EC2: Concepts and Practice!

What is EC2?

Amazon EC2 (Elastic Compute Cloud) is an AWS service that provides configurable virtual machines in the cloud. It enables you to select resources such as the operating system, CPU, memory, and storage. EC2 is commonly used to run servers, APIs, and various applications, offering SSH access and security managed through key pairs and security groups. Billing is based on actual usage time.

Security Groups

A Security Group in AWS is a virtual firewall that controls the network traffic allowed to and from your EC2 instances. It acts as a set of rules that define who can connect to your machine and where it can connect.

Inbound Rules

Inbound rules define which incoming connections are allowed to reach your instance. For example:

- Allowing port 22 (SSH) only from your IP address
- Allowing port 80 (HTTP) for public web access
- Permitting internal communication between instances

Outbound Rules

Outbound rules define which destinations your instance can access. By default, instances can access the internet, but this can be restricted for security purposes.

Why restricting SSH is critical?

Port 22 (SSH) provides full remote access to your server. If it is open to the world (`0.0.0.0/0`), anyone on the internet can attempt brute-force attacks or exploit vulnerabilities. Restricting SSH access to your own IP address or network is essential to:

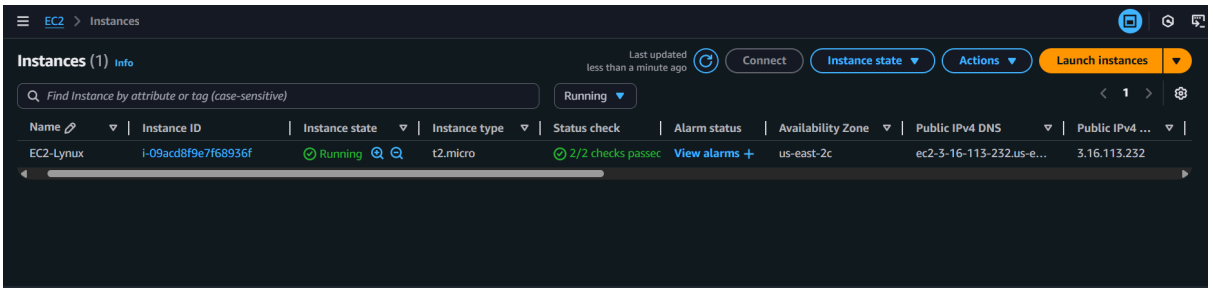
- Prevent unauthorized access
- Reduce exposure to automated attacks
- Ensure only trusted users can log in

The Principle of Least Privilege

The Principle of Least Privilege means giving users, applications, or systems only the permissions they need to perform their tasks. For example, a developer who only needs to read data from a database should not have permission to delete tables. Similarly, an EC2 instance that uploads files to S3 should only have `PutObject` permissions, not full access to the entire bucket. This minimizes the risk of accidents or security breaches by limiting what each component can do.

In practice!

My primary goal was to launch an EC2 instance and connect to it via SSH through the console.



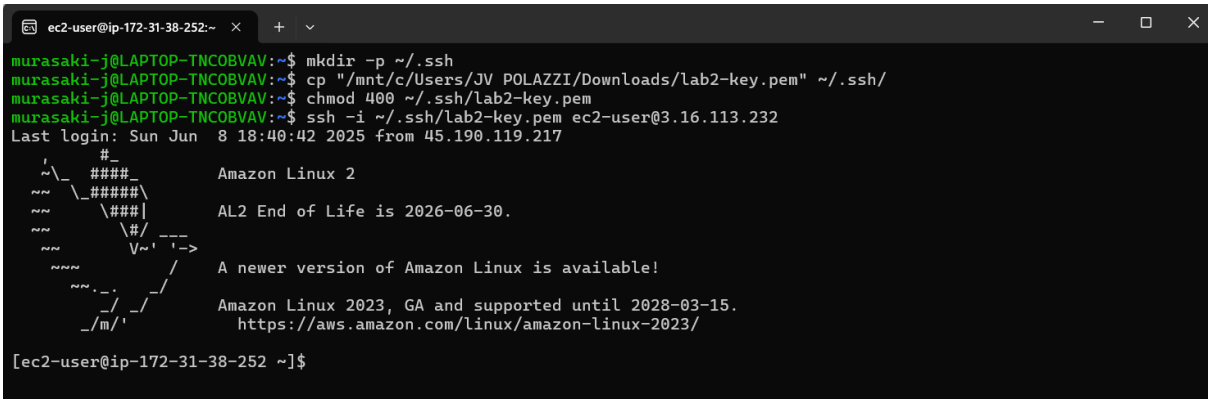
Public IP: 3.16.113.232

After launching the instance, and already inside the Ubuntu terminal, the first command I ran was `mkdir`, used to create the `.ssh` directory in my home folder. The `-p` flag was included to prevent an error if the directory already existed.

The SSH key had been downloaded to my local Windows machine. To use it on the Linux VM, I copied it from the Windows Downloads folder to the `.ssh` directory using the `cp` command.

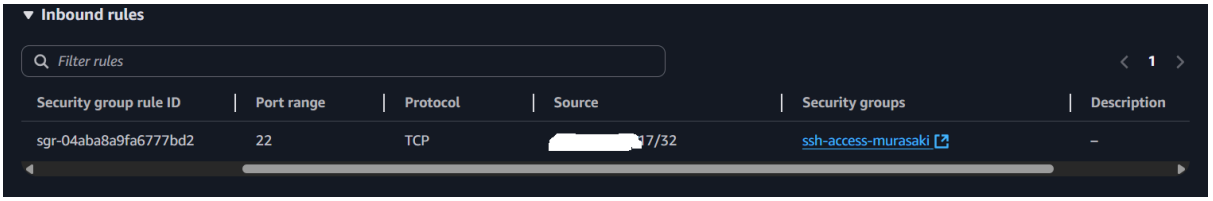
Before connecting to the EC2 instance via SSH, it was necessary to restrict the permissions of the private key. The `chmod` command was used to set the permissions so that only I could read the file. Without this step, SSH would refuse the connection for security reasons.

Finally, the `ssh -i` command was used to initiate a secure connection to my EC2 instance, using the private key for authentication. The `-i` flag specified the path to the `.pem` file containing the key required to connect securely to the server.

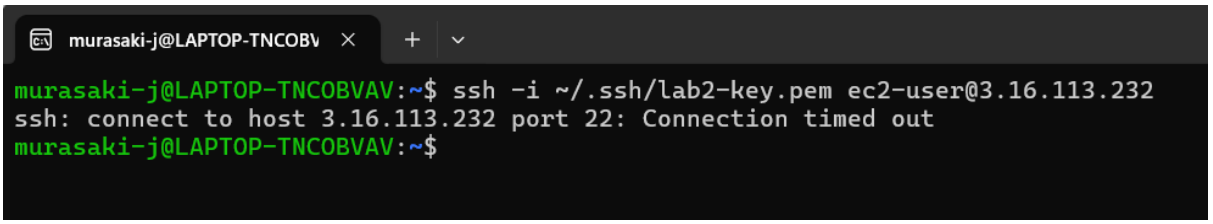


The connection was successful!

To enhance security, I created a custom security group for my EC2 instance that restricts SSH (port 22) access exclusively to my IP address, instead of allowing access from all sources.



To verify that the rule was working correctly, I then switched to a mobile data network, which uses a different IP address. As expected, the connection attempt failed, confirming that the security group was properly restricting access as intended.



Challenge Faced

The challenge arose when trying to use the `.pem` key, which had originally been downloaded on a Windows system, inside a Linux virtual machine. Since the Linux terminal didn't have direct access to the Windows Downloads folder, it was necessary to manually move the key using the `cp` command, copying it from the Windows path (`/mnt/c/...`) to the `~/.ssh/` directory in the Linux environment. Then, it was necessary to adjust the file permissions using `chmod 400`, as SSH requires the private key to have restricted permissions in order to be accepted. Only after completing these steps was it possible to successfully connect using `ssh -i`.