**CUSTOM AWS VPC SETUP WITH PUBLIC AND PRIVATE SUBNETS FOR SECURE INFRASTRUCTURE**

**Project Summary:**

*This project demonstrates the setup of a custom Virtual Private Cloud (VPC) architecture in AWS, designed to support a scalable, secure network infrastructure for a retail-based application. The scenario assumes a retail company launching a new e-commerce platform that requires both internet-facing services (such as a web front end) and backend systems (like inventory management) hosted in a private environment.*

*To meet these requirements, I created a VPC with a /16 CIDR block, subdivided into a public and a private subnet, each with 256 IPs. The public subnet is configured with an Internet Gateway to enable external access, while the private subnet is secured and connected via a NAT Gateway for controlled outbound internet access. Route tables and security groups were configured accordingly. I deployed EC2 instances in both subnets and enabled secure access to the private instance using AWS Systems Manager (SSM), ensuring a fully functional and secure environment for the retail application’s infrastructure.*

***Note: The project replicates a real-world VPC setup delivered for a retail client. While the configuration mirrors the original environment, sensitive client details have been omitted in this portfolio for privacy reasons.***

S

VPC

10.0.0.0/16

65536 IP

NAT

SG

SG

PRIV RT

PUB RT

IGW

PRIVATE SUBNET

10.0.2.0/24

256 IP

PUBLIC SUBNET

10.0.1.0/24

256 IP

1. Go to AWS Console → **VPC** → Create VPC → Select VPC only→Name: **Project-A** →IPv4 CIDR Block: **10.0.0.0/16**→Tenancy: Default→ finally click on Create VPC.

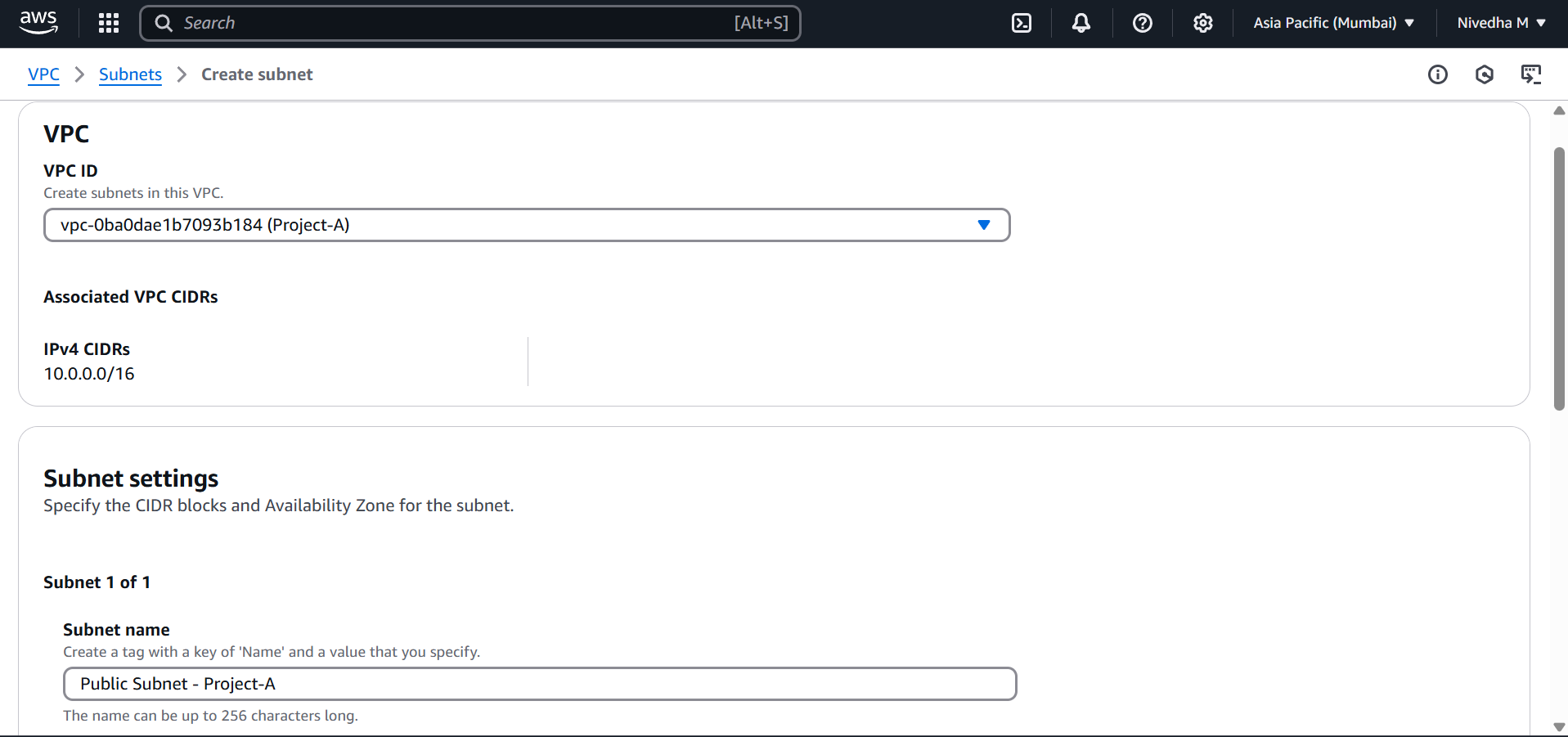
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create Subnet→ **Public Subnet →**Name: **Public Subnet-Project-A** →CIDR Block: **10.0.1.0/24**→Availability Zone: **ap-south-1a.**



A screenshot of a computer

AI-generated content may be incorrect.

1. Now create private subnet→ **Private Subnet** →Name: **Private Subnet-Project-A**→CIDR Block: **10.0.2.0/24**→Availability Zone: **ap-south-1b**.
2. Create and Attach **Internet Gateway (IGW)**→ For that go to → VPC → Internet Gateway → Create→Name: **IGW-Project-A**→Attach to created **VPC(Project-A)**→Create IGW

A screenshot of a computer

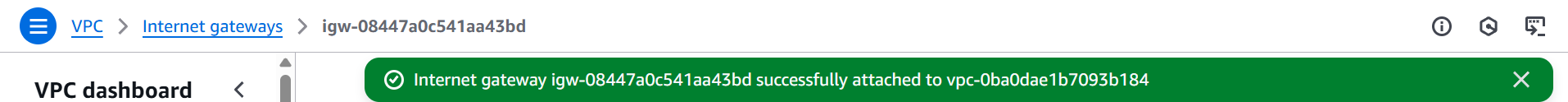
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



1. Create **Route Tables** → Give Name- **Public-RT-Project-A** → select the created **VPC(Project-A**)→ select create route tableA screenshot of a computer

   AI-generated content may be incorrect.
2. Create **Route Tables** → Give Name- **Private-RT-Project-A** → select the created **VPC(Project-A)→** select create route table.

A screenshot of a computer

AI-generated content may be incorrect.

1. Now in the Route Table → Select Public RT→ Click on **Subnet Associations** → select **Edit subnet associations**. In the available subnets, **select Public subnet**→ Save association. **Repeat the same for Private subnet also.**

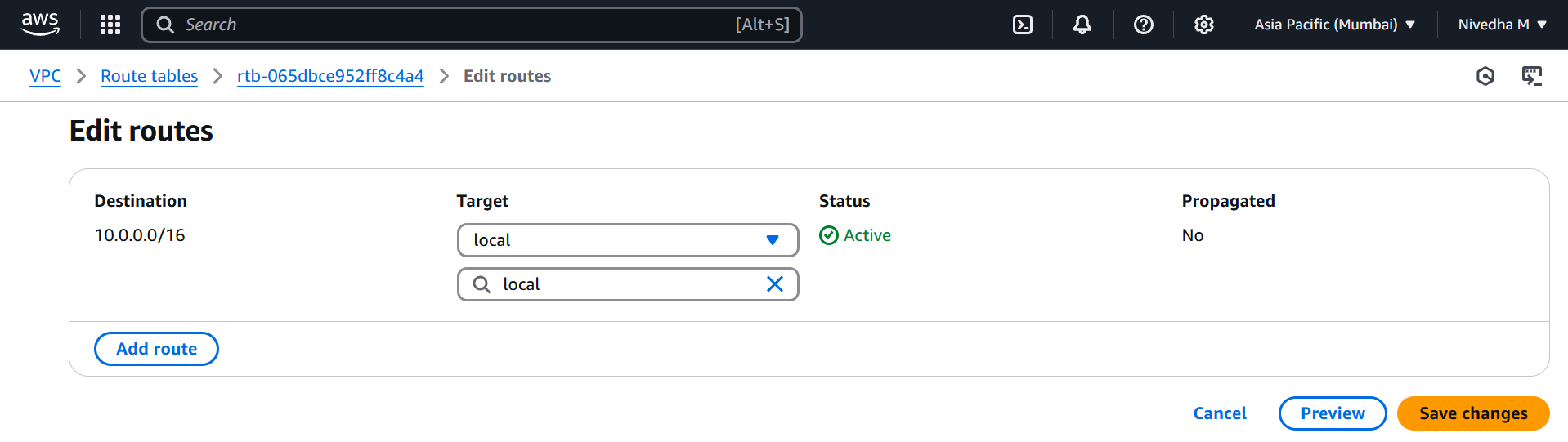
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Again, in Route Table Select public RT→ go to the **Routes** → Edit-> Routes →**Add routes→0.0.0.0/0→ select IGW and select the created IGW**→ save.



A screenshot of a computer

AI-generated content may be incorrect.

1. Create **Security Group**→ give SG name: **Public SG-Project-A**→ VPC : select the **created VPC(Project-A)**→ **Inbound rules: add SSH HTTP HTTPS RDP**→ **Outbound rules: all traffic**→ create SG. Once created, **Note down the public SG id for reference.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Repeat the same for private SG→ **Private SG-Project-A**→ Select the VPC created **(Project-A)** → Inbound rules→ select **All TCP**→ in Source Scroll down little bit and **select the created public SG**→ Save.

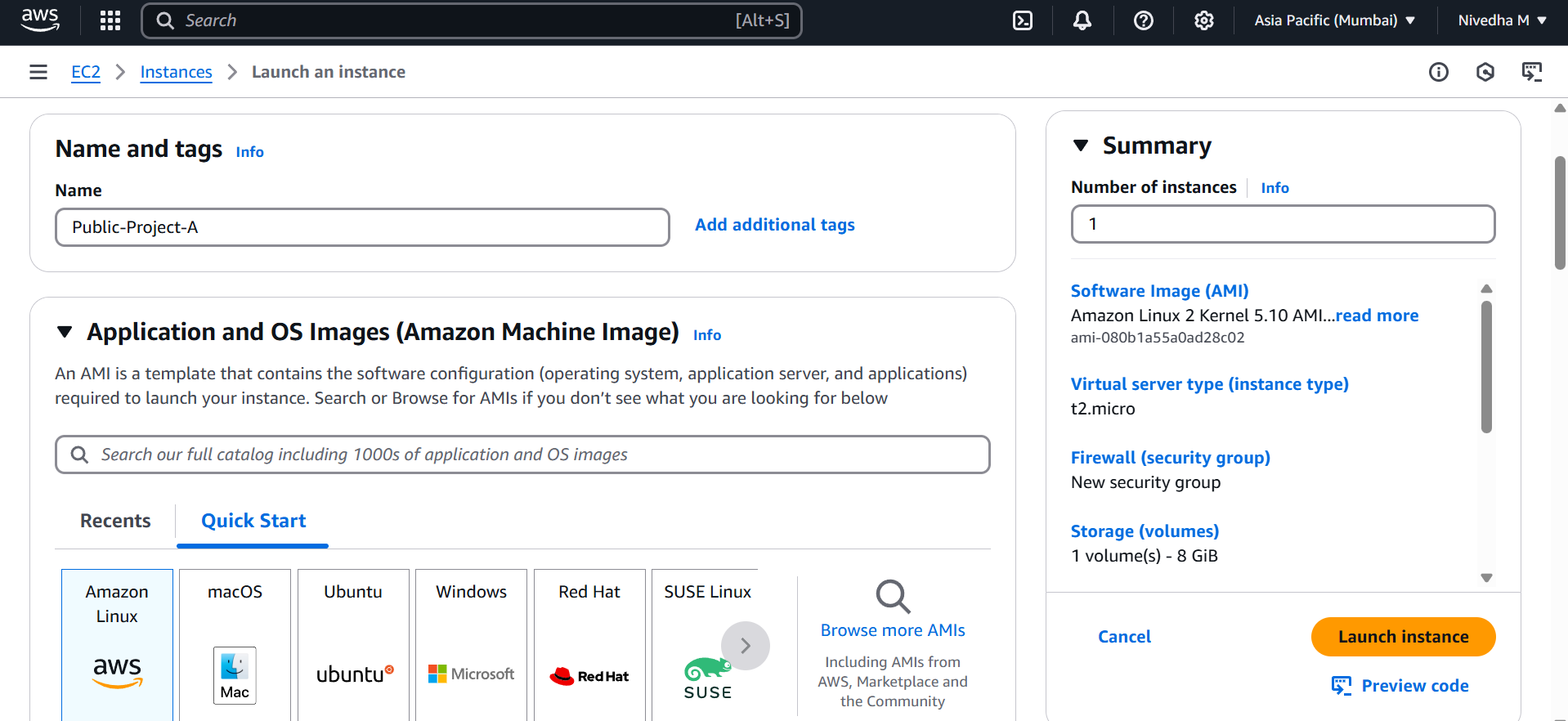
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create a **EC2 instance** → select Linux AMI→in network settings select **VPC created**→ select the **Public Subnet**→ **Auto assign public IP:Enable**→ **SG click on existing SG → select public SG**→ Launch.



A screenshot of a computer

AI-generated content may be incorrect.

1. Repeat the same for creating another in private EC2. In subnet select **private subnet→ auto assign public IP: Disable** → SG click on existing SG→ select public SG→ Launch.

A screenshot of a computer

AI-generated content may be incorrect.

1. Now if you login into Public EC2 with public IP and check if you can access the internet from inside.

A computer screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. To get internet in private EC2 machine we need to Create **NAT**. Goto NAT Gateway→ create→ Name: **NAT-Project-A**→ select **the public subnet(source for the internet)→ Allocate Elastic IP**→ create NAT Gateway.  
   A screenshot of a computer

   AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Now go to VPC→ Select Route Table→ select **Private RT→ Rotes→Edit Routes: 0.0.0.0/0→ in target select NAT Gateway** and select the created one→ Save.

A screenshot of a computer

AI-generated content may be incorrect.

1. Now to login in to private EC2 we can use SSM. Make sure our local PC is installed with SSM plugin.
2. Go to AWS console**: IAM→ Roles→ Create Roles→ Trusted entity → EC2→Attach Policy → AmazonSSMManagedInstanceCore →Role name: SSM-Access-Role→ Create.**
3. Now got to **EC2→ Select Private EC2 → Actions → Security → Modify IAM Role → Choose SSM-Access-Role → Update**.
4. Then Open Powershell→ type AWS configure→ it will ask for Access key and secret access key. Click on AWS Console right top corner → Account Name / Email ID → Select → Security Credentials→ Scroll down → Access Keys section varum→Click → Create Access Key.

A screenshot of a computer

AI-generated content may be incorrect.

1. Then type **aws ssm start-session --target i-0857444c77ad6eb84 --region ap-south-1.** Now we are into private EC2 instance, and we are accessible to internet.

A screenshot of a computer

AI-generated content may be incorrect.

**Troubleshooting Steps**

* While logging in to Private ec2 via powershell I got the below errors

PS C:\Users\NivGokDhiv> aws ssm start-session --target i-0857444c77ad6eb84 --region ap-south-1 SessionManagerPlugin is not found. Please refer to SessionManager Documentation here: http://docs.aws.amazon.com/console/systems-manager/session-manager-plugin-not-found PS C:\Users\NivGokDhiv>

* So I downloaded plugins.
* Again I faced

PS C:\Users\NivGokDhiv> session-manager-plugin session-manager-plugin : The term 'session-manager-plugin' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again. At line:1 char:1 + session-manager-plugin + ~~~~~~~~~~~~~~~~~~~~~~ + CategoryInfo : ObjectNotFound: (session-manager-plugin:String) [], CommandNotFoundException + FullyQualifiedErrorId : CommandNotFoundException

* To rectify it I added the environment variables manually.
* Right click *This PC* → Properties → Advanced system settings → Environment Variables → System Variables → Path → Edit → New(Add→ C:\Program Files\Amazon\SessionManagerPlugin\bin)→ Save & Ok.
* Close the powershell and open a new window→ and check again→session-manager-plugin