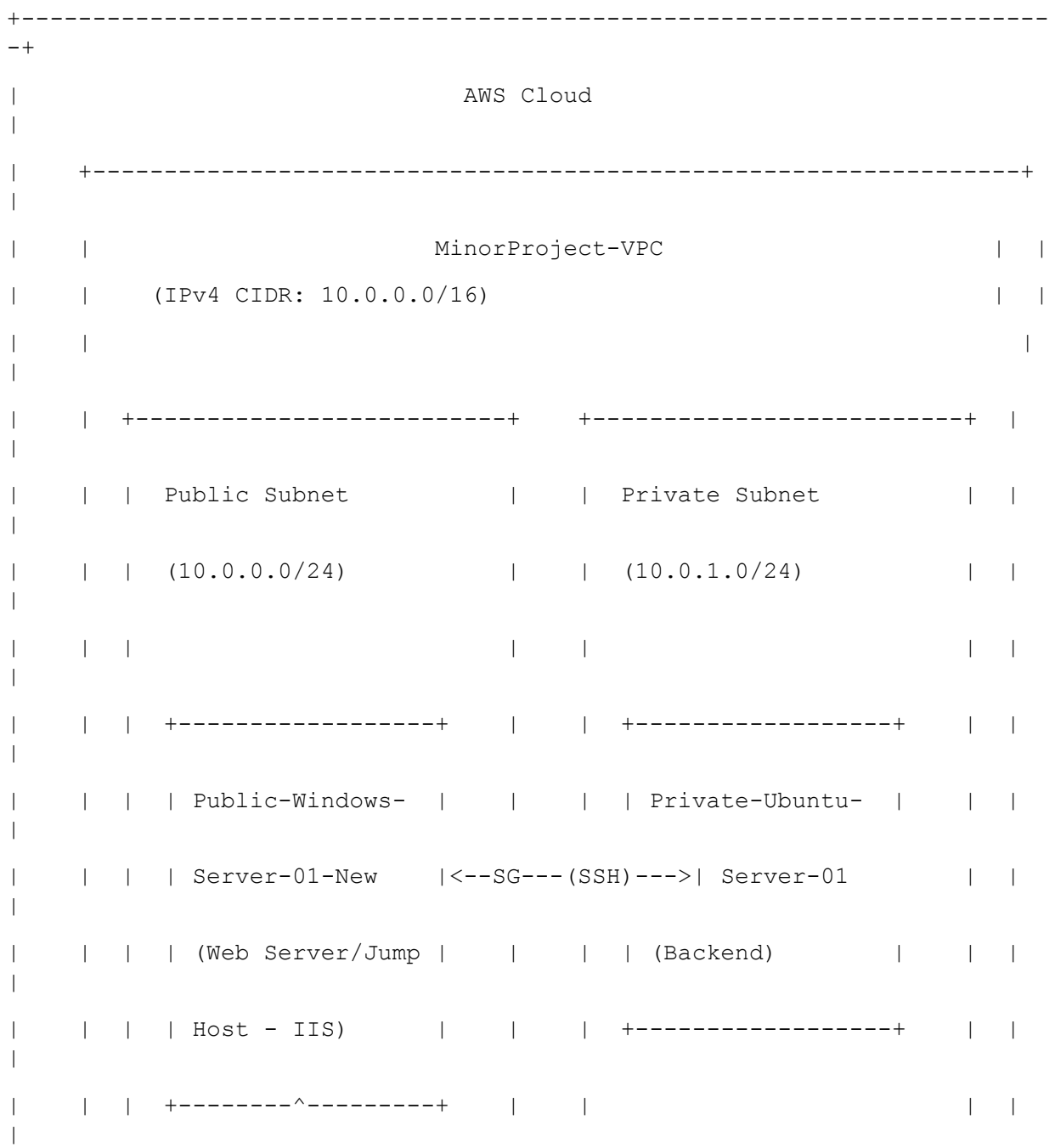
AWS Cloud Infrastructure Deployment Project

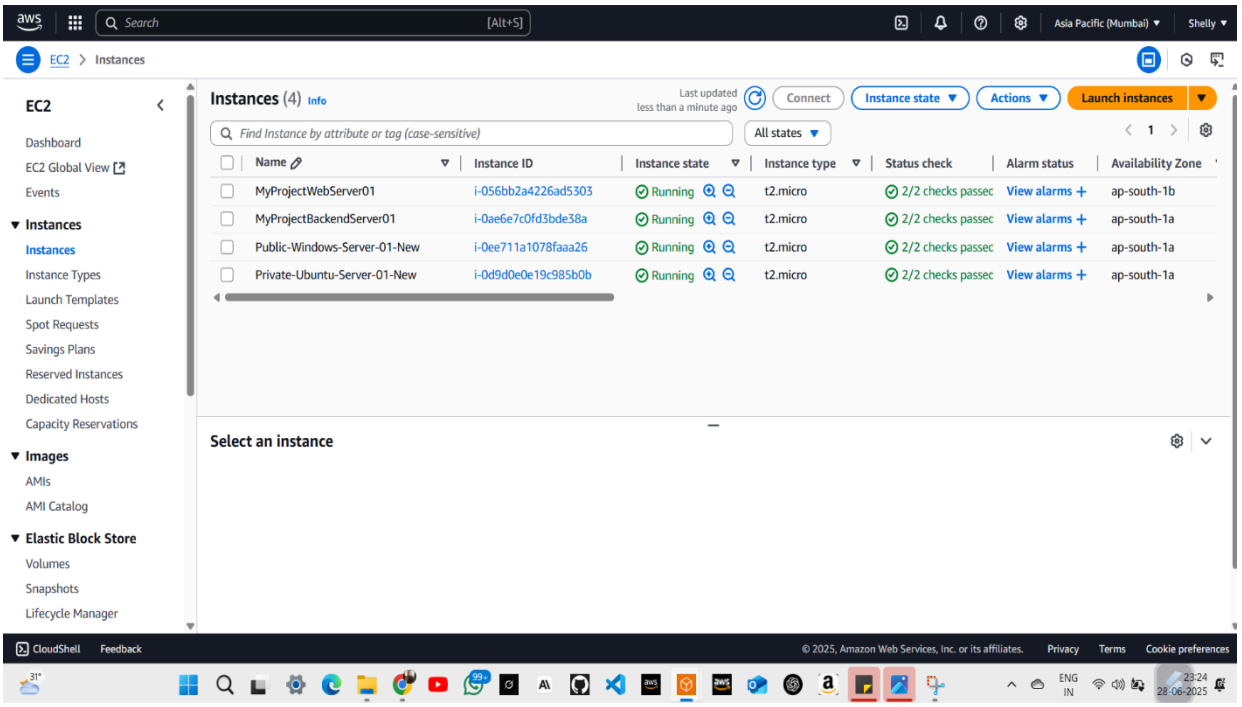
1. Project Overview and Architecture

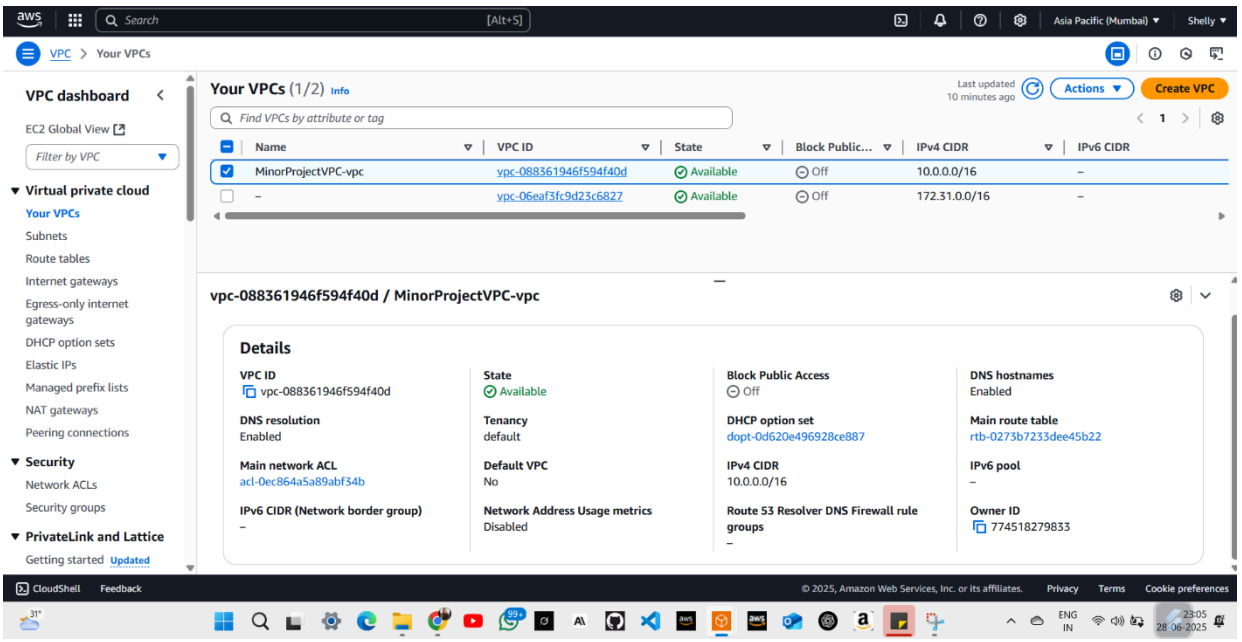
This project involved designing and implementing a two-tier application architecture within a Virtual Private Cloud (VPC) on Amazon Web Services (AWS). The primary objective was to demonstrate secure network segmentation, deploying a publicly accessible web server (Windows) in a public subnet and a privately accessible backend server (Ubuntu) in a private subnet. The private server was configured to be accessible only via the public web server acting as a secure "jump host."

Architecture Diagram (Conceptual)



```
|     |   |              | (HTTP)          |     |                                |   |
|
|     |   +------------|--------------+     +--------------------------+   |
|
|     |                (IGW)                                                |
|
|     +--------------------^-------------------------------------------+
|
|                          |
|
|         Internet         |<---(RDP/HTTP)-----------------------------|
+--------------------------+
```



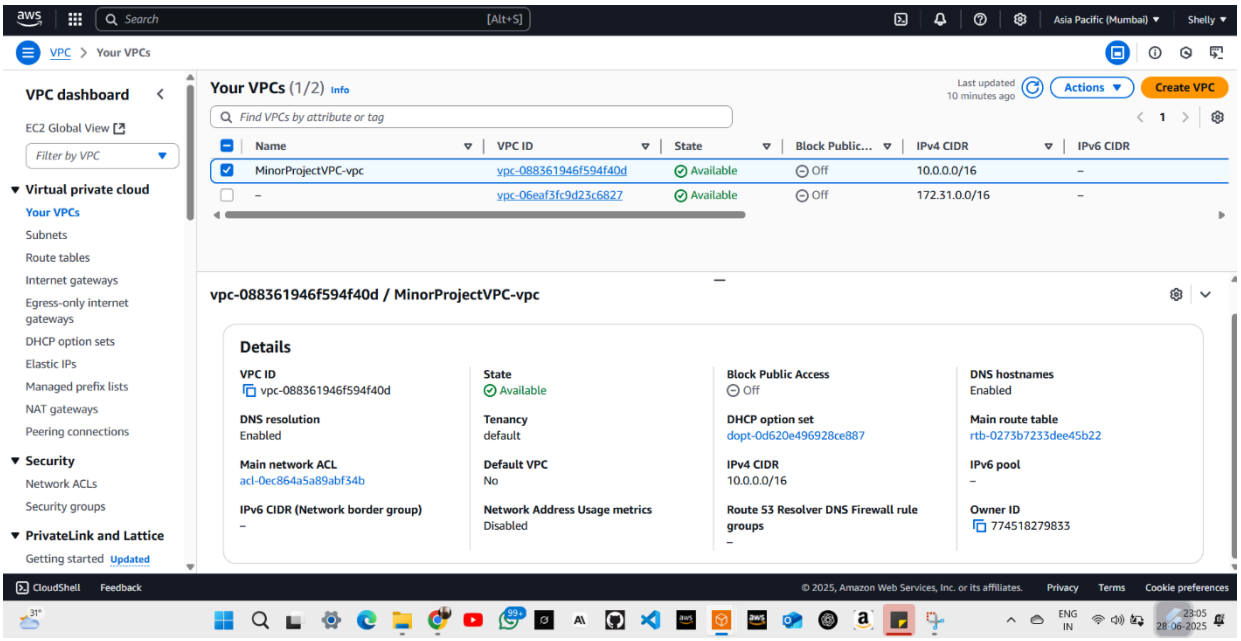


2. Detailed Deployment Steps

2.1 Virtual Private Cloud (VPC) and Subnets Creation

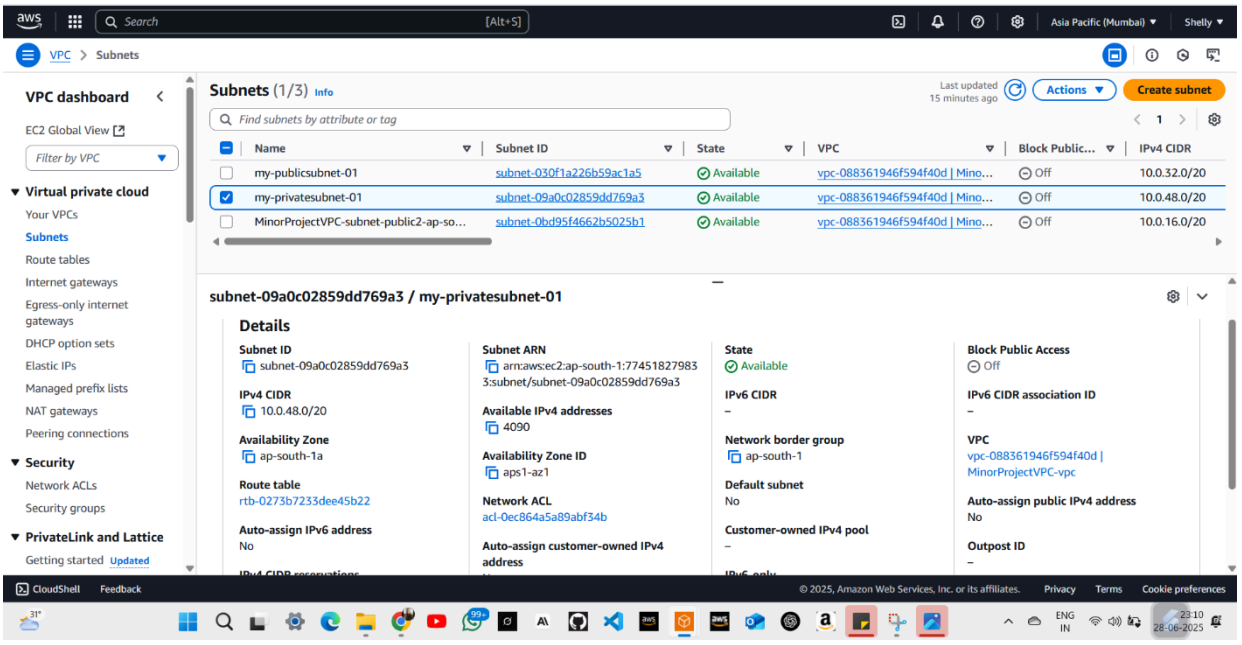
The foundation of our isolated network environment was the custom VPC.

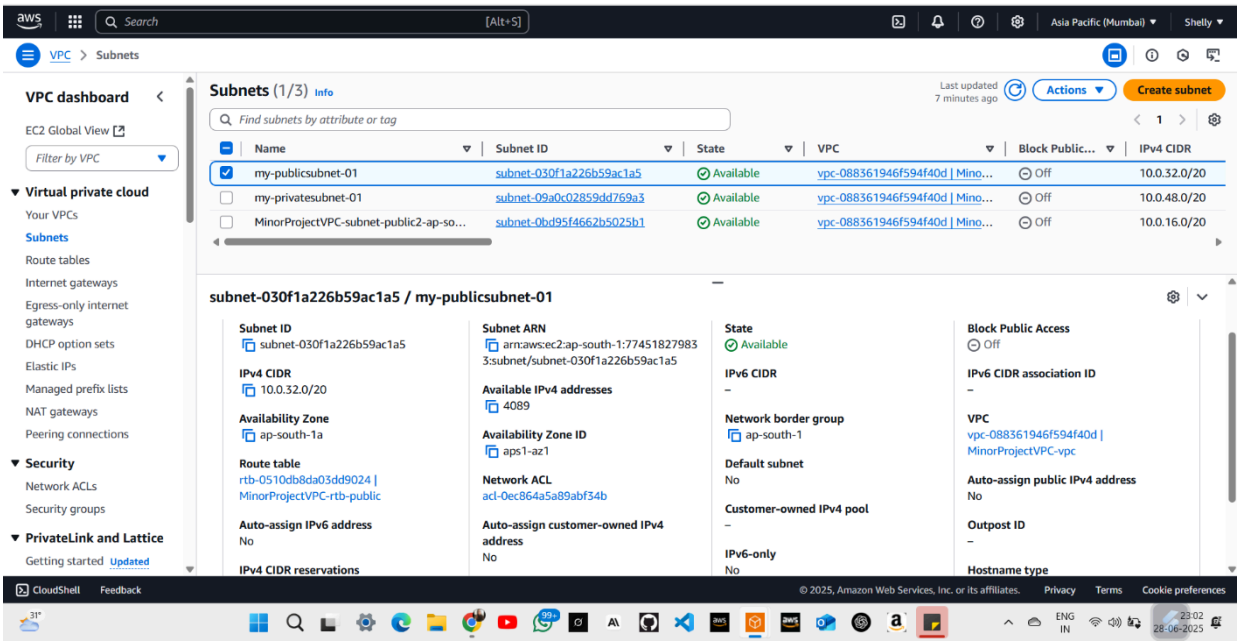
- **VPC Creation:** We created a new VPC named MinorProject-vpc with an IPv4 CIDR block of 10.0.0.0/16. This large CIDR provided ample IP addresses for future expansion.



- **Subnet Creation:** Within this VPC, we created two distinct subnets:

- **Public Subnet (my-publicsubnet-01):** 10.0.0.0/24 (a smaller block within the VPC CIDR). This subnet was designated for publicly accessible resources.
- **Private Subnet (my-privatesubnet-01):** 10.0.1.0/24 (another smaller block). This subnet was for resources that should not be directly accessible from the internet.



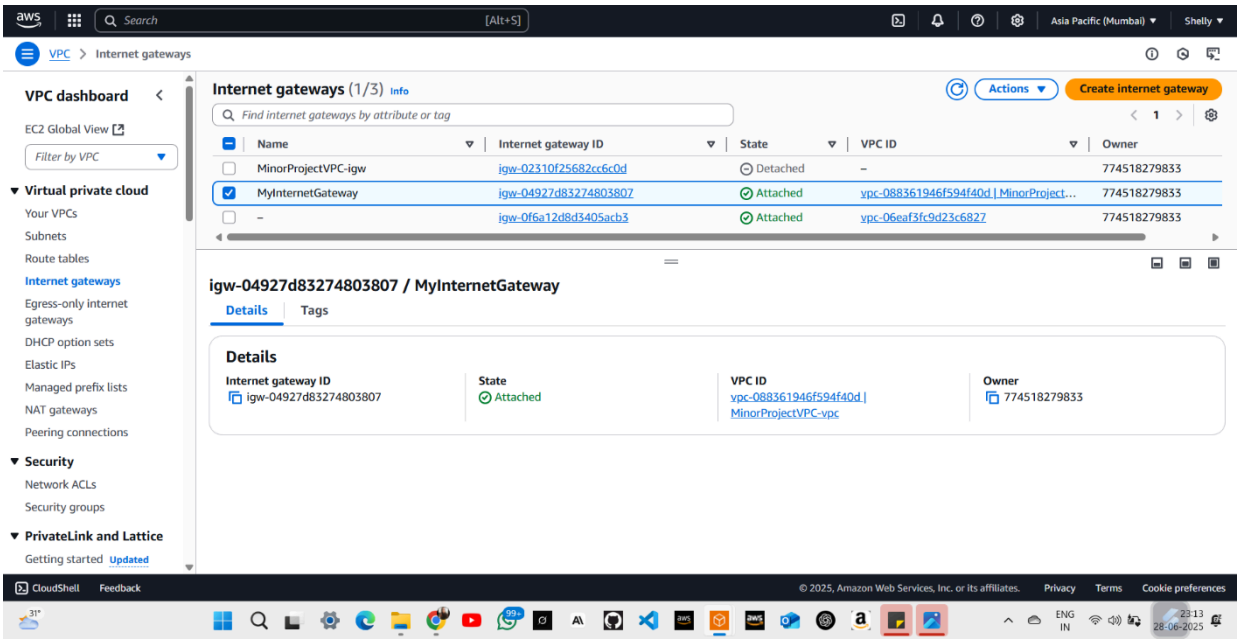


2.2 Internet Gateway (IGW) and Route Tables Configuration

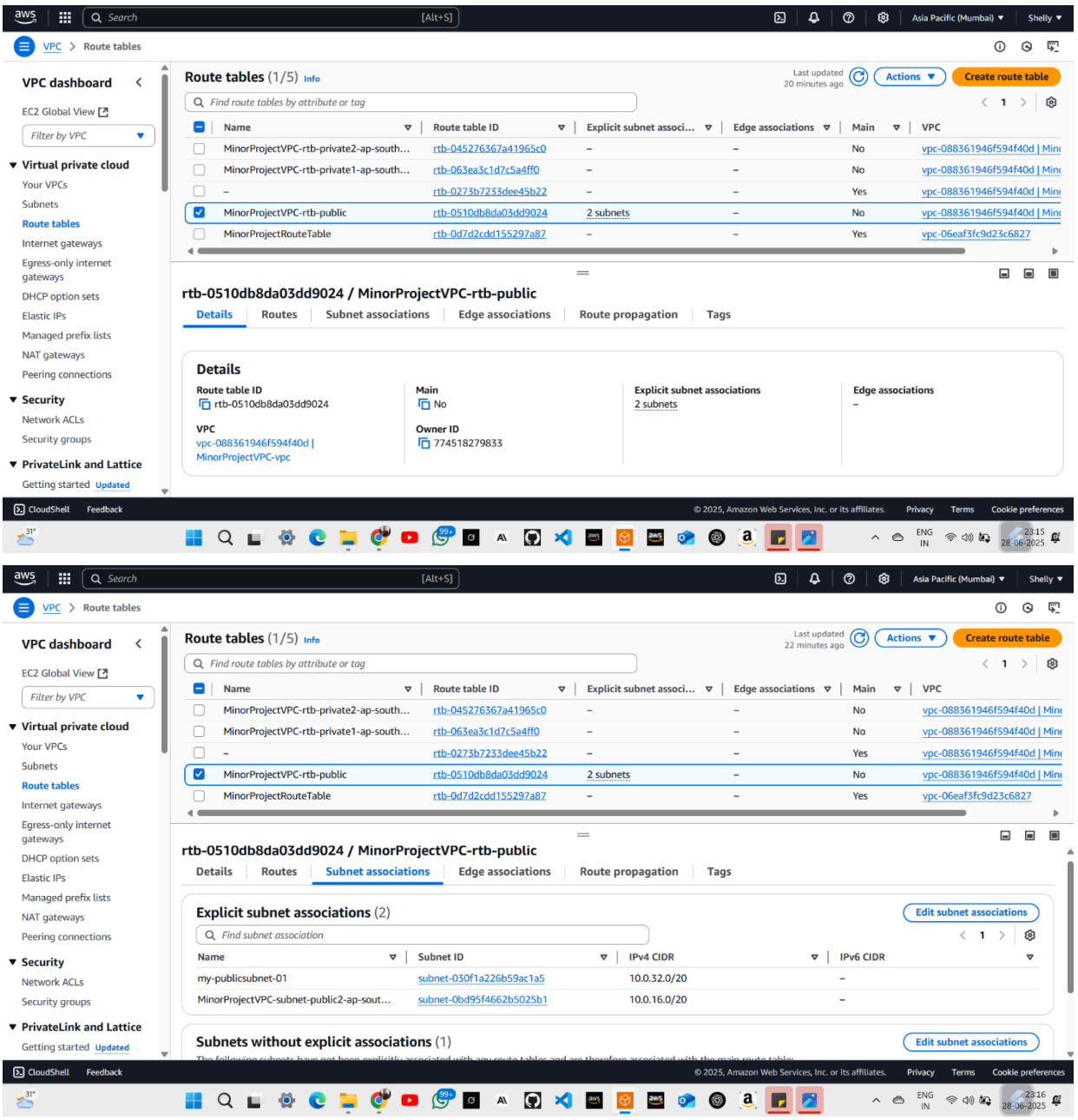
To enable internet connectivity for our public resources, an Internet Gateway and a custom route table were configured.

- **Internet Gateway Creation and Attachment:** An Internet Gateway named MinorProjectVPC-igw was created and then attached to our

MinorProject-vpc. This acts as a logical connection between the VPC and the internet.



- **Custom Route Table Configuration:**
  - A custom route table named MinorProjectRouteTable was created within MinorProject-vpc.
  - A default route (0.0.0.0/0) was added to this route table, pointing all internet-bound traffic to the MinorProjectVPC-igw.
  - The my-publicsubnet-01 was explicitly associated with MinorProjectRouteTable, making it a public subnet. The private subnet remained associated with the VPC's main route table (or was left unassociated to prevent direct internet access).

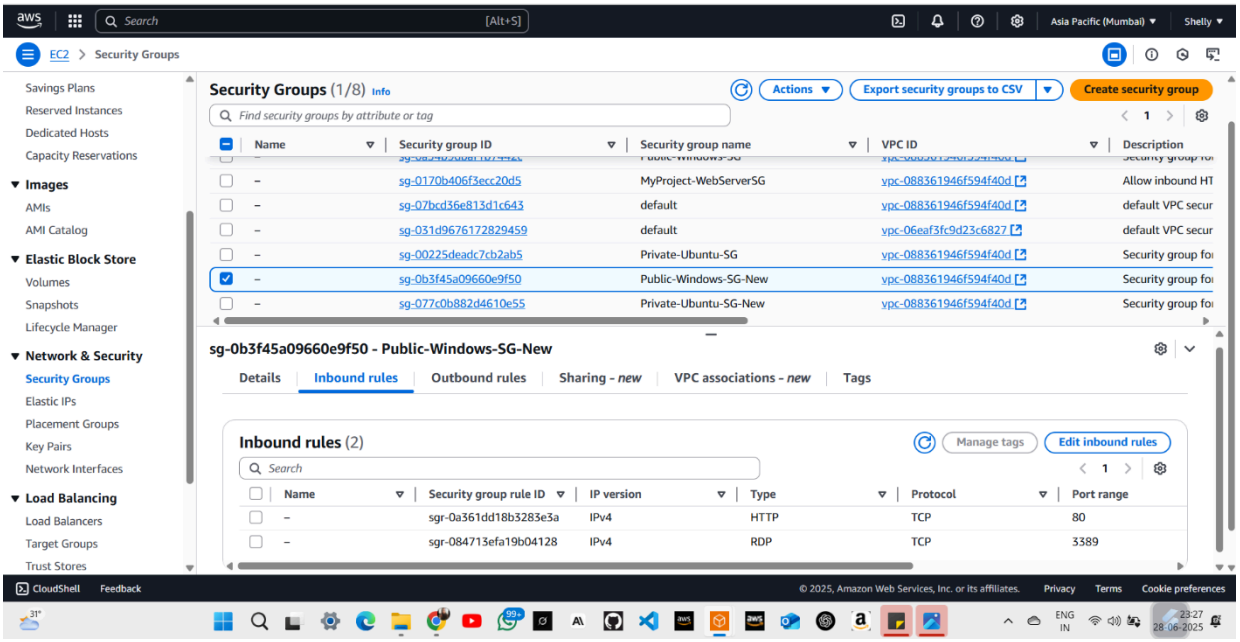


2.3 Security Group Configuration

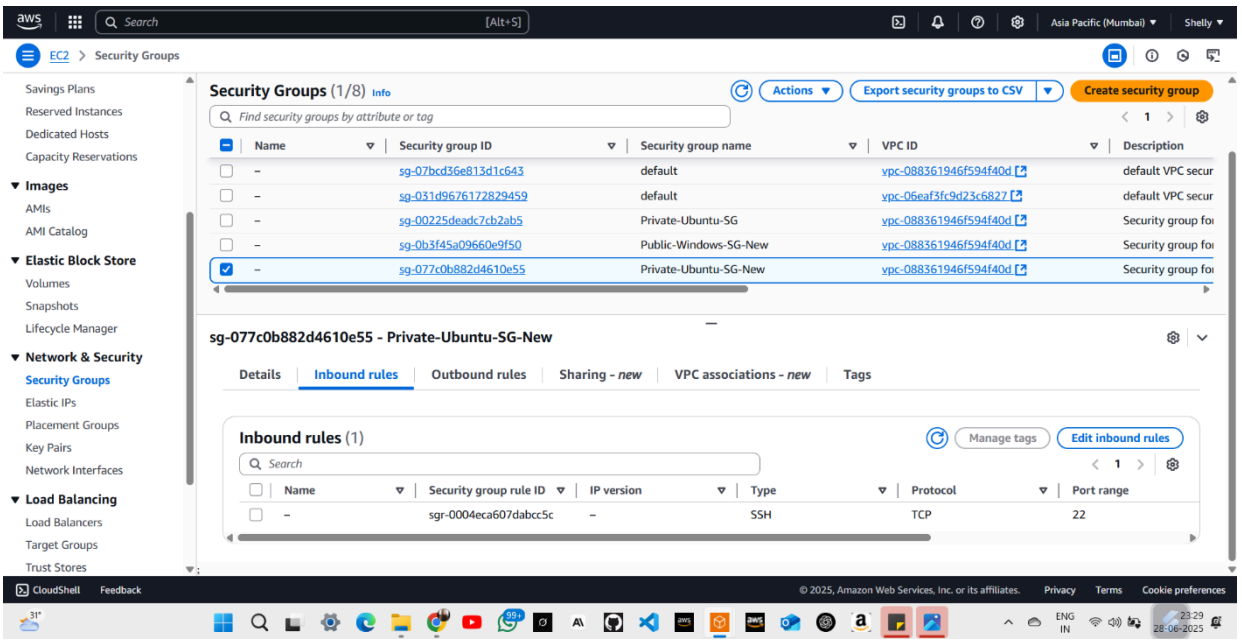
Security groups acted as virtual firewalls to control inbound and outbound traffic for our instances.

- **Public-Windows-SG-New (for Public Windows Server):**
    - **Inbound Rules:**
        - **RDP (Port 3389):** Allowed from our specific public IP address (or 0.0.0.0/0 for initial testing) to enable remote desktop access for management.

- **HTTP (Port 80):** Allowed from 0.0.0.0/0 (anywhere) to enable public access to the web server.



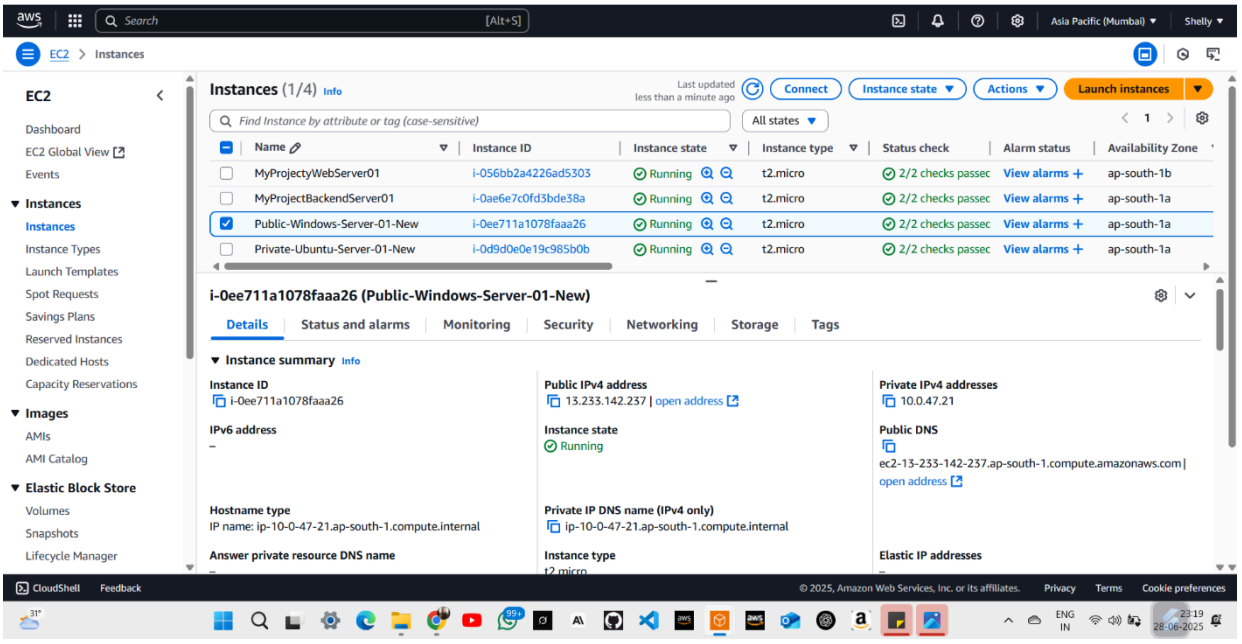
- **Private-Ubuntu-SG-New (for Private Ubuntu Server):**
  - **Inbound Rules:**
    - **SSH (Port 22): Crucially, this was allowed only from the Security Group ID of Public-Windows-SG-New.** This enforces that the private server can only be managed (via SSH) from the public Windows jump host, ensuring network isolation and enhanced security.



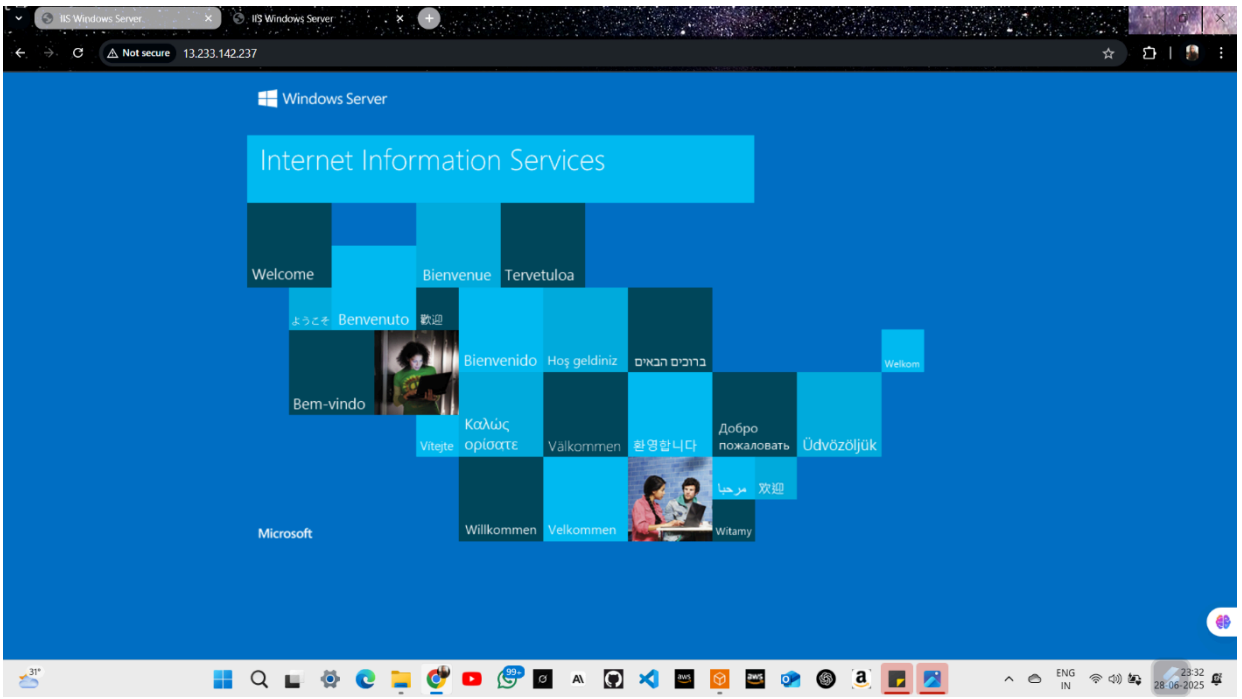
2.4 EC2 Instance Deployment - Public Windows Server

The public-facing web server was deployed to host web content and serve as a jump host.

- **Launch Instance:** An EC2 instance named Public-Windows-Server-01-New (type t2.micro or t3.micro) was launched using a Windows Server AMI.
- **Network Configuration:** It was placed in my-publicsubnet-01 and configured to auto-assign a Public IPv4 address. The Public-Windows-SG-New was attached to it.
- **Key Pair:** A new key pair, MyProjectNewKey, was created and used for secure initial access.
- **RDP Connection:** We successfully connected to this instance via Remote Desktop Protocol (RDP) from our local machine using its Public IPv4 address and the decrypted administrator password.



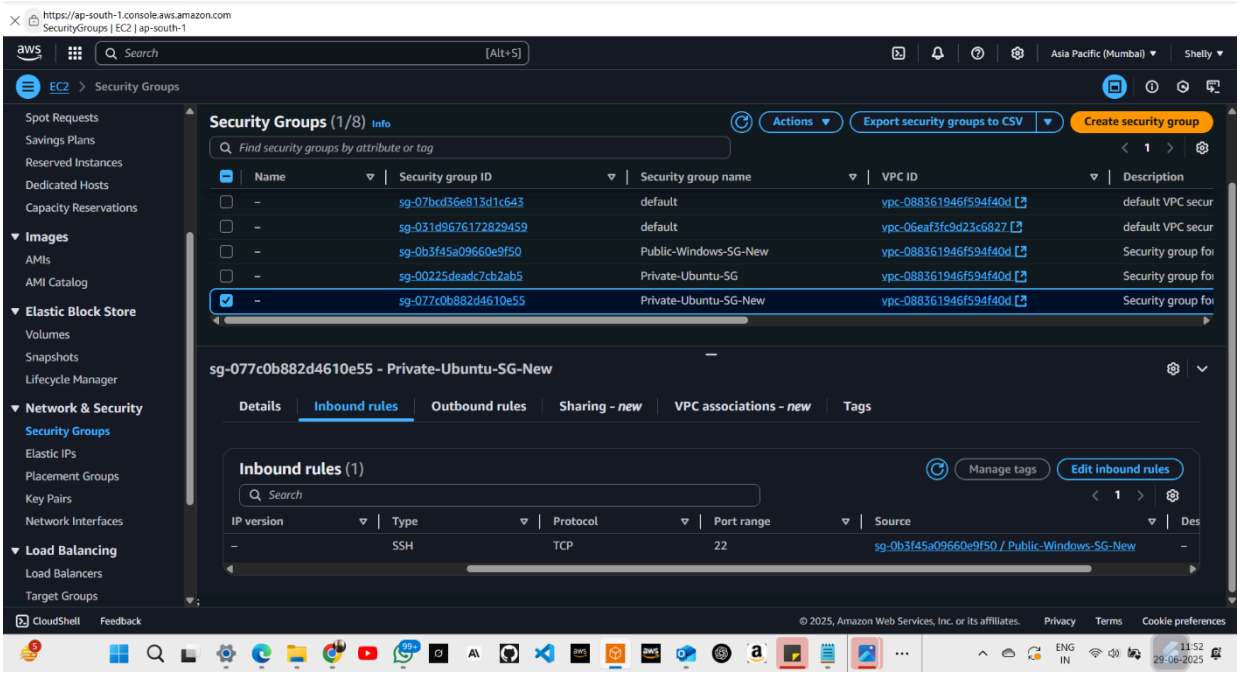
- **IIS Installation:** Internet Information Services (IIS) was installed on the Windows Server, turning it into a functional web server.
- **Public Accessibility Verification:** We accessed the instance's Public IPv4 address from a local web browser, confirming that the default IIS welcome page was displayed, proving its public accessibility and web server functionality.



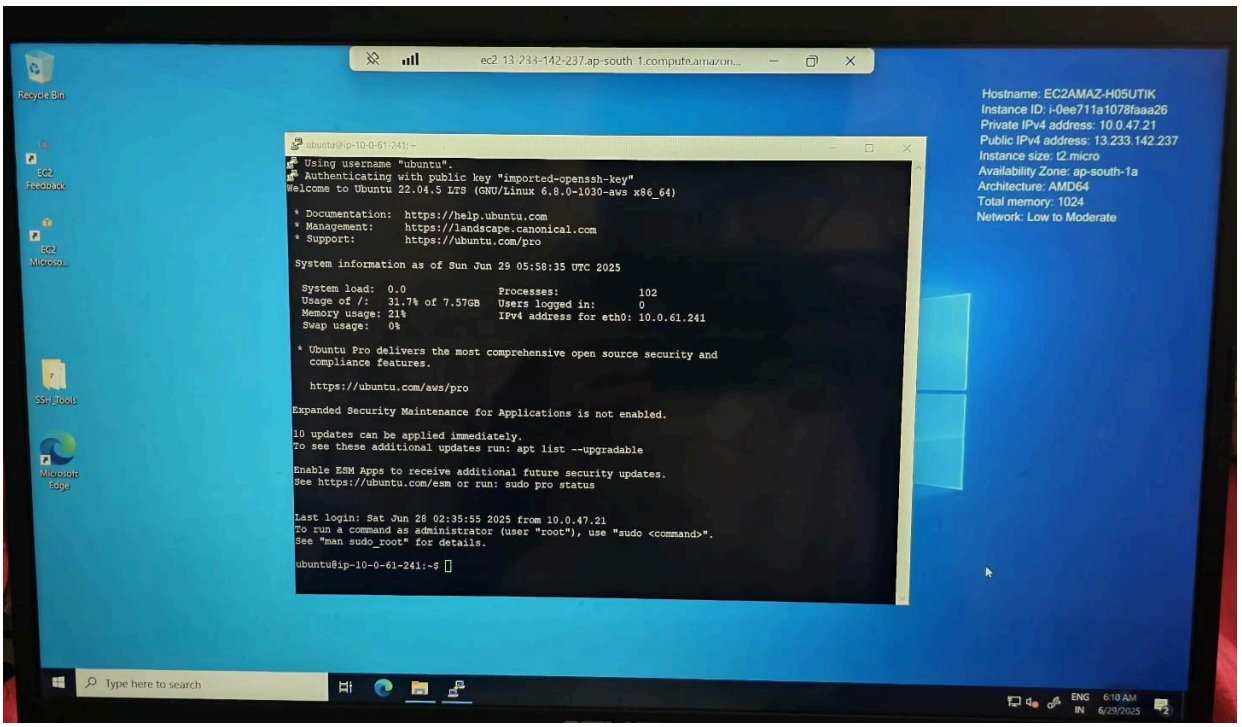
2.5 EC2 Instance Deployment - Private Ubuntu Server

The backend server was deployed in the private subnet, accessible only from the jump host.

- **Launch Instance:** An EC2 instance named Private-Ubuntu-Server-01 (type t2.micro or t3.micro) was launched using an Ubuntu Server AMI.
- **Network Configuration:** It was placed in my-privatesubnet-01 and **configured NOT to auto-assign a Public IPv4 address,** ensuring its isolation from the internet. The Private-Ubuntu-SG-New was attached.
- **Key Pair:** The same MyProjectNewKey was used for this instance.



- **SSH Connection via Jump Host (PuTTY):**
    - From within the RDP session on Public-Windows-Server-01-New, PuTTY and PuTTYgen were used.
    - The MyProjectNewKey.pem file was transferred to the Windows server and converted to MyProjectNewKey.ppk using PuTTYgen.
    - A PuTTY SSH connection was established from the Public-Windows-Server-01-New to the Private-Ubuntu-Server-01 using its Private IPv4 address and the .ppk key. This successfully demonstrated secure, indirect access to the private resource.



3. Proof of Concept and Verification

The successful deployment was verified by:

- Accessing the IIS welcome page on the Public-Windows-Server-01-New from a local web browser, demonstrating public web server functionality.
- Successfully establishing an SSH connection from the Public-Windows-Server-01-New (acting as a jump host) to the Private-Ubuntu-Server-01, proving the secure private network access. The inability to directly access the Private-Ubuntu-Server-01 from outside the VPC further confirmed its isolation.

4. Resource Cleanup

Upon project completion and documentation, all created AWS resources were systematically terminated and deleted to avoid ongoing costs. This involved deleting resources in a specific order due to interdependencies:

1. Terminated all EC2 instances (Public-Windows-Server-01-New, Private-Ubuntu-Server-01, and any other project-related instances).
2. Released any allocated Elastic IP addresses.
3. Deleted any Load Balancers or NAT Gateways.
4. Disassociated subnets from custom route tables.
5. Deleted custom route tables.
6. Deleted all custom Security Groups (Public-Windows-SG-New, Private-Ubuntu-SG-New).

7. Detached and deleted the Internet Gateway (MinorProjectVPC-igw).
8. Deleted the custom subnets (my-publicsubnet-01, my-privatesubnet-01).
9. Finally, deleted the custom VPC (MinorProject-vpc).

5. Conclusion

This project successfully demonstrated the creation of a secure and segmented cloud network architecture on AWS. We effectively deployed a publicly accessible web server and a privately isolated backend server, utilizing security groups and a jump host model for controlled access. The experience reinforced key concepts in VPC networking, EC2 instance management, and network security best practices in a cloud environment.