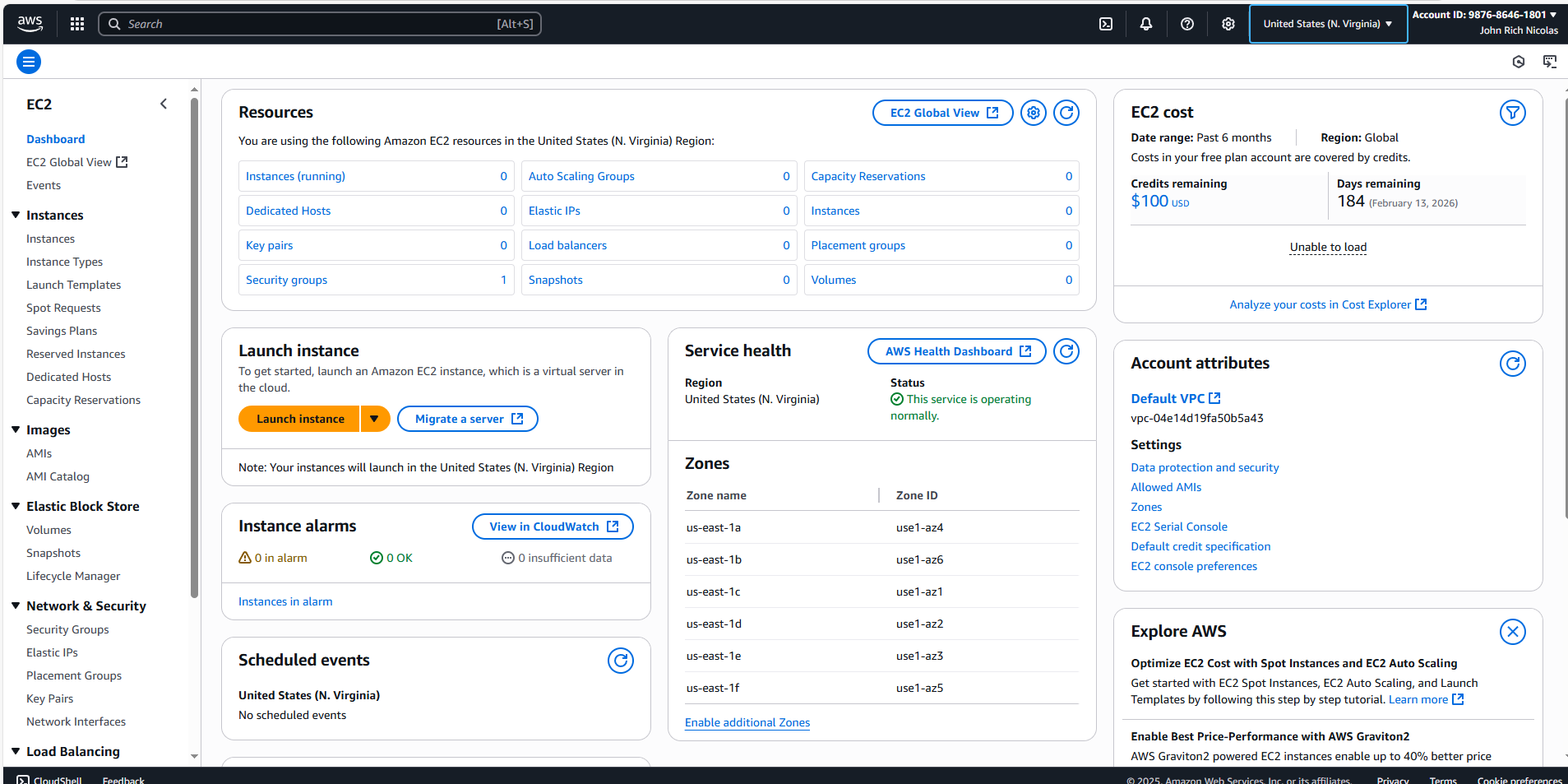
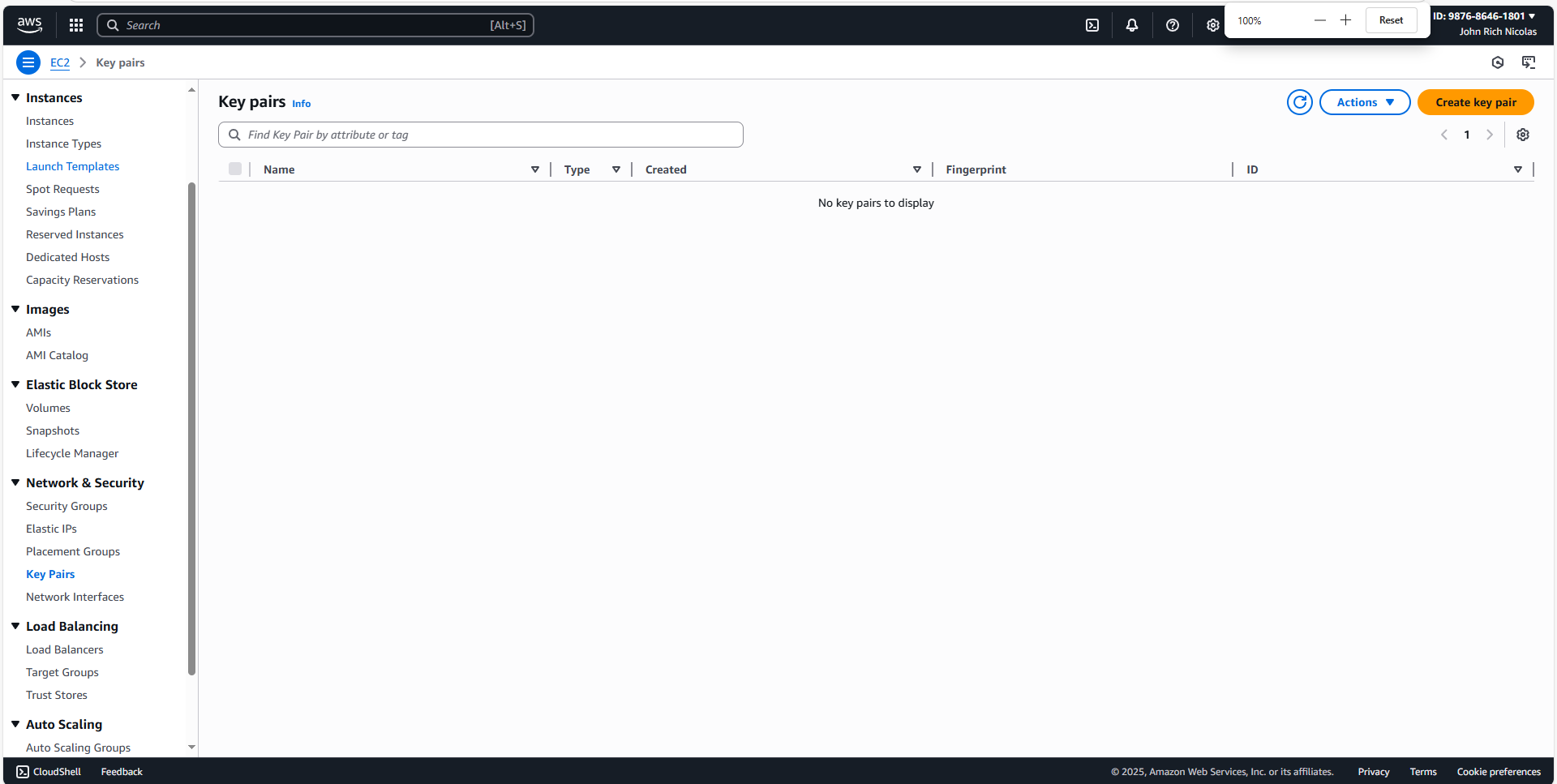
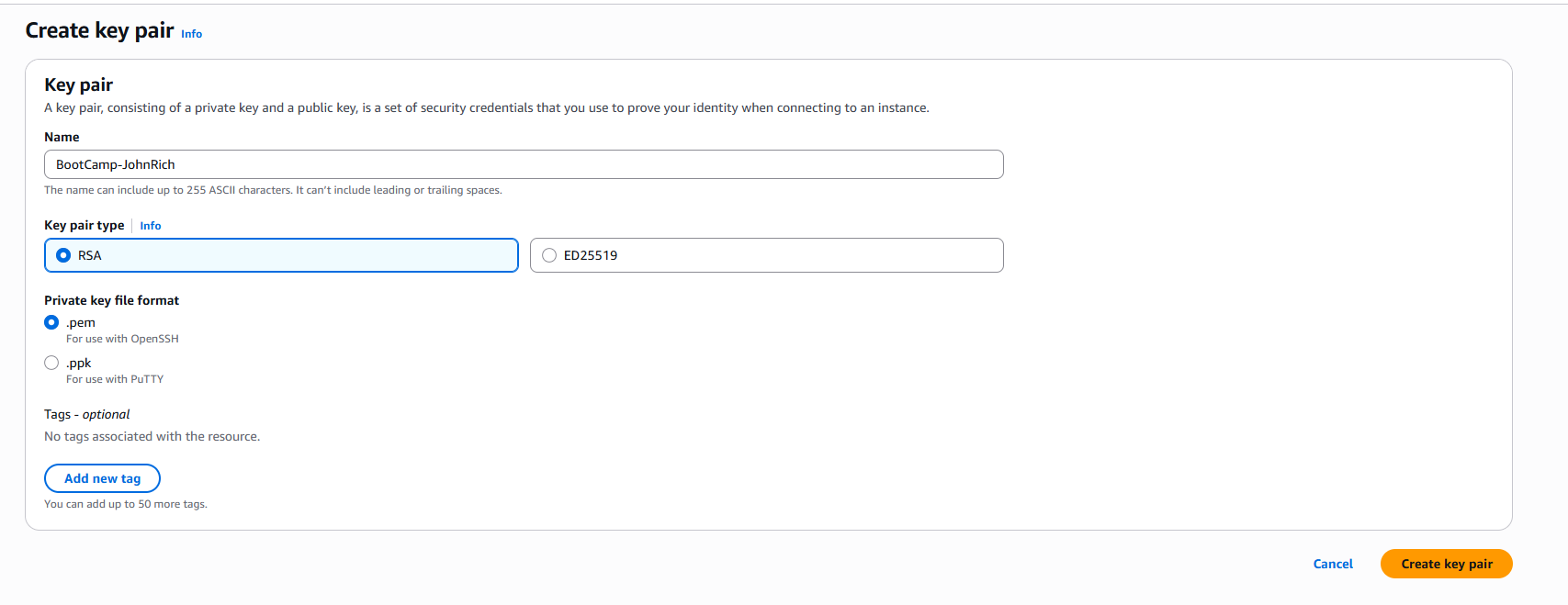
**CREATING A KEY PAIR**

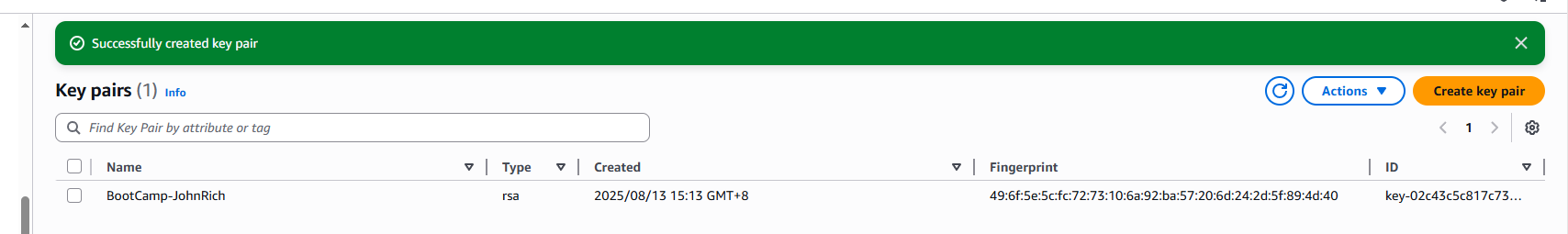
STEP 1 & STEP 2



STEP 3

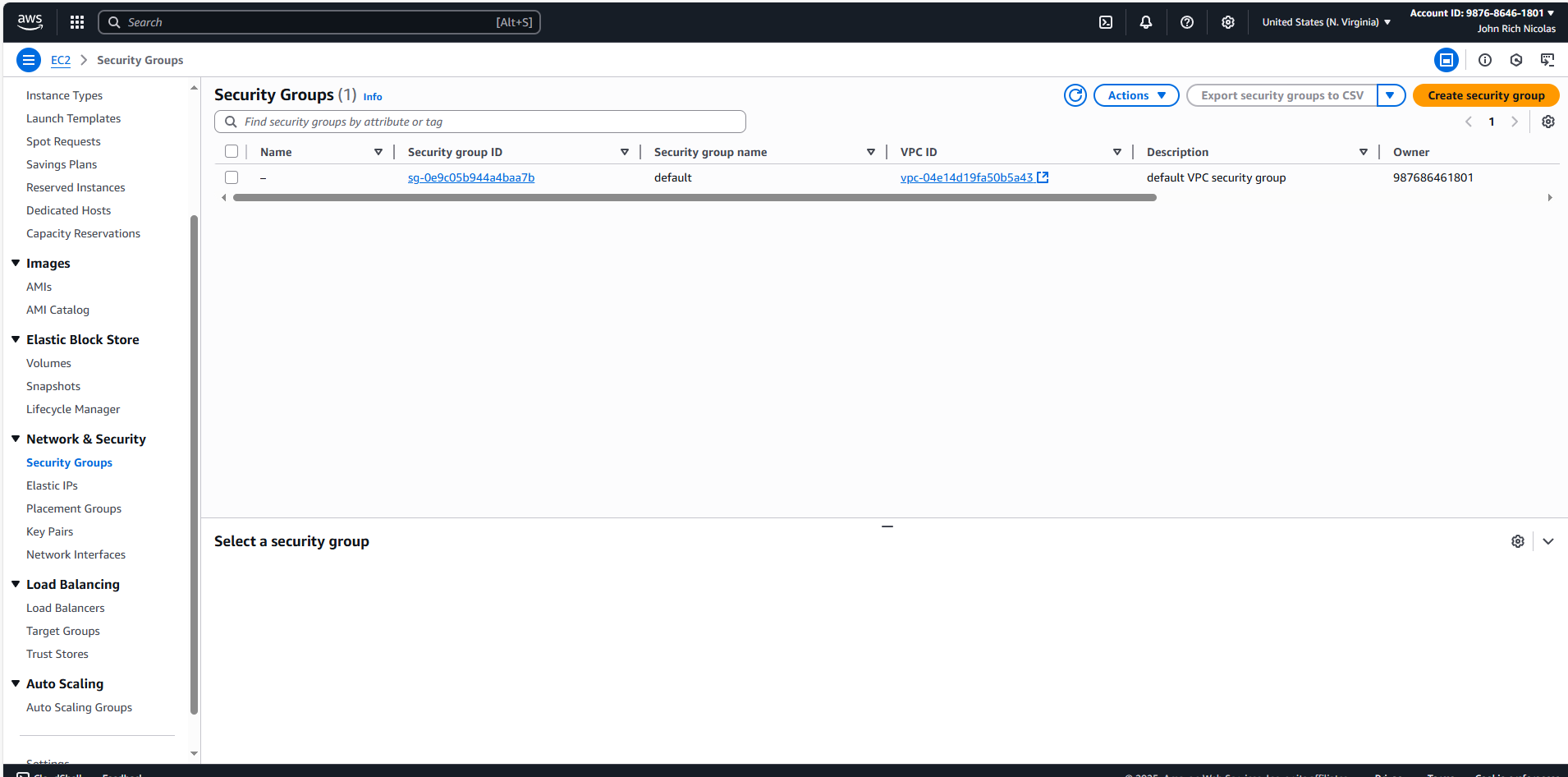
STEP 4 -8



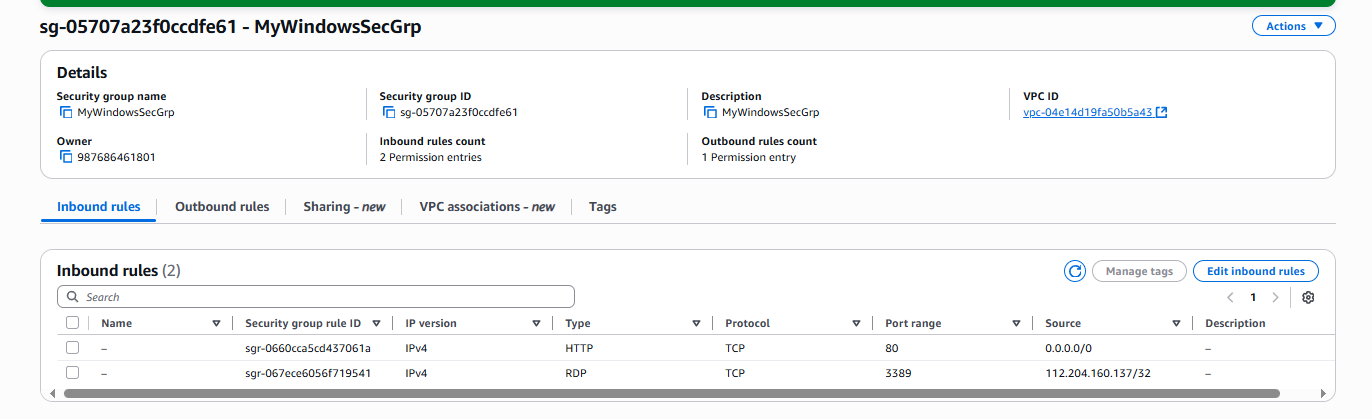
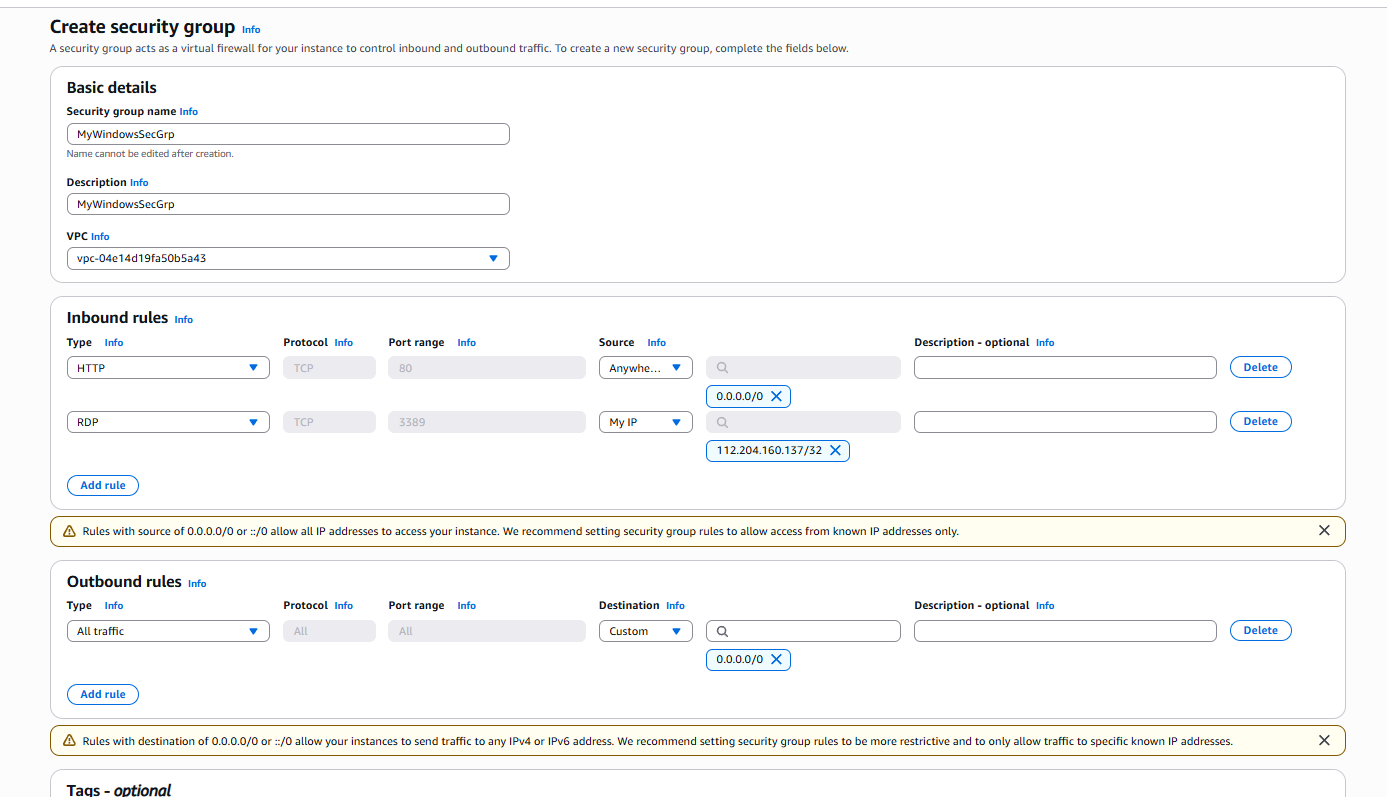


**Create a Security Group (Linux and Windows)**

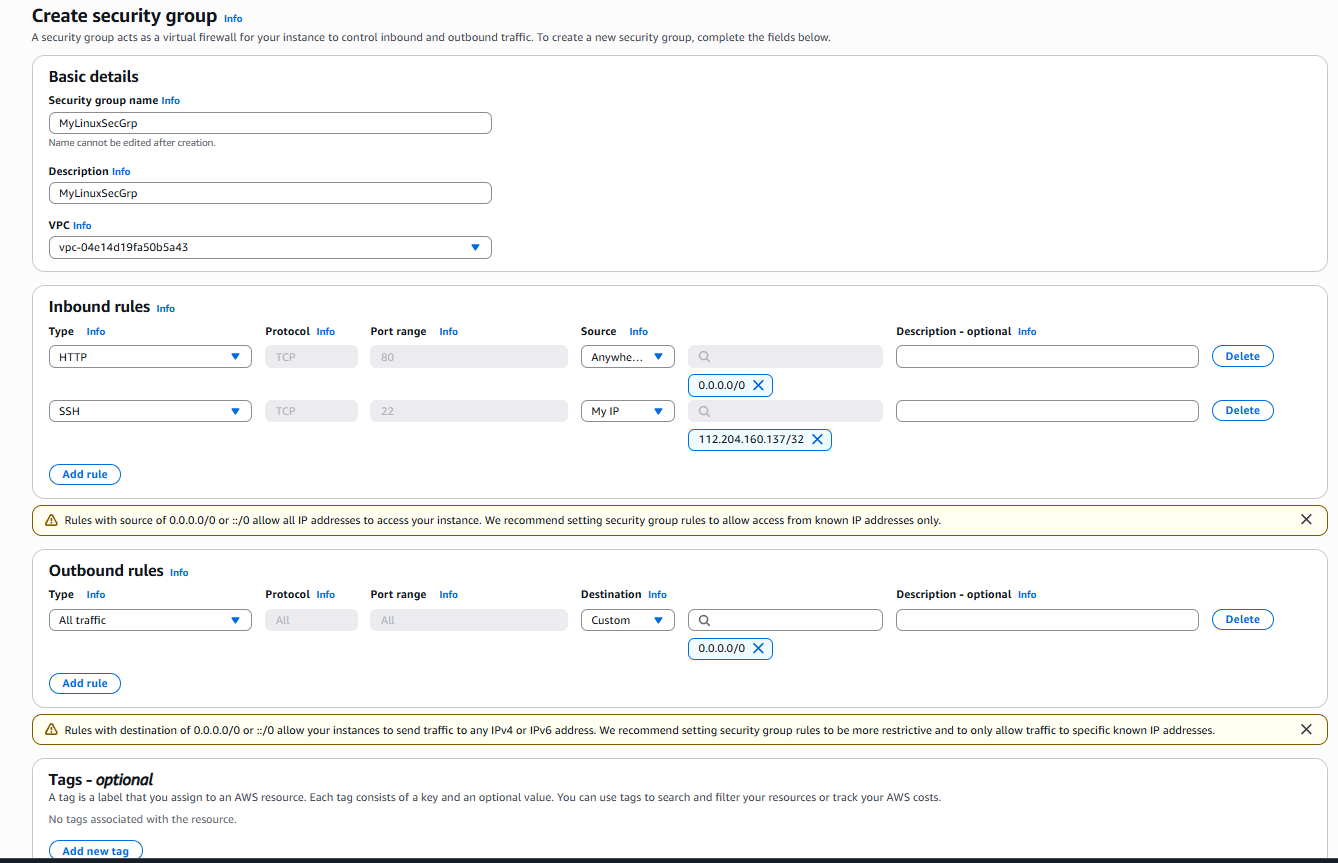
**STEP 1 – 4**

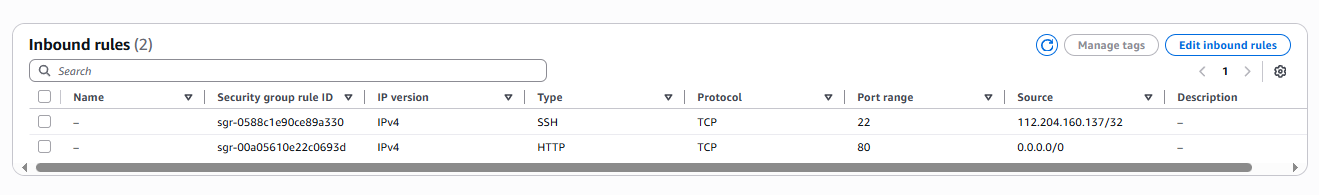
****

**RESULTS WINDOWS**

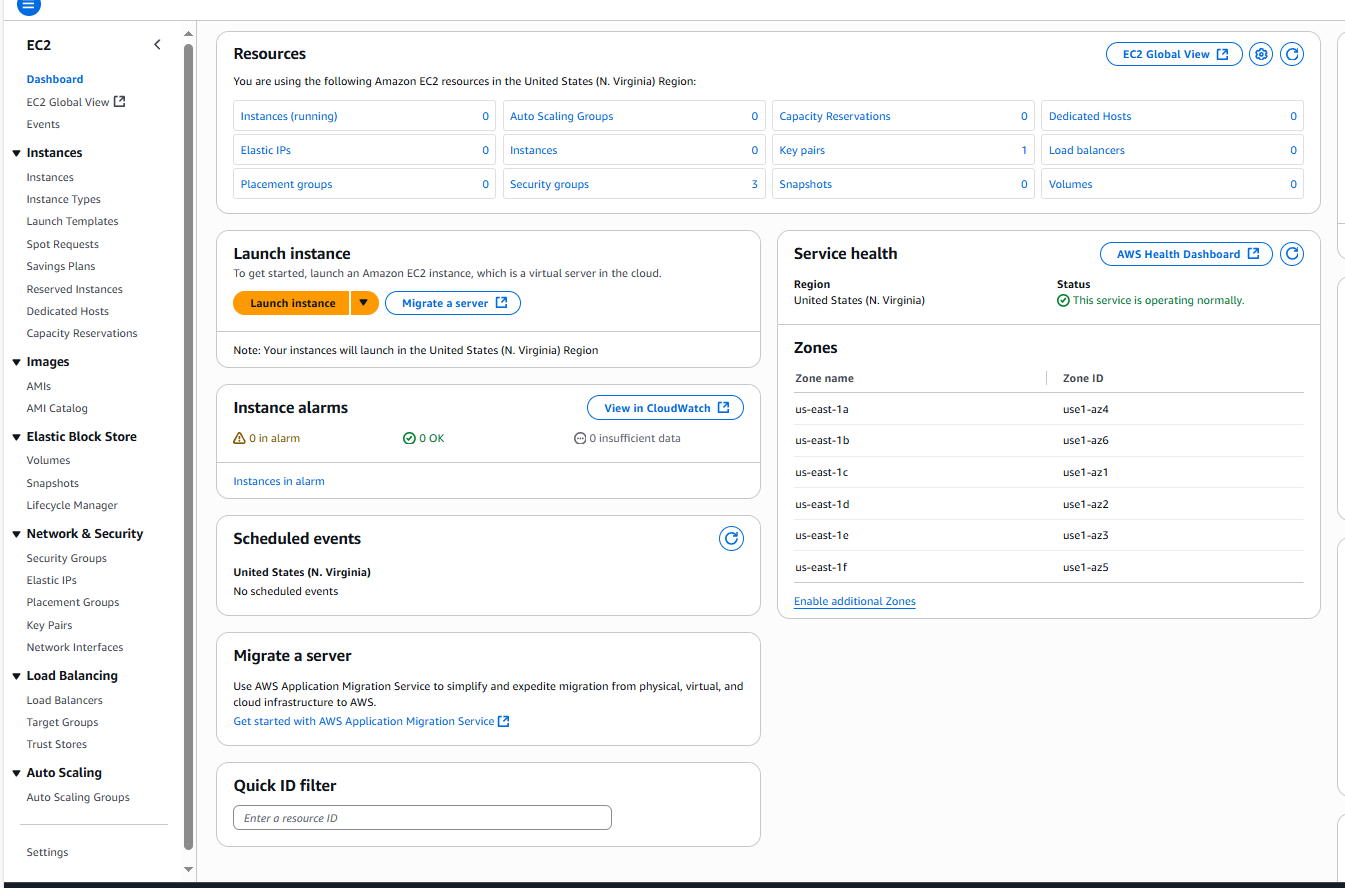
****

**RESULTS LINUX**

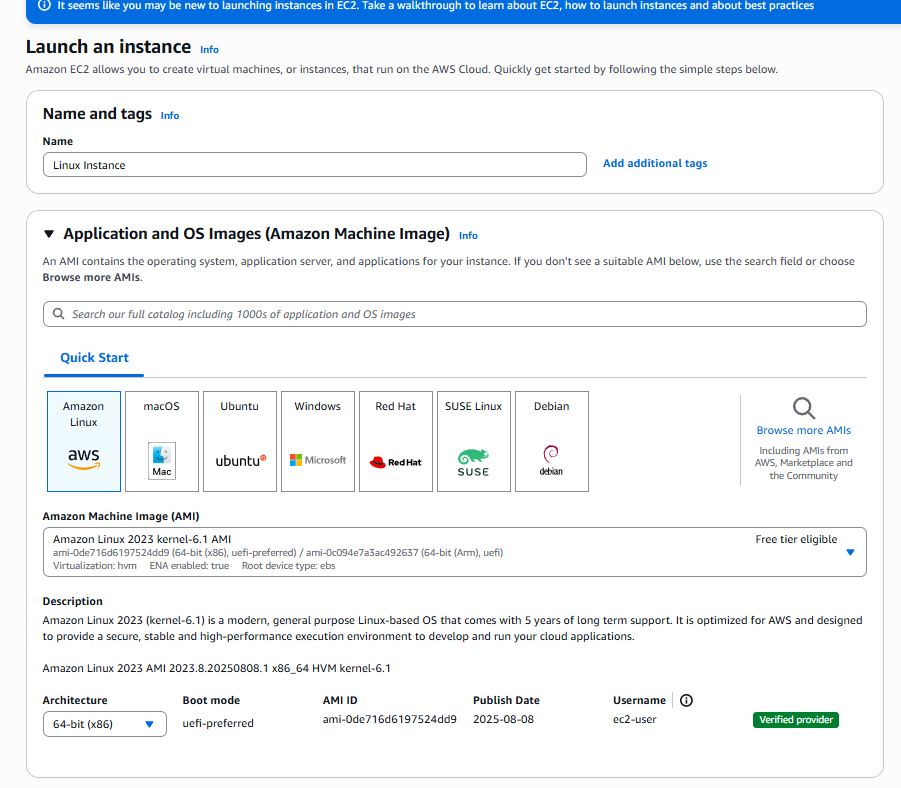
****

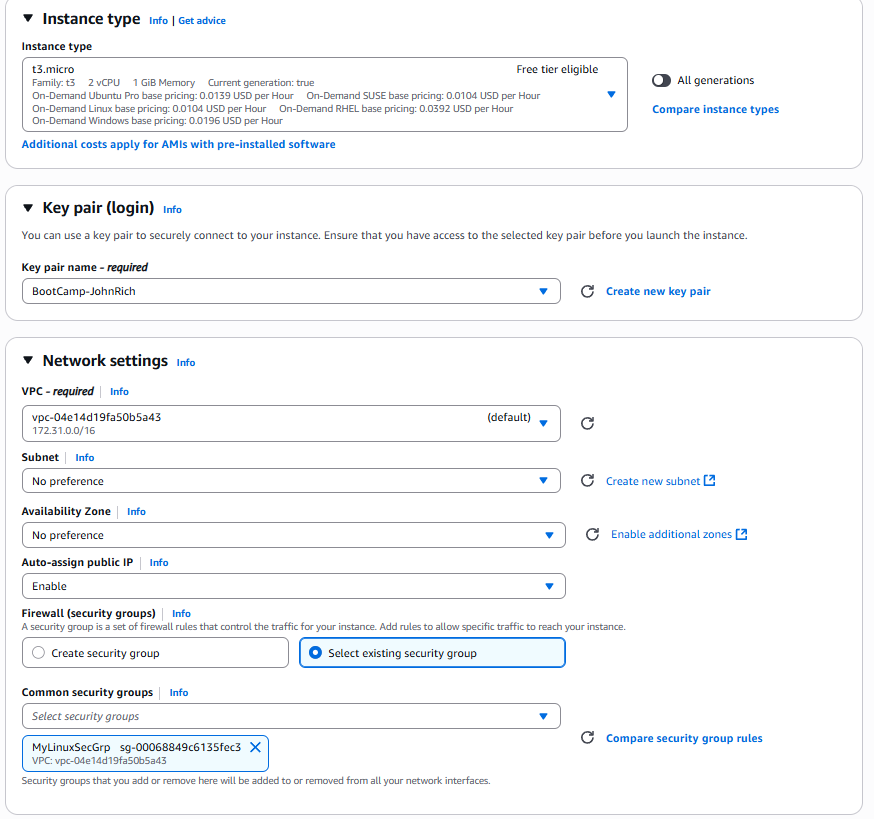
****

**LAUNCHING AN INSTANCE – LINUX**

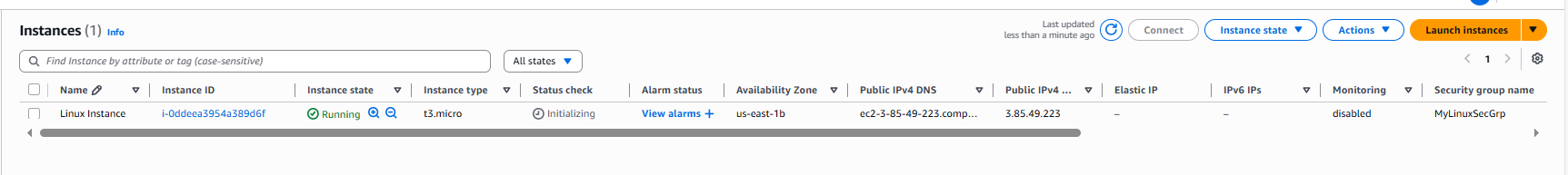
****

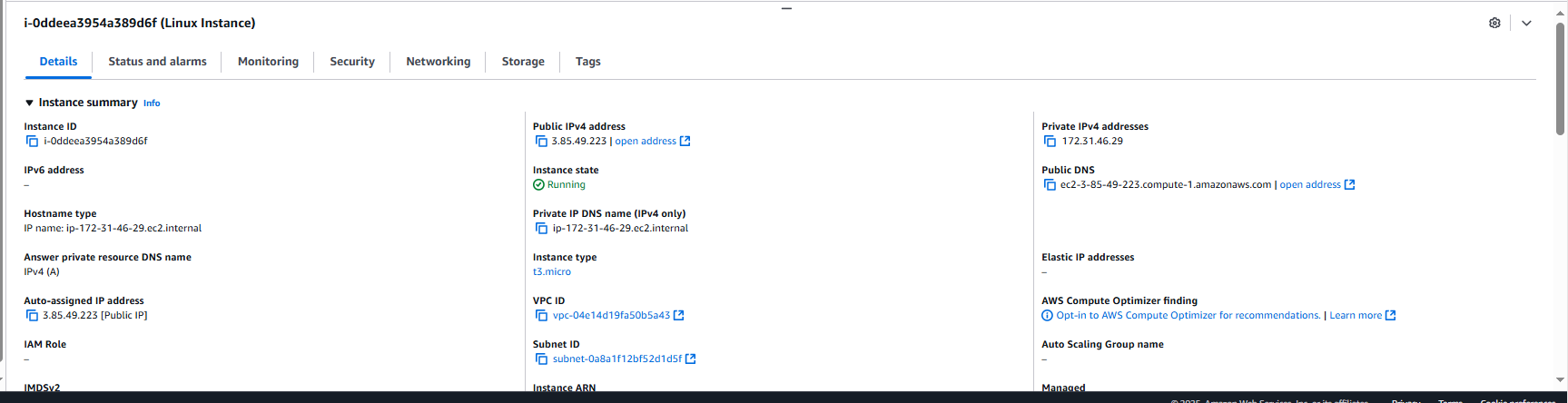
STEP 2





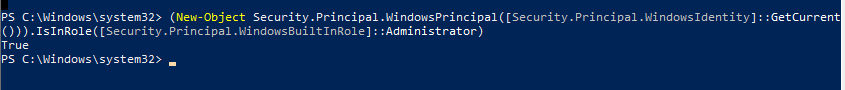
INSTANCE CONFIMATION



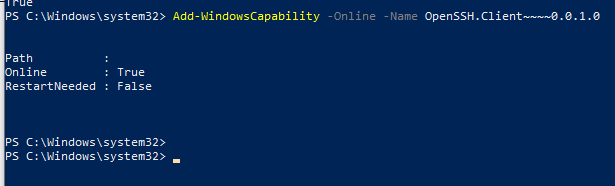


**Connect to your Linux instance from Windows using OpenSSH**

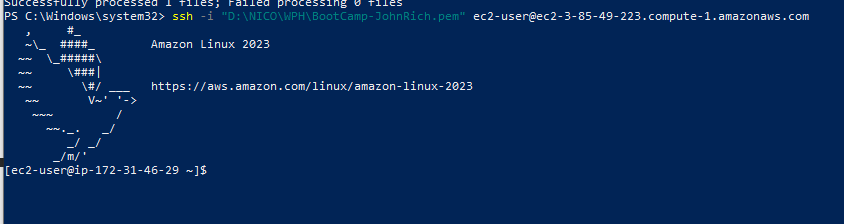
**STEP 1 – CONFIRMATION**

****

**INSTALLATION OF OPENSSH**

****

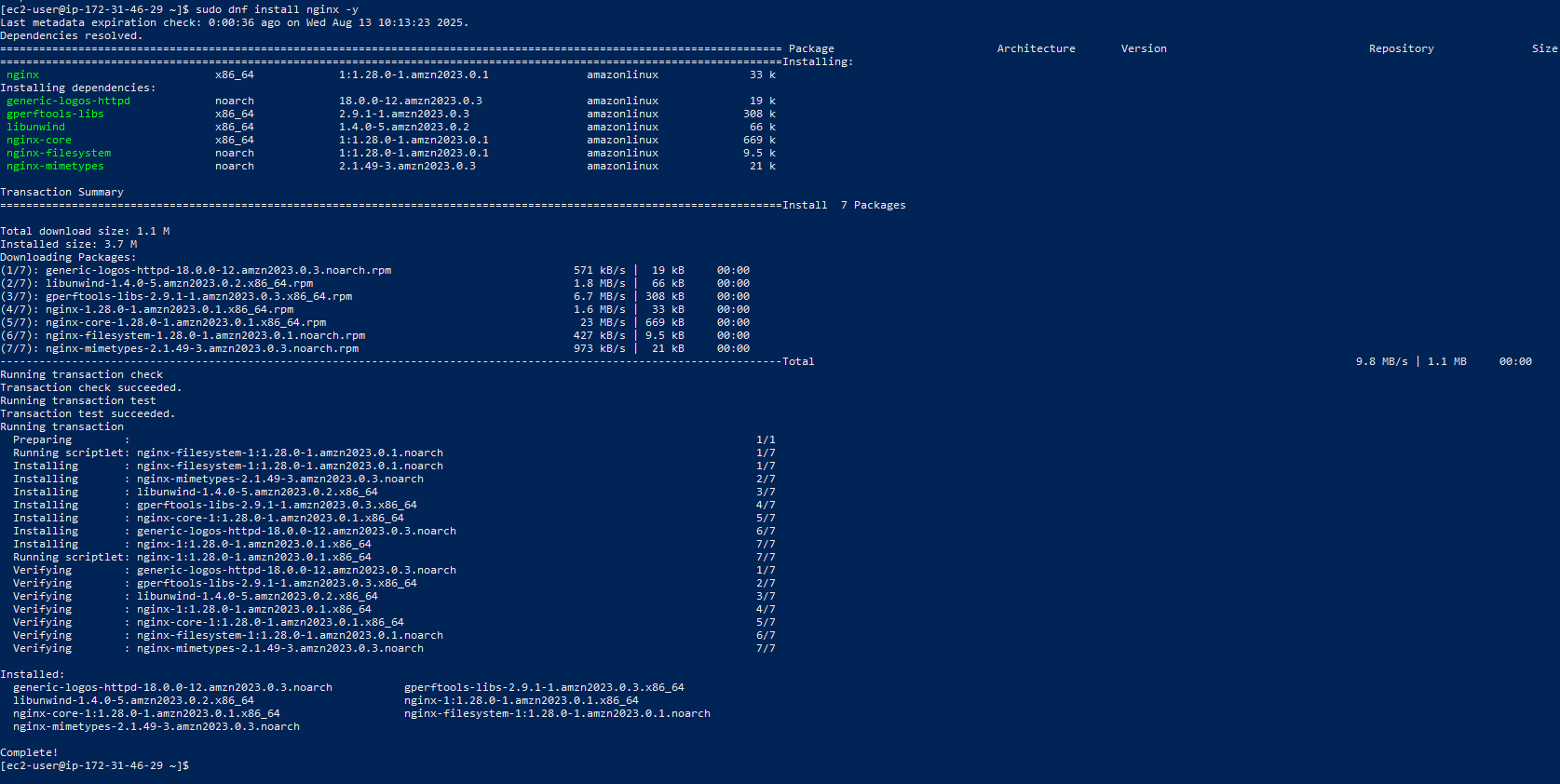
**SSH COMMAND**

****

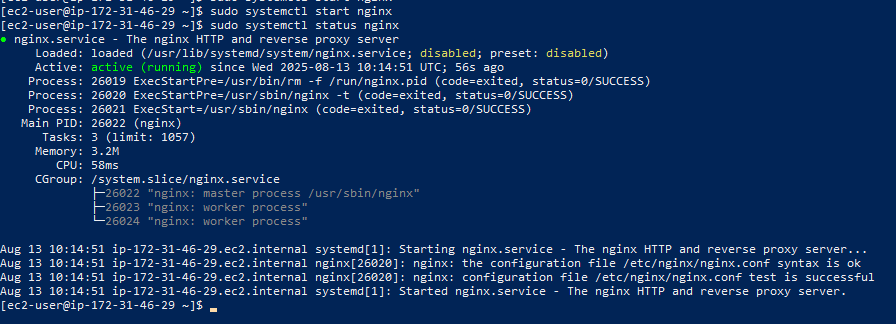
**SUDO DNF UPDATE -Y**

****

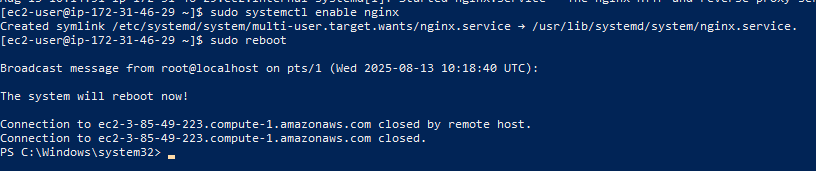
**sudo dnf install nginx -y**

****

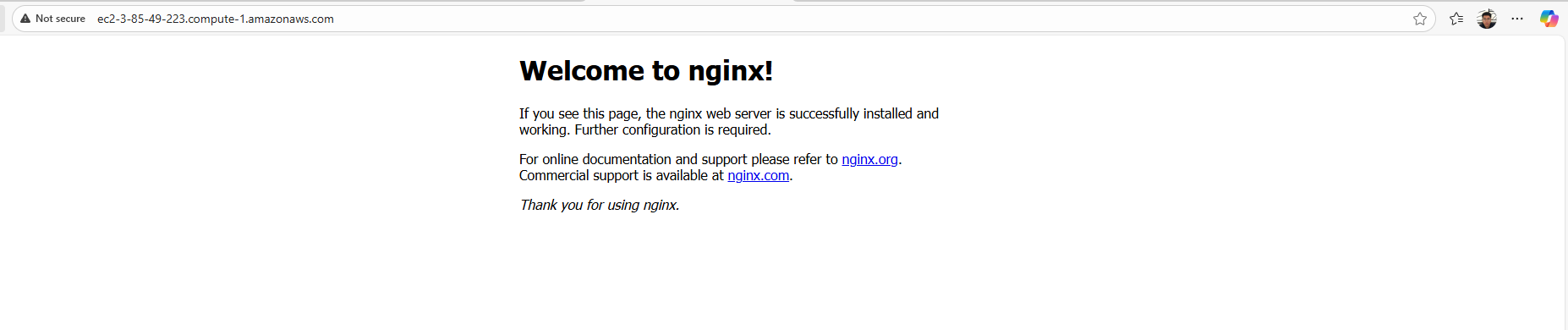
**sudo systemctl start nginx & sudo systemctl status nginx**



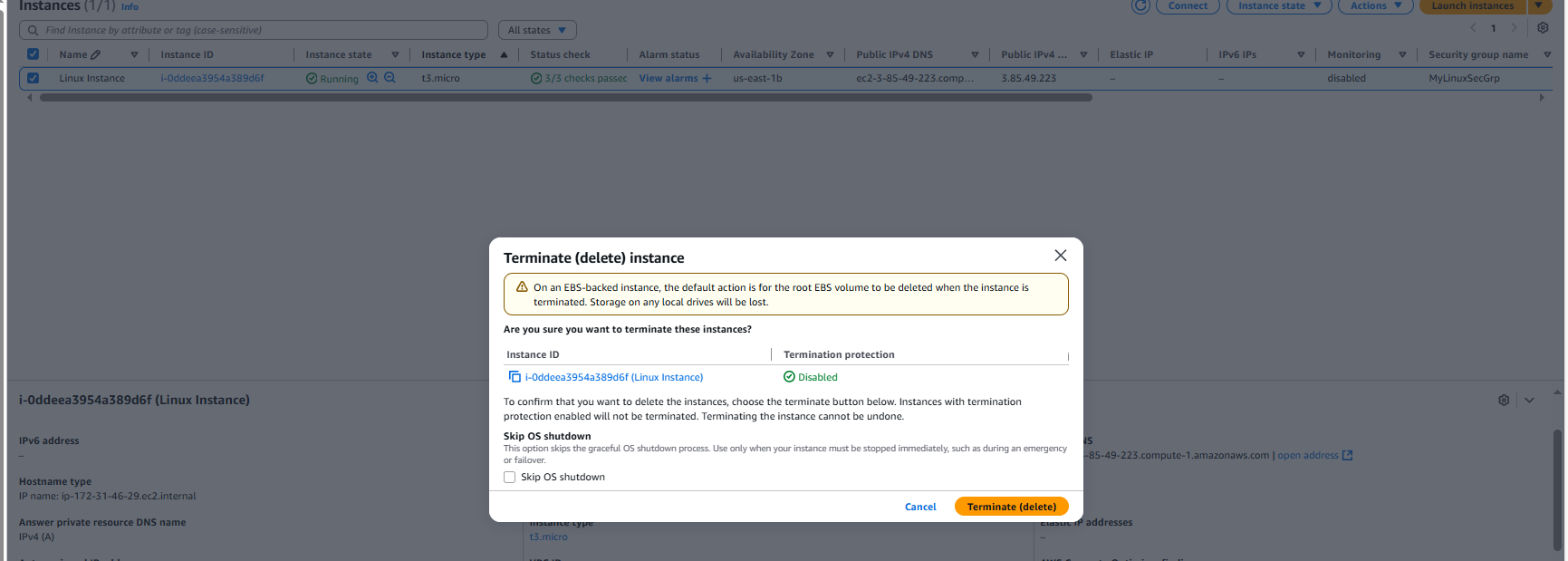
sudo systemctl enable nginx & sudo reboot

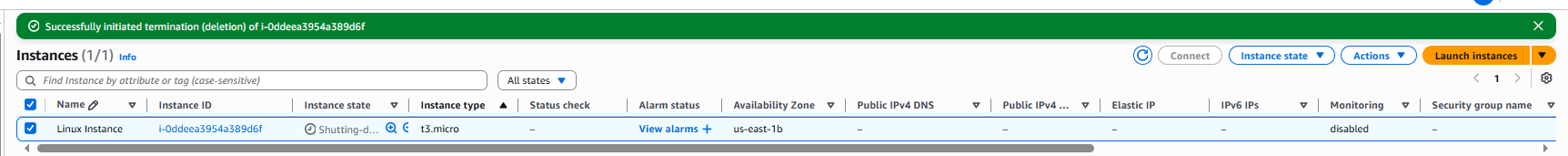


CHECK IF ITS RUNNING

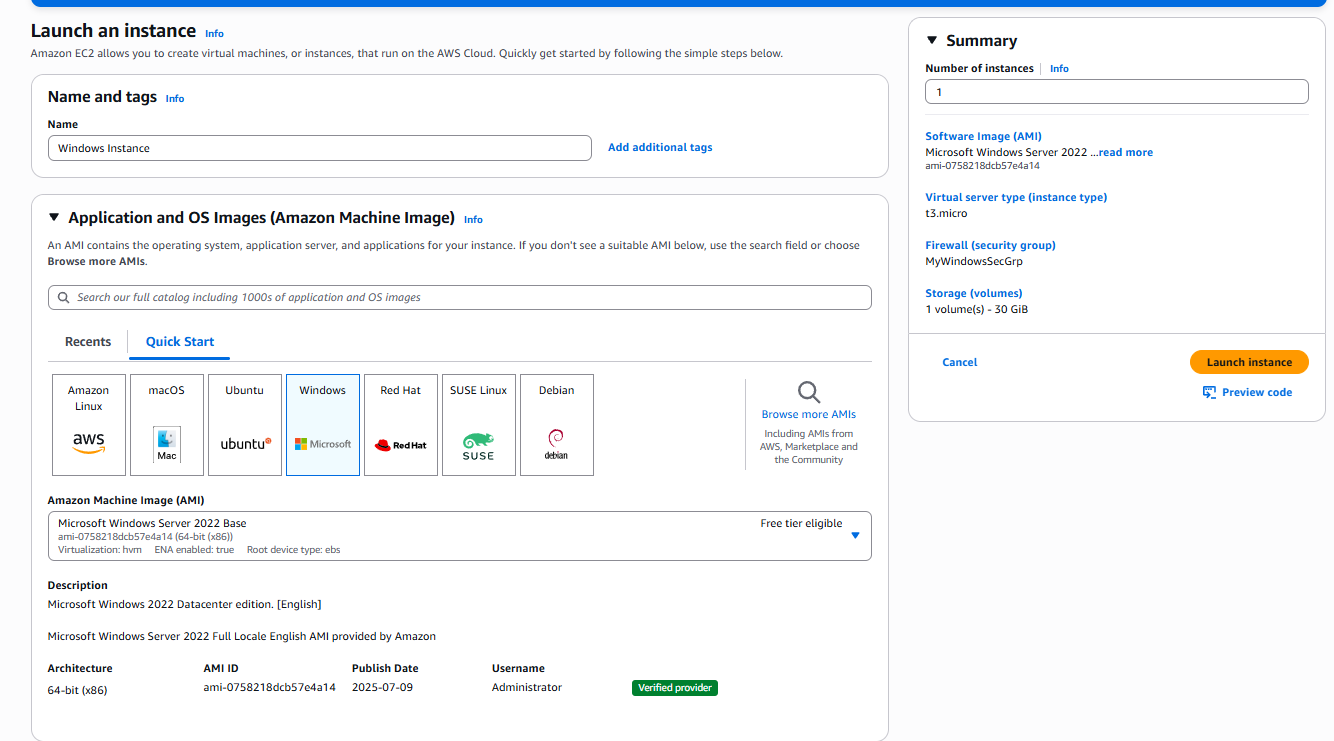


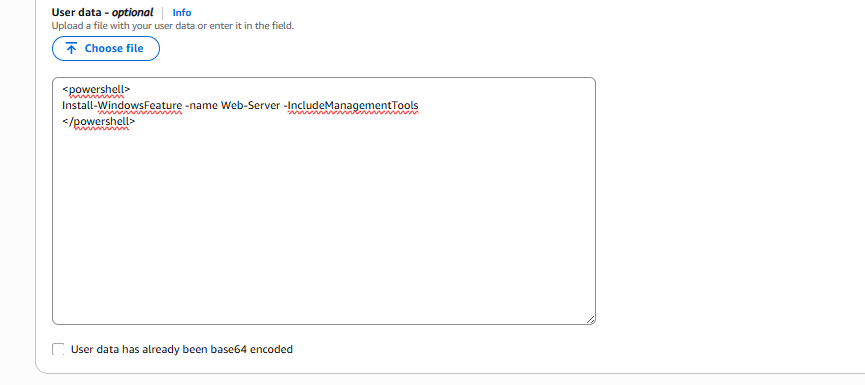
**CLEAN UP INSTANCE**

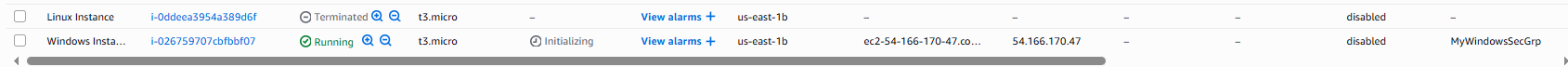
****

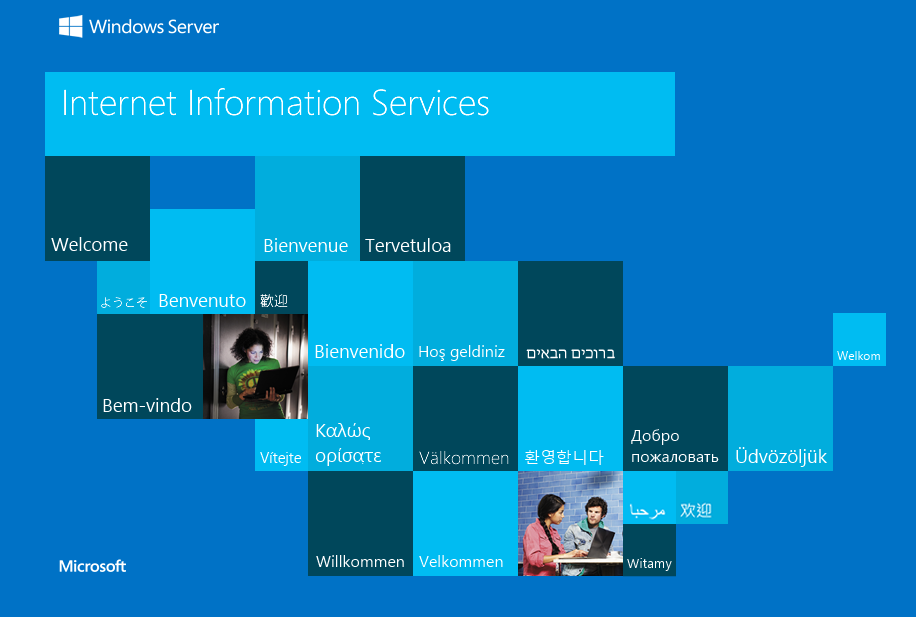
****

**LAUNCH AN INSTANCE – WINDOWS**

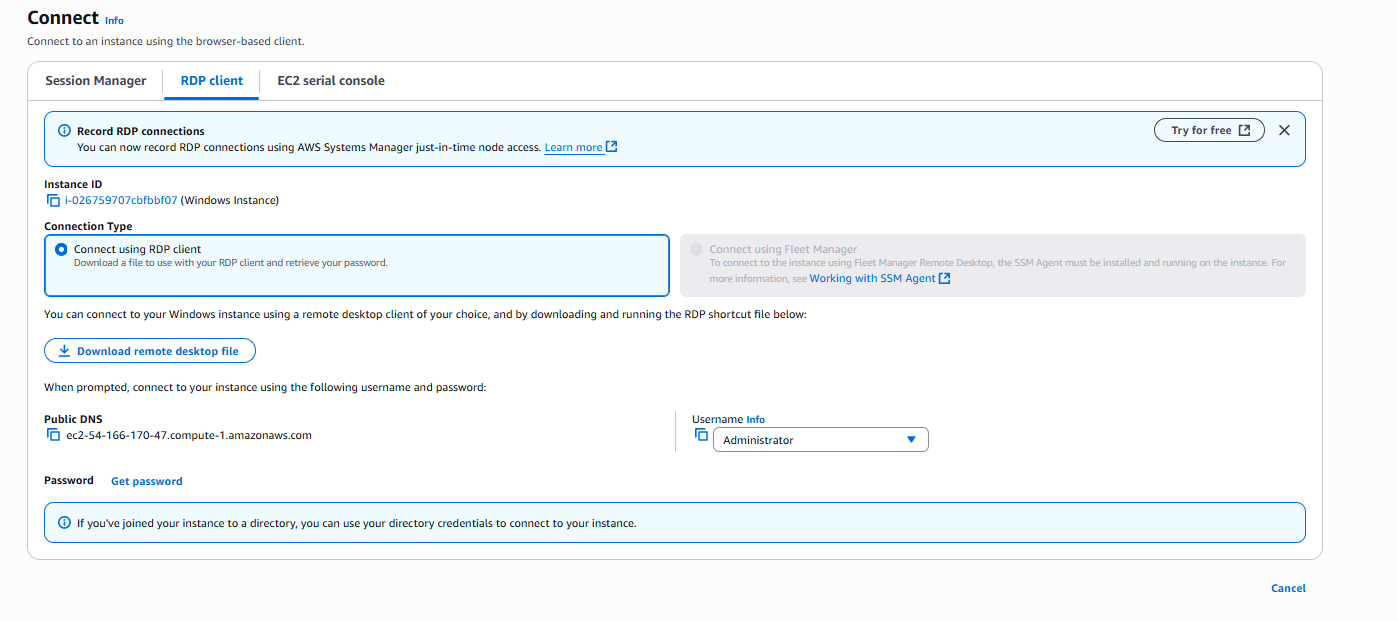
****

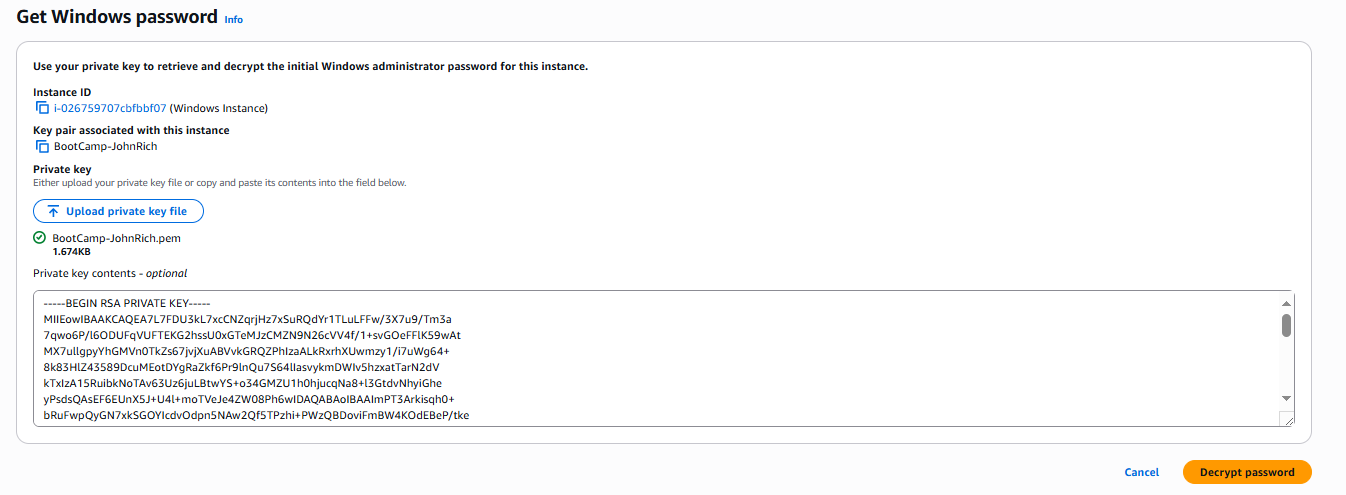
****

****

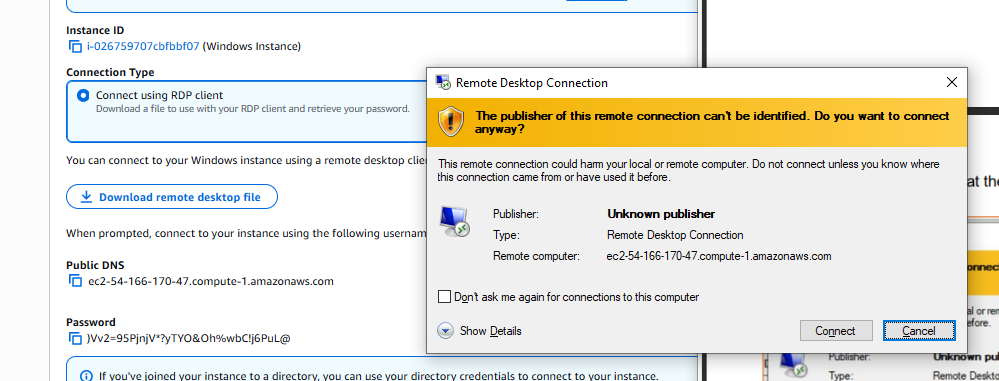


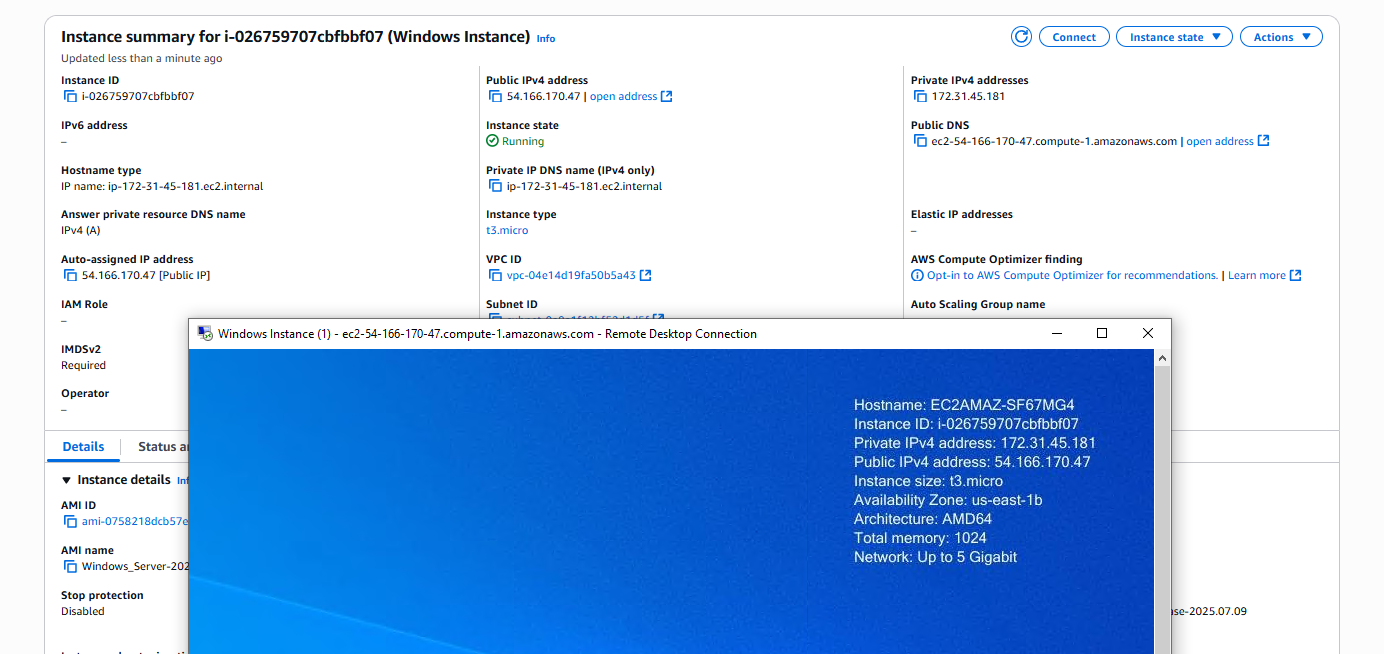
CONNECTING TO RDP



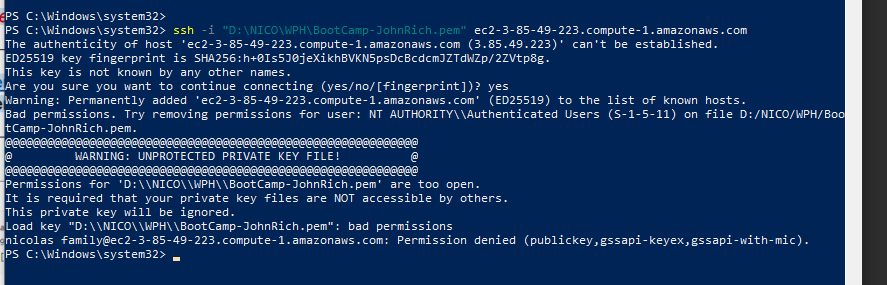


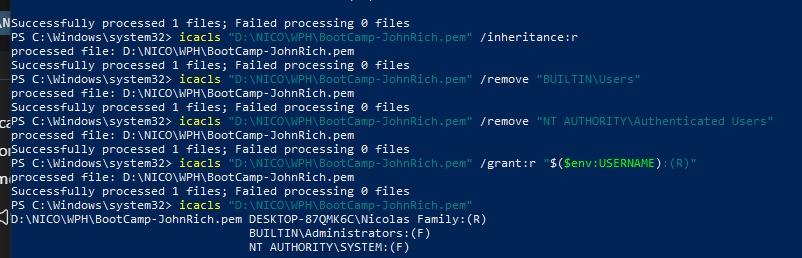
CONNECT





ERROR ENCOUNTERED

****

SOLUTION

REFLECTION

Amazon EC2 (Elastic Compute Cloud) is a service in AWS that lets you create and run virtual machines in the cloud. In this activity, the goal was to create and access both Linux and Windows instances so I could get hands-on experience with different operating systems in EC2. Before starting, I had to prepare some prerequisites like creating a key pair, setting up security groups for both Linux and Windows, installing an SSH client for Linux access, and making sure I had Remote Desktop Protocol (RDP) for Windows. I also had to make sure I was in the correct AWS region so everything I set up would be consistent.

Part of the process was generating a key pair, making sure I selected the right format (.pem) for Linux, downloading it, and saving it securely. For Linux, I learned that I needed to set the correct file permissions before connecting, otherwise it wouldn’t work. I also had to create separate security groups—one for Linux with an inbound rule for SSH (port 22) and one for Windows with an inbound rule for RDP (port 3389). Both security groups also had outbound rules that allowed all traffic so the instances could send and receive data.

When launching the Windows Server 2022 instance, I attached it to the Windows security group and added a user data script in the advanced settings to install IIS Web Server automatically. After it launched, I was able to connect to it through RDP and see that IIS was installed. Finally, to avoid unnecessary costs, I made sure to properly terminate both my Linux and Windows instances after testing.

One of the challenges I faced was when I tried connecting to my Linux instance but got an error saying my private key file had “bad permissions.” The error also mentioned removing permissions for NT AUTHORITY\Authenticated Users on my .pem file. I solved this by adjusting the file permissions so it wasn’t accessible to other users, and after that, I was able to connect successfully.

Overall, the whole process was time-consuming and honestly overwhelming because there was just too much information to keep track of. On top of that, I was nervous to continue at some points because I borrowed my sister’s credit card for my AWS account and I was scared that if I made a mistake, I might accidentally get charged. This activity taught me not only how to launch and connect to instances, but also how to troubleshoot common connection problems—while being careful with every step so I wouldn’t get billed unexpectedly.