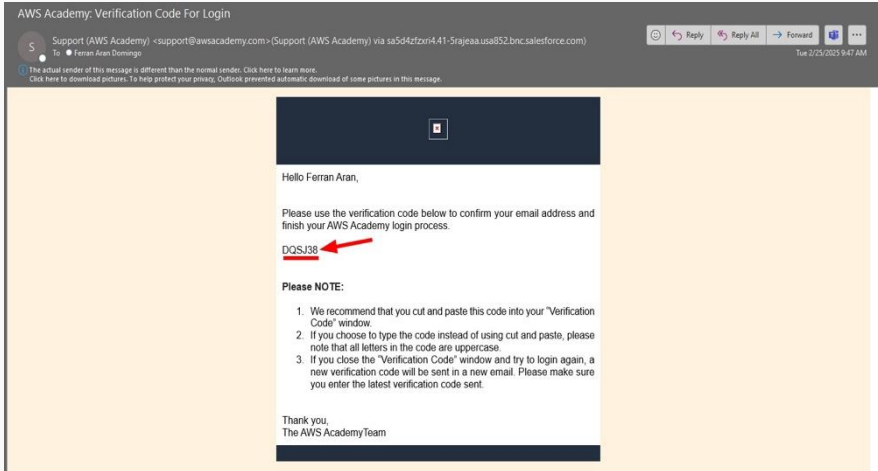


Figure 13: Enter verification code

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).

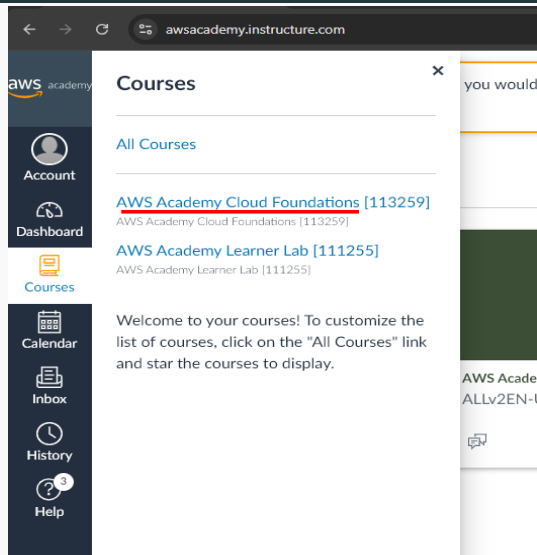


Figure 14: Available courses

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.

- 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.

Accessing the Learner Lab

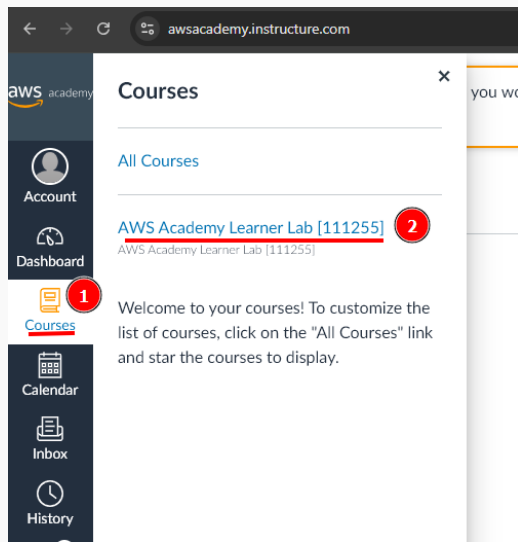


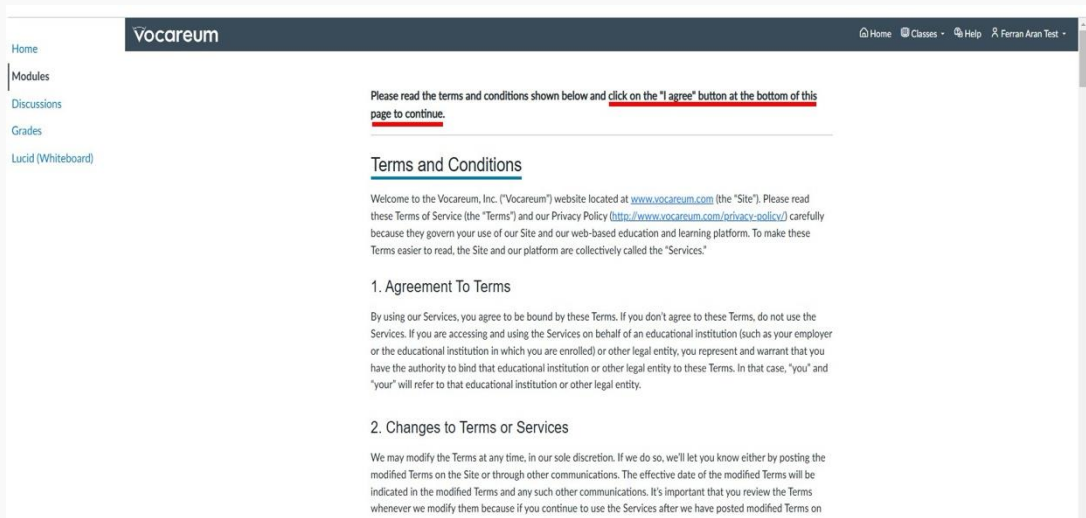
Figure 15: Choosing the learner lab course

Accessing the Learner Lab

The screenshot shows the AWS Academy Learner Lab interface. The browser address bar displays `awsacademy.instructure.com/courses/111255/modules`. The page title is `ALLv2EN-US-LTI113-111255 > Modules`. On the left sidebar, the `Modules` link is highlighted with a red circle containing the number `1`. The main content area lists several modules:

- `Course Welcome and Overview`
- `Pre-Course Survey`
- `AWS Academy Learner Lab Student Guide`
- `AWS Academy Learner Lab Compliance and Security`
- `Learn how to effectively use the AWS Academy Learner Lab`
- `Module Knowledge Check` (100 pts, Score at least 70.0)
- `AWS Academy Learner Lab`
- `Launch AWS Academy Learner Lab` (highlighted with a red circle containing the number `2`)

Figure 16: Choosing the learner lab module



The screenshot shows the Vocareum website interface. On the left is a navigation menu with links for Home, Modules, Discussions, Grades, and Lucid (Whiteboard). The top navigation bar includes Home, Classes, Help, and a user profile for Ferran Aran Test. The main content area displays a message: "Please read the terms and conditions shown below and click on the 'I agree' button at the bottom of this page to continue." Below this is a section titled "Terms and Conditions" with a blue underline. The text reads: "Welcome to the Vocareum, Inc. ('Vocareum') website located at www.vocareum.com (the 'Site'). Please read these Terms of Service (the 'Terms') and our Privacy Policy (<http://www.vocareum.com/privacy-policy/>) carefully because they govern your use of our Site and our web-based education and learning platform. To make these Terms easier to read, the Site and our platform are collectively called the 'Services.'" The page lists two sections: "1. Agreement To Terms" and "2. Changes to Terms or Services".

Home Classes Help Ferran Aran Test

Vocareum

Home Modules Discussions Grades Lucid (Whiteboard)

Please read the terms and conditions shown below and click on the "I agree" button at the bottom of this page to continue.

Terms and Conditions

Welcome to the Vocareum, Inc. ("Vocareum") website located at www.vocareum.com (the "Site"). Please read these Terms of Service (the "Terms") and our Privacy Policy (<http://www.vocareum.com/privacy-policy/>) carefully because they govern your use of our Site and our web-based education and learning platform. To make these Terms easier to read, the Site and our platform are collectively called the "Services."

1. Agreement To Terms

By using our Services, you agree to be bound by these Terms. If you don't agree to these Terms, do not use the Services. If you are accessing and using the Services on behalf of an educational institution (such as your employer or the educational institution in which you are enrolled) or other legal entity, you represent and warrant that you have the authority to bind that educational institution or other legal entity to these Terms. In that case, "you" and "your" will refer to that educational institution or other legal entity.

2. Changes to Terms or Services

We may modify the Terms at any time, in our sole discretion. If we do so, we'll let you know either by posting the modified Terms on the Site or through other communications. The effective date of the modified Terms will be indicated in the modified Terms and any such other communications. It's important that you review the Terms whenever we modify them because if you continue to use the Services after we have posted modified Terms on

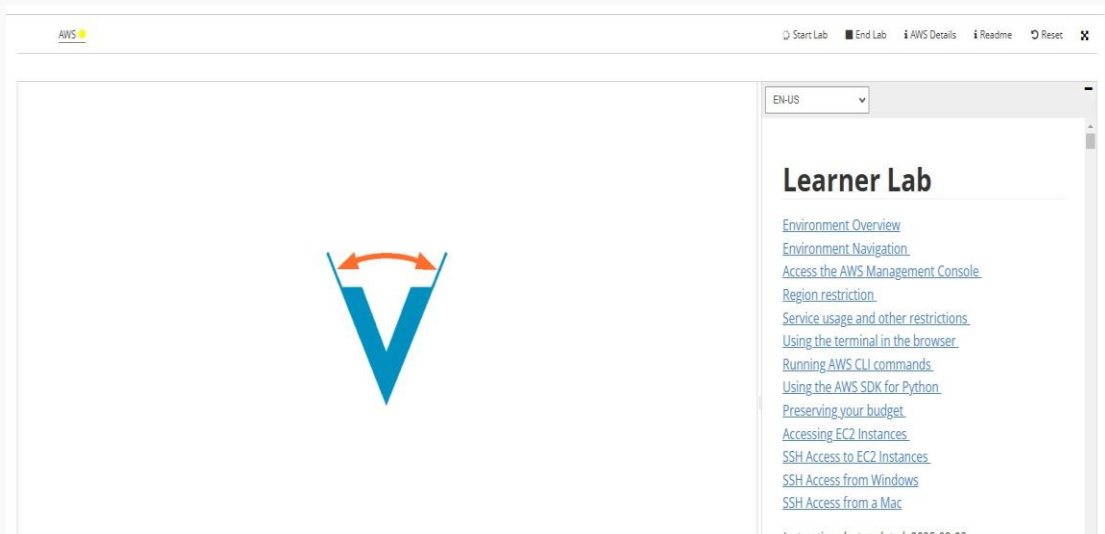
Figure 17: Scroll down and agree the terms and conditions

Accessing the Learner Lab

The screenshot displays the AWS Academy Learner Lab interface. At the top, the breadcrumb navigation reads: ALLv2EN-US-LTI13-111255 > Modules > AWS Academy Learner Lab > Launch AWS Academy Learner Lab. Below this, a navigation bar contains the text 'AWS' with a red dot, followed by a 'Start Lab' button with a right-pointing arrow, 'End Lab', 'AWS Details', 'Readme', 'Reset', and a close icon. A red arrow points to the 'Start Lab' button. On the left side, a sidebar menu lists 'Home', 'Modules', 'Discussions', 'Grades', and 'Lucid (Whiteboard)'. The main content area is split into two panes. The left pane is a terminal window with the prompt 'eee_h_4181718@runweb1627061--\$'. The right pane is a document titled 'Learner Lab' with a dropdown menu set to 'EN-US'. The document contains a list of links: 'Environment Overview', 'Environment Navigation', 'Access the AWS Management Console', 'Region restriction', 'Service usage and other restrictions', 'Using the terminal in the browser', 'Running AWS CLI commands', 'Using the AWS SDK for Python', 'Preserving your budget', 'Accessing EC2 Instances', 'SSH Access to EC2 Instances', 'SSH Access from Windows', and 'SSH Access from a Mac'. Below the links, it states 'Instructions last updated: 2025-02-03' and 'Environment Overview'. The text 'This Learner Lab provides a sandbox environment for' is partially visible at the bottom.

Figure 18: Click on 'Start'

Accessing the Learner Lab



The screenshot displays the AWS Learner Lab interface. At the top left, the AWS logo is visible. The top right contains navigation buttons: Start Lab, End Lab, AWS Details, Readme, and Reset. The main content area is mostly blank, featuring a large blue 'V' logo with an orange double-headed arrow above it, indicating a loading or waiting state. On the right side, there is a sidebar with a dropdown menu set to 'EN-US'. Below the dropdown, the title 'Learner Lab' is displayed. A list of navigation links follows:

- [Environment Overview](#)
- [Environment Navigation](#)
- [Access the AWS Management Console](#)
- [Region restriction](#)
- [Service usage and other restrictions](#)
- [Using the terminal in the browser](#)
- [Running AWS CLI commands](#)
- [Using the AWS SDK for Python](#)
- [Preserving your budget](#)
- [Accessing EC2 Instances](#)
- [SSH Access to EC2 Instances](#)
- [SSH Access from Windows](#)
- [SSH Access from a Mac](#)

Figure 19: Wait for the environment to set up

Accessing the Learner Lab

ALLv2EN-US-LTI13-111255 > Modules > AWS Academy Learner Lab > Launch AWS Academy Learner Lab

AWS Used \$0 of \$50 03:55 ▶ Start Lab ■ End Lab ⓘ AWS Details ⓘ Readme ↺ Reset ✕

Home

Modules

Discussions

Grades

Lucid (Whiteboard)

```
eee_t_4181718@runweb162706:~$
```

Learner Lab

- [Environment Overview](#)
- [Environment Navigation](#)
- [Access the AWS Management Console](#)
- [Region restriction](#)
- [Service usage and other restrictions](#)
- [Using the terminal in the browser](#)
- [Running AWS CLI commands](#)

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

Accessing the Learner Lab

The screenshot shows the AWS Management Console Home dashboard. At the top, there is a search bar and navigation icons. The main content area is divided into several widgets:

- Recently visited**: A widget showing no recently visited services. It includes a 3D cube icon and a list of commonly visited services: EC2, S3, RDS, and Lambda. A "View all services" link is at the bottom.
- Applications (0)**: A widget for managing applications. It shows the current region as "us-east-1 (Current Region)" and a search bar. Below is a table header with columns: Name, Description, Region, and Origin. The table is empty, displaying "No applications" and a "Create application" button. A "Go to myApplications" link is at the bottom.
- Welcome to AWS**: A widget with a rocket icon and a link to "Getting started with AWS". It includes text about learning fundamentals and a "Training and" link.
- AWS Health**: A widget showing "Open issues" as 0 (Past 7 days) and "Scheduled changes" as 0 (Upcoming and past 7 days).
- Cost and usage**: A widget showing "Current month costs" as \$0.00 and "Forecasted month end costs" as -. It includes a small bar chart with a red bar and a blue bar.

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

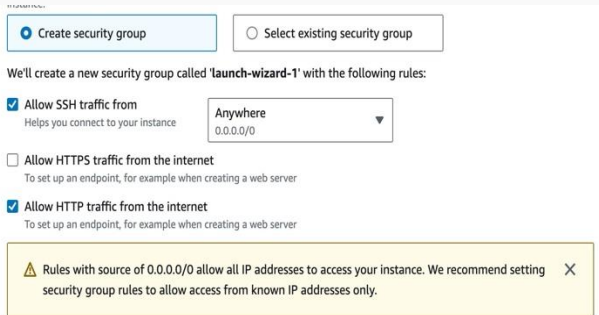
Security Group

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



The screenshot shows the AWS console interface for creating a security group. At the top, there are two radio buttons: "Create security group" (selected) and "Select existing security group". Below this, a message states: "We'll create a new security group called 'launch-wizard-1' with the following rules:". There are three rule options, each with a checkbox and a description:

- Allow SSH traffic from**
Helps you connect to your instance
- Allow HTTPS traffic from the internet**
To set up an endpoint, for example when creating a web server
- Allow HTTP traffic from the internet**
To set up an endpoint, for example when creating a web server

The "Allow SSH traffic from" rule has a dropdown menu set to "Anywhere" with the IP address "0.0.0.0/0" displayed below it. At the bottom, a yellow warning box contains the text: "Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." with a close button (X) on the right.

Connecting to the instance

The screenshot shows the AWS Management Console interface for EC2 instances. The top navigation bar includes the AWS logo, a search bar, and the user's profile information. The left sidebar contains navigation options for EC2, including Dashboard, Global View, Events, Instances, Images, Elastic Block Store, and Network & Security. The main content area displays a list of instances, with one instance selected and its details expanded.

Instances (1/1) info

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input checked="" type="checkbox"/>		i-0e3b4ee3d75d697ca	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-54-227-111-117.co...	54.227.111.117	-

Instance details for i-0e3b4ee3d75d697ca

Instance summary info

- Instance ID:** i-0e3b4ee3d75d697ca
- IPv6 address:** -
- Hostname type:** IP name: ip-172-31-27-2.ec2.internal
- Answer private resource DNS name:** IPv4 (A)
- Public IPv4 address:** 54.227.111.117 | open address
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-27-2.ec2.internal
- Instance type:** t2.micro
- Private IPv4 addresses:** ec2-54-227-111-117.compute-1.amazonaws.com | open address
- Elastic IP addresses:** -

A tooltip above the private IPv4 address indicates: **Public IPv4 DNS copied**. A red circle and arrow point to the public IPv4 DNS field in the instance details.

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 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.

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 ..
```

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.

EC2 > Security Groups > sg-04dec016fbffc6eb6 - launch-wizard-1 > Edit inbound rules

Edit inbound rules [info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [info](#)

Security group rule ID	Type info	Protocol info	Port range info	Source info	Description - optional info
sg-00ae218866e01136	SSH	TCP	22	0.0.0.0/0	
sg-087910e23780c486e	HTTP	TCP	80	0.0.0.0/0	
-	Custom TCP	TCP	8888	0.0.0.0/0	

[Add rule](#)

EC2 > Security Groups > sg-04dec016fbffc6eb6 - launch-wizard-1

sg-04dec016fbffc6eb6 - launch-wizard-1 [Actions](#)

Details

Security group name: launch-wizard-1
Security group ID: sg-04dec016fbffc6eb6
Description: launch-wizard-1 created 2024-02-15T11:55:21.850Z
VPC ID: vpc-9d821a6aa1843bc1

Owner: 654654389365
Inbound rules count: 2 Permission entries
Outbound rules count: 1 Permission entry

[Inbound rules](#) [Outbound rules](#) [Tags](#)

Inbound rules (2) [Manage tags](#) [Edit inbound rules](#)

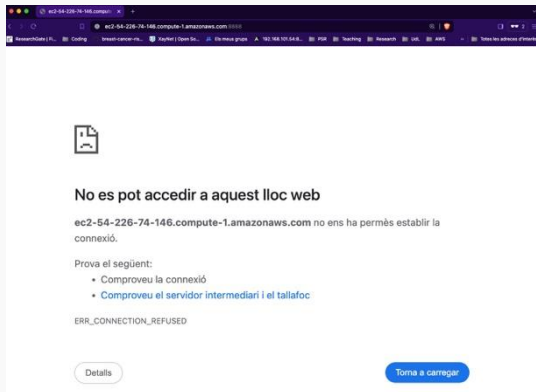
Q Search

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sg-00ae218866e01136...	IPv4	SSH	TCP	22	0.0.0.0/0
<input type="checkbox"/>	-	sg-00eb49668893ac6...	IPv4	Custom TCP	TCP	8888	0.0.0.0/0

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



No es pot accedir a aquest lloc web

ec2-54-226-74-146.compute-1.amazonaws.com no ens ha permès establir la connexió.

Prova el següent:

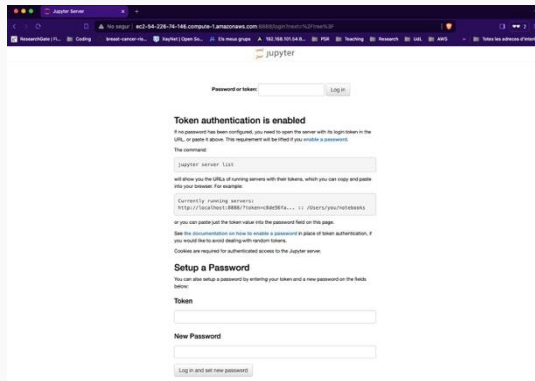
- Comproveu la connexió
- Comproveu el servidor intermediari i el tallafoc

ERR_CONNECTION_REFUSED

Details

Torna a carregar

After running the command



jupyter

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:8886/?token=08e56fa... //Users/you/notes
```

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

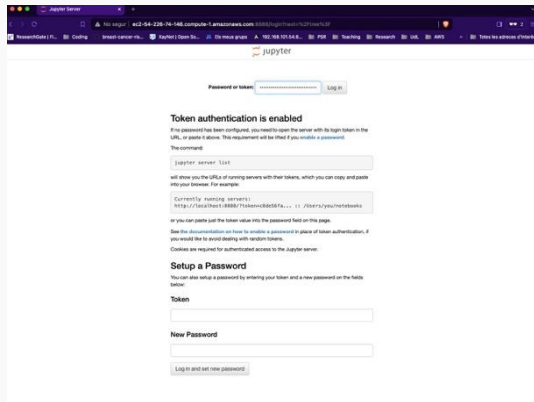
Log in and set new password

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).

```
ec2-user@ip-172-31-27-105:~/notebooks
[I 2024-02-15 16:20:35.961 ServerApp] http://127.0.0.1:8888/tree?token=03c33d65561688656f4d870407bef3977c7a34376d8fe5d3
[I 2024-02-15 16:20:35.961 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 2024-02-15 16:20:35.969 ServerApp] No web browser found: Error('could not locate runnable browser').
[C 2024-02-15 16:20:35.969 ServerApp]

To access the server, open this file in a browser:
file:///home/ec2-user/.local/share/jupyter/runtime/jpserver-3354-open.html
Or copy and paste one of these URLs:
http://172.31.27.105:8888/tree?token=03c33d65561688656f4d870407bef3977c7a34376d8fe5d3
http://127.0.0.1:8888/tree?token=03c33d65561688656f4d870407bef3977c7a34376d8fe5d3
[I 2024-02-15 16:20:35.991 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-languageserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yamllanguage-server
[I 2024-02-15 16:20:42.444 ServerApp] 302 GET / (@91.126.177.34) 0.74ms
[I 2024-02-15 16:20:42.545 JupyterNotebookApp] 302 GET /tree? (@91.126.177.34) 0.89ms
```



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).

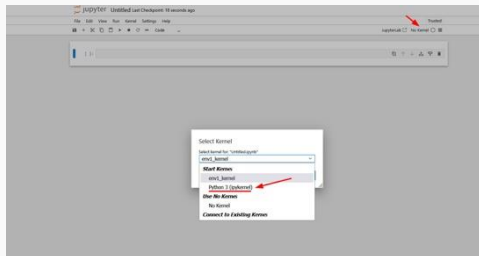


Figure 23: How to choose the right kernel

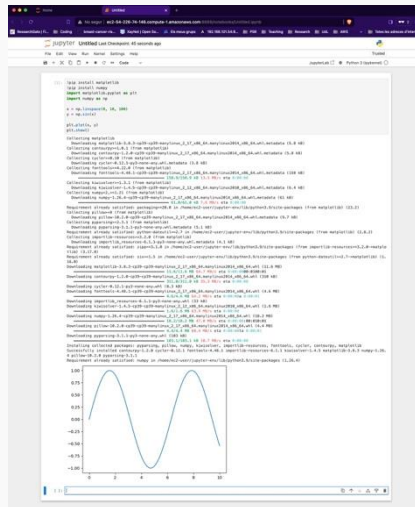
Running python code in the notebook

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.