

AWS – GCP VPN CONNECTION



AUGUST 17

Devamadhav Kattimuttathu



Creating a hybrid network and routes between two clouds

Introduction

In this lab, we will set up AWS and Google Cloud environments to configure a hybrid cloud connection between them. We will create and isolate one Virtual Private Cloud (VPC) in AWS and one in Google Cloud Platform (GCP). After configuring the VPCs, we will establish routing to manage traffic between them. Additionally, we will set up a site-to-site VPN to ensure secure communication between both cloud environments.

Observations and Screenshots

We began the lab by creating a VPC in AWS. First, we navigated to the VPC section in the AWS Management Console and selected the option to create a new VPC. We named it dk5197-VPC1 and set the IP address range to 10.30.0.0/20. The VPC was successfully created.

Next, we created two subnets within this VPC. We named them subnet01 and subnet02, assigning them IP ranges 10.30.0.0/24 and 10.30.0.128/24, respectively. We then created a new Route Table named dk5197-RT to manage network traffic, associating it with dk5197-VPC1. In the subnet association of the Route Table, we explicitly associated the two subnets. This configuration isolates the networks and allows for more efficient routing.

Following this, we created an Internet Gateway named dk5197-IG and successfully attached it to our VPC. Finally, we updated the Route Table to include a new route with a destination of 0.0.0.0/0, which allows all outbound traffic to pass through the Internet Gateway.

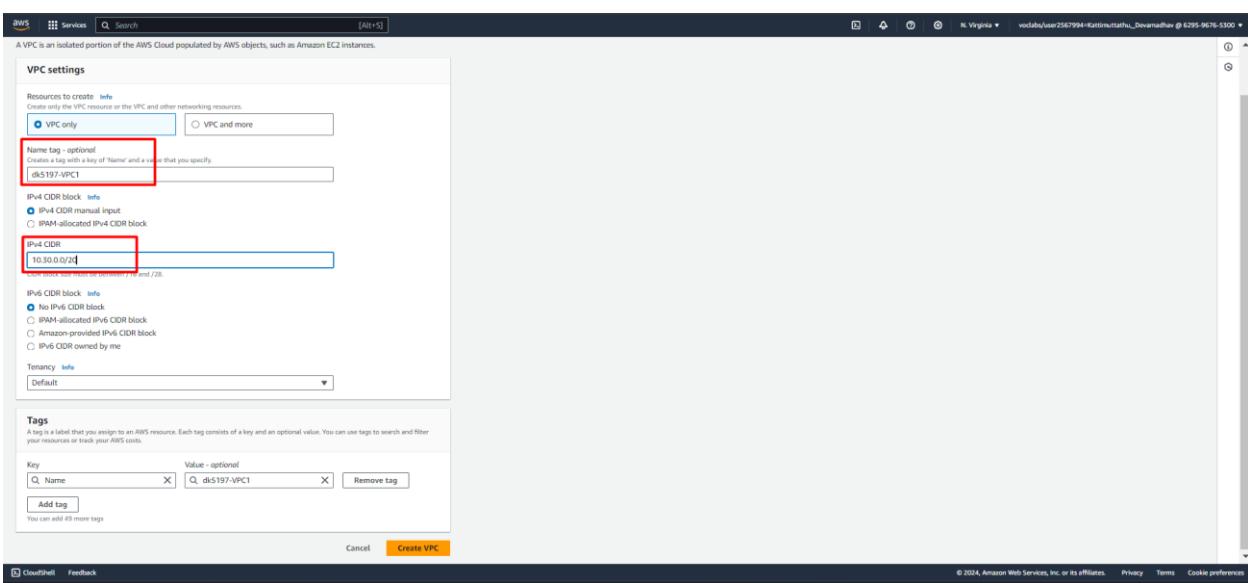


Figure 1 - Creating a new VPC in AWS

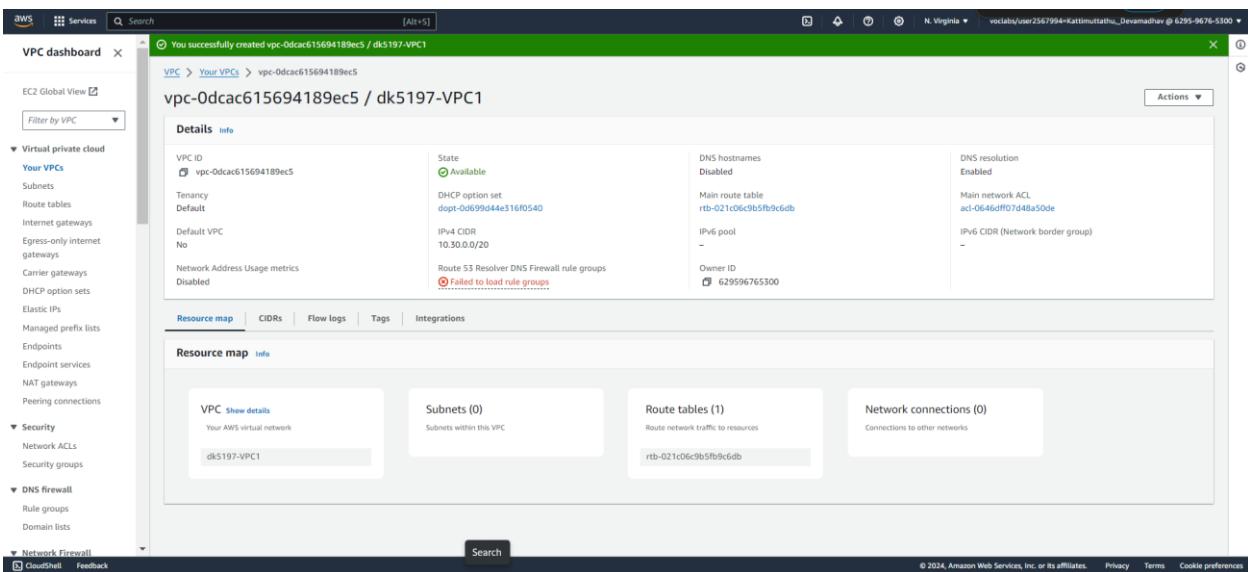


Figure 2 - Successfully created a new VPC named dk5197-VPC1

The screenshot shows the AWS Subnet settings page. The subnet name is set to "subnet01". The availability zone is set to "No preference". The IPv4 CIDR block is set to "10.30.0.0/20". An optional tag "Name" is added with the value "subnet01". The "Create subnet" button is highlighted in orange.

Figure 3 - Creating the first subnet for the VPC

The screenshot shows the AWS Subnet settings page. The subnet name is set to "subnet02". The availability zone is set to "No preference". The IPv4 CIDR block is set to "10.30.1.0/24". An optional tag "Name" is added with the value "subnet02". The "Create subnet" button is highlighted in orange.

Figure 4 - Creating the second subnet for the VPC

The screenshot shows the AWS VPC dashboard with the Subnets table. The table lists eight subnets, each with a name, subnet ID, state, VPC, IPv4 CIDR, IPv6 CIDR, available IPv4 addresses, and availability zone. Two subnets, "subnet01" and "subnet02", are highlighted with a red border.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addresses	Availability
-	subnet-0fae50b7c026e630b	Available	vpc-0748a41bf4f617f14	172.31.0.0/20	-	4091	us-east-1
-	subnet-0796664b2a6147fa6	Available	vpc-0748a41bf4f617f14	172.31.48.0/20	-	4091	us-east-1
-	subnet-056c5ba9d5d9d0b0e	Available	vpc-0748a41bf4f617f14	172.31.16.0/20	-	4091	us-east-1
-	subnet-07b72106530x7270	Available	vpc-0748a41bf4f617f14	172.31.64.0/20	-	4091	us-east-1
-	subnet-058aa70268ac25349	Available	vpc-0748a41bf4f617f14	172.31.80.0/20	-	4091	us-east-1
-	subnet-057bd1aa53a1823	Available	vpc-0748a41bf4f617f14	172.31.32.0/20	-	4091	us-east-1
subnet01	subnet-09207dc78e7599	Available	vpc-06cac615694189ec5 dk51...	10.30.0.0/24	-	251	us-east-1
subnet02	subnet-068c1489aa37ea03	Available	vpc-06cac615694189ec5 dk51...	10.30.1.0/24	-	251	us-east-1

Figure 5 - Successfully created two subnets for the VPC in AWS

The screenshot shows the AWS VPC > Route tables > Create route table interface. It displays the "Route table settings" section where a route table named "dk5197-RT" is being created. The "VPC" dropdown is set to "vpc-06cac615694189ec5 (dk5197-VPC1)". Under the "Tags" section, a single tag "Q_Name:dk5197-RT" is added. The "Create route table" button is highlighted in orange at the bottom right.

Figure 6 - Creating a route table in AWS

The screenshot shows the AWS VPC dashboard with a successful creation message: "Route table rtb-0c7a471bfba56eb8b | dk5197-RT was created successfully." The main pane displays the details of the newly created route table, including its ID (rtb-0c7a471bfba56eb8b), Main status (No), and Owner ID (629596765300). The "Routes" tab shows one route entry: Destination 10.30.0.0/20, Target local, Status Active, and Propagated No. The sidebar on the left lists various VPC-related services and options.

Figure 7 - Successfully created new route table for VPC

The screenshot shows the AWS VPC dashboard after adding explicit subnet associations. A success message indicates: "You have successfully updated subnet associations for rtb-0c7a471bfba56eb8b / dk5197-RT." The main pane now shows the "Subnet associations" tab selected, displaying two explicit subnet associations: subnet01 and subnet02. Each association is linked to the IPv4 CIDR 10.30.0.0/24. A red box highlights this section. Below it, the "Subnets without explicit associations" section shows zero subnets. The sidebar remains the same as in Figure 7.

Figure 8 - Added our two subnets as explicit subnet associations

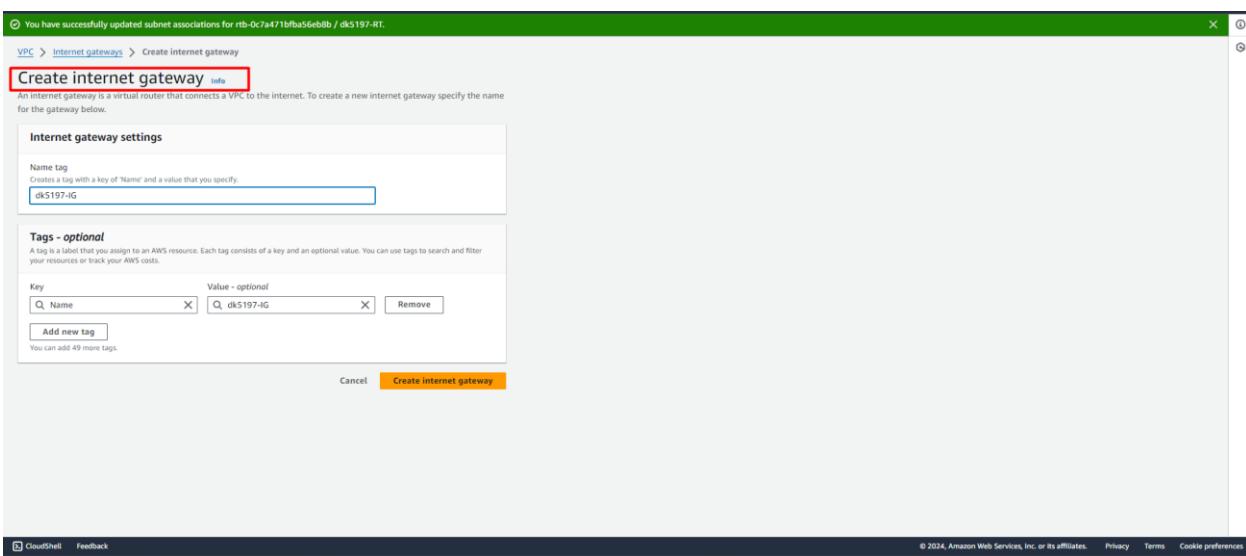


Figure 9 - Creating a new internet gateway for AWS VPC

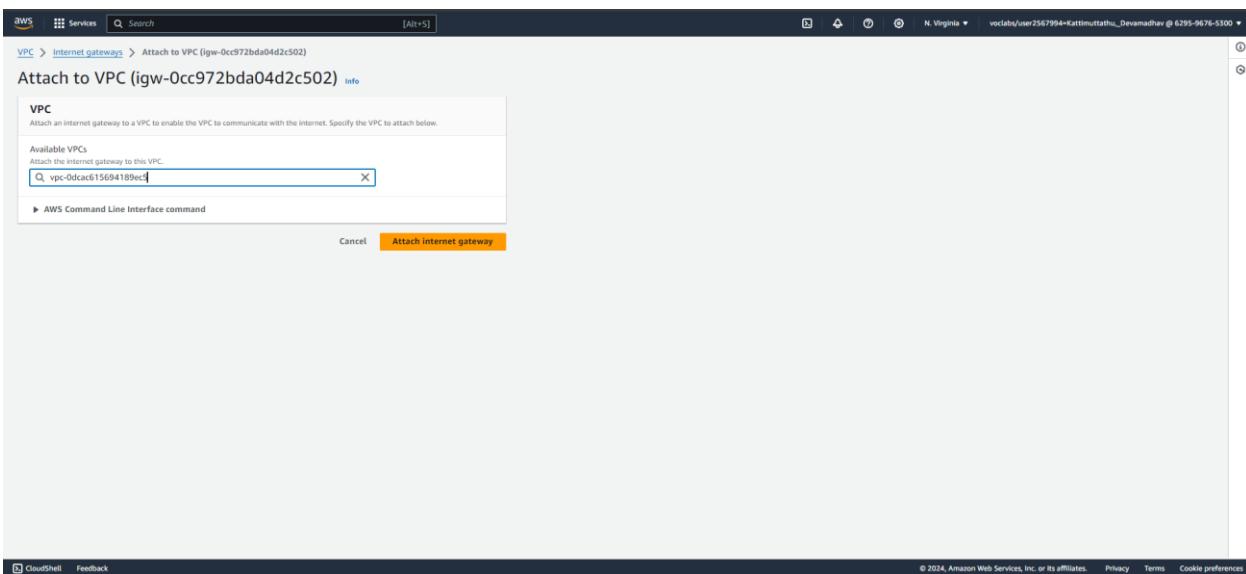


Figure 10 - Attaching the internet gateway to the VPC

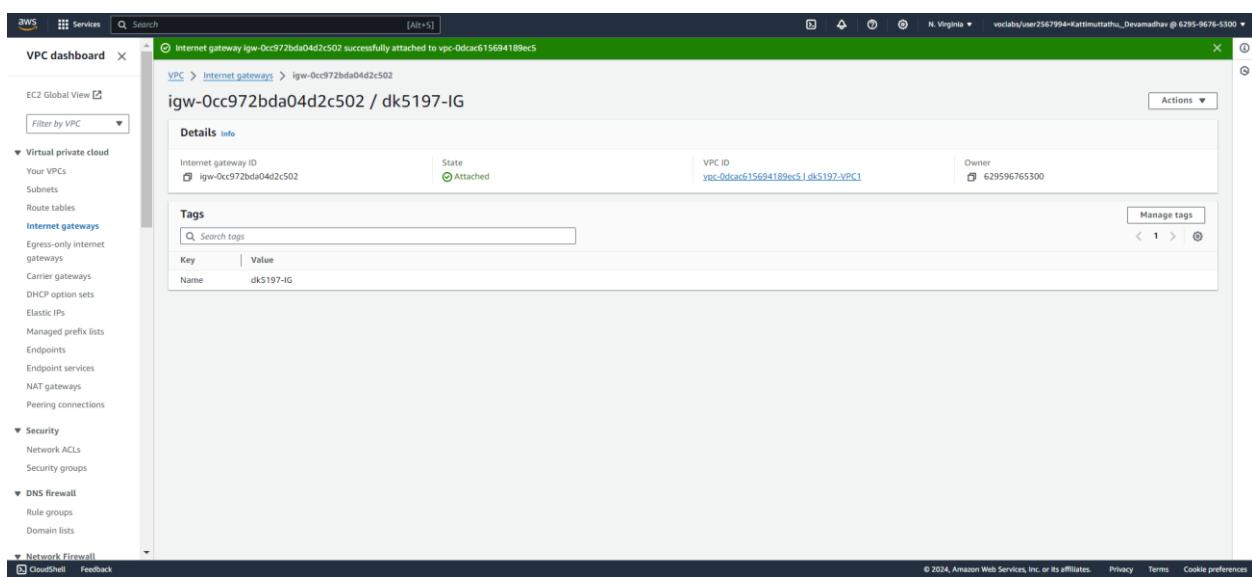


Figure 11 - Successfully attached the internet gateway to our vpc

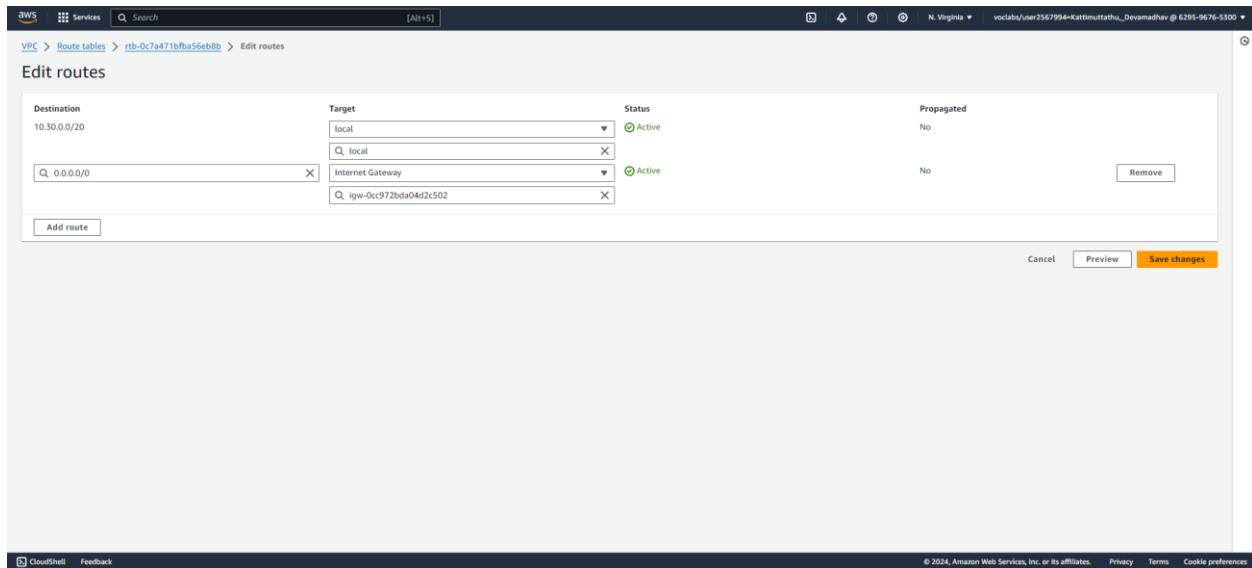


Figure 12 - Editing the route table to enable all traffic to internet through internet gateway

The screenshot shows the AWS VPC Route Tables interface. On the left, a sidebar lists various VPC-related services like EC2 Global View, Filter by VPC, Virtual private cloud, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections, Security (Network ACLs, Security groups), DNS firewall (Rule groups, Domain lists), and Network Firewall (CloudShell, Feedback). The main panel displays a route table named 'rtb-0c7a471bfba56eb8b / dk5197-RT'. The 'Routes' tab is selected, showing two routes listed in a table:

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0cc972bd04d2c502	Active	No
10.30.0.0/20	local	Active	No

A red box highlights the second row of the table.

Figure 13 - Successfully created a new route

Next, we created a new web server named dk5197-server with Linux as the operating system. A new key pair was generated for the system. We configured the network by selecting the VPC and assigning subnet01 to the instance. The instance was successfully launched, and we connected to it using SSH.

Upon logging into the system, we installed the Apache web server. After the installation, we created a new webpage and copied it to the appropriate directory. To ensure the webpage was accessible over the internet, we added an additional inbound rule allowing HTTP access from any IP address (0.0.0.0/0). This configuration enabled us to successfully display our webpage using the instance's public IP address.

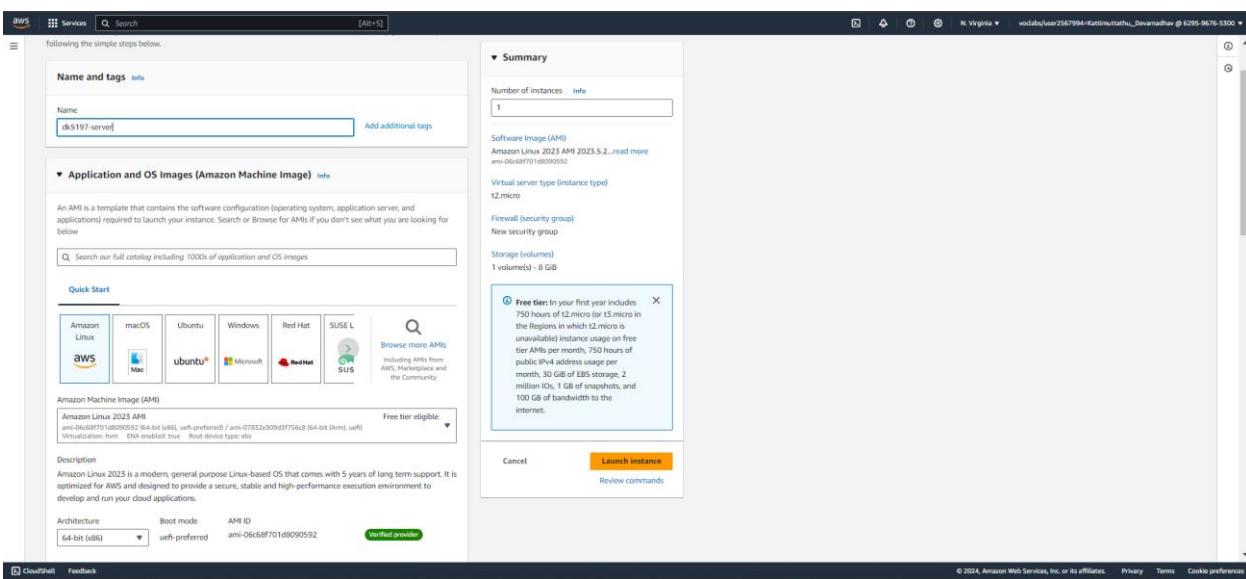


Figure 14 - Creating a new instance in AWS for our webserver

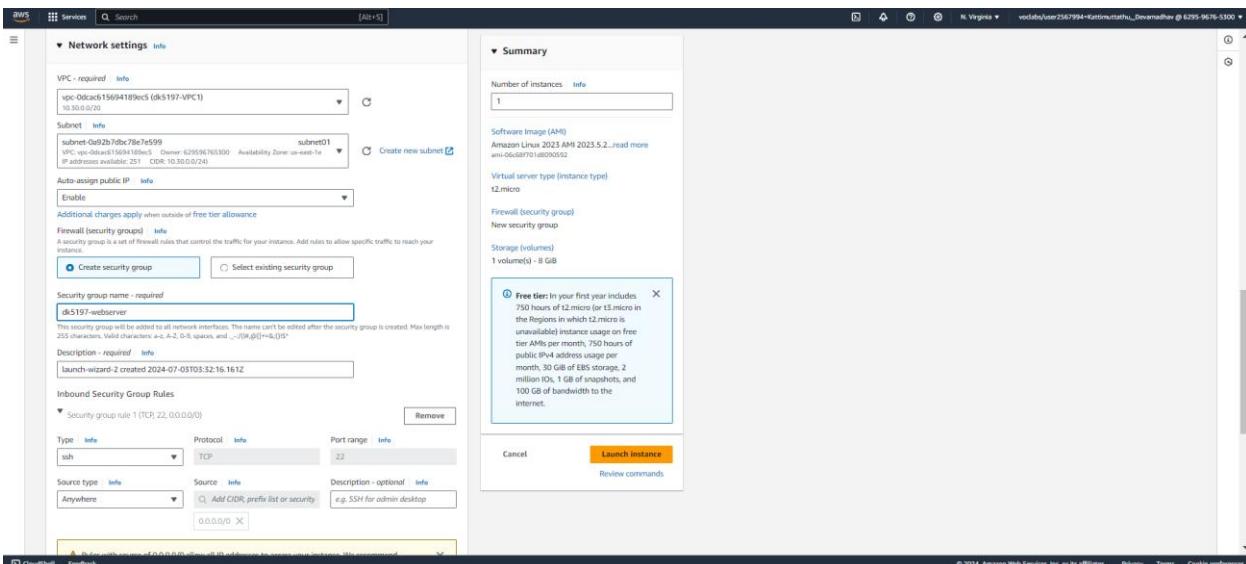


Figure 15 - Configuring the network settings in our VPC

Instance summary for i-07a11d0e368cec106 (dk5197-server) [info](#)

Updated less than a minute ago

Instance ID	i-07a11d0e368cec106 (dk5197-server)	Public IPv4 address	100.26.170.98 open address
IPv6 address	-	Instance state	Pending
Hostname type	IP name: ip-10-30-0-214.ec2.internal	Private IP DNS name (IPv4 only)	ip-10-30-0-214.ec2.internal
Answer private resource DNS name	-	Instance type	t2.micro
Auto-assigned IP address	100.26.170.98 [Public IP]	VPC ID	vpc-0dcac615694189ec5 (dk5197-VPC1)
IAM Role	-	Subnet ID	subnet-0a92b7dbc78e7e599 (subnet01)
IMDsv2	Required	Instance ARN	arn:aws:ec2:us-east-1:629596765300:instance/i-07a11d0e368cec106

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

[Instance details](#) [info](#)

Platform	Amazon Linux (Inferred)	AMI ID	ami-06c68f701d8090592	Monitoring	disabled
Platform details	Linux/UNIX	AMI name	al2023-ami-2023.5.20240701.0-kernel-6.1-x86_64	Termination protection	Disabled
Stop protection	Disabled	Launch time	Tue Jul 02 2024 23:37:27 GMT-0400 (Eastern Daylight Time) (less than a minute)	AMI location	amazon/al2023-ami-2023.5.20240701.0-kernel-6.1-x86_64
Placement Groups	-	Lifecycle	normal	Stop-hibernate behavior	Disabled
Key Pairs	Mathew's Key Pair	Default	-	-	-

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Figure 16 - Displaying the details of our instance

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-10-30-0-214 ~]#

```

i-07a11d0e368cec106 (dk5197-server)
PublicIPs: 100.26.170.98 PrivateIPs: 10.30.0.214

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Figure 17 - Successfull SSH connection to our instance

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

Dependencies resolved.

Package          Architecture      Version       Repository  Size
Installing:
httpd            x86_64          2.4.59-2.amzn2023
Installing dependencies:
apr              x86_64          1.7.2-2.amzn2023.0.2
apr-util         x86_64          1.6.3-1.amzn2023.0.1
apr-util-openssl x86_64          1.6.3-1.amzn2023.0.3
httpd-core       x86_64          2.4.59-2.amzn2023
httpd-fs system  noarch         2.4.59-2.amzn2023
httpd-tools      x86_64          2.4.59-2.amzn2023
libaprutil1     x86_64          1.0.9-4.amzn2023.0.2
mailcap          noarch         2.1.1-9.amzn2023.0.3
Installing weak dependencies:
apr-util-openssl x86_64          1.6.3-1.amzn2023.0.1
mod_http2        x86_64          2.0.1-1.amzn2023.0.2
mod_lua          x86_64          2.4.59-2.amzn2023

Transaction Summary
Install 12 Packages

Total download size: 2.3 M
Installed size: 6.0 M
Is this ok [y/N]: ■

i-07a11d0e368cec106 (dk5197-server)
PublicIPs: 100.26.170.98 PrivateIPs: 10.30.0.214

```

Figure 18 - Installing apache server in our instance

```

Preparing :
Installing : apr-1.7.2-2.amzn2023.0.2.x86_64
Installing : apr-util-1.6.3-1.amzn2023.0.1.x86_64
Installing : mailcap-2.1.49-3.amzn2023.0.3.noarch
Installing : httpd-tools-2.4.59-2.amzn2023.x86_64
Installing : libaprutil1-1.0.9-4.amzn2023.x86_64
Running scriptlet: httpd-fs system-2.4.59-2.amzn2023.noarch
Installing : httpd-fs system-2.4.59-2.amzn2023.noarch
Installing : httpd-core-2.4.59-2.amzn2023.x86_64
Installing : mod_lua-2.4.59-2.amzn2023.x86_64
Installing : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
Installing : httpd-2.4.59-2.amzn2023.x86_64
Running scriptlet: httpd-tools-2.4.59-2.amzn2023.noarch
Verifying : apr-1.7.2-2.amzn2023.0.2.x86_64
Verifying : apr-util-1.6.3-1.amzn2023.0.1.x86_64
Verifying : mailcap-2.1.49-3.amzn2023.0.3.noarch
Verifying : httpd-fs system-2.4.59-2.amzn2023.noarch
Verifying : httpd-core-2.4.59-2.amzn2023.x86_64
Verifying : mod_lua-2.4.59-2.amzn2023.x86_64
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
Verifying : httpd-2.4.59-2.amzn2023.x86_64
Verifying : libaprutil1-1.0.9-4.amzn2023.0.2.x86_64
Verifying : mailcap-2.1.49-3.amzn2023.0.3.noarch
Verifying : mod_http2-2.0.1-1.amzn2023.x86_64
Verifying : mod_lua-2.4.59-2.amzn2023.x86_64

Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64      apr-util-1.6.3-1.amzn2023.0.1.x86_64      apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64      generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch  httpd-2.4.59-2.amzn2023.x86_64
httpd-core-2.4.59-2.amzn2023.x86_64    httpd-fs system-2.4.59-2.amzn2023.noarch  httpd-tools-2.4.59-2.amzn2023.x86_64    libaprutil1-1.0.9-4.amzn2023.0.2.x86_64   mailcap-2.1.49-3.amzn2023.0.3.noarch
mod_http2-2.0.1-1.amzn2023.x86_64     mod_lua-2.4.59-2.amzn2023.x86_64

Created!
[root@ip-10-30-0-214 ec2-user]# vi index.html
[root@ip-10-30-0-214 ec2-user]# mv index.html /var/www/html/
[root@ip-10-30-0-214 ec2-user]# ls /var/www/html/
index.html
[root@ip-10-30-0-214 ec2-user]# 

i-07a11d0e368cec106 (dk5197-server)
PublicIPs: 100.26.170.98 PrivateIPs: 10.30.0.214

```

Figure 19 - Created a new webpage and copied to the corresponding folder

```
[root@ip-10-30-0-214 ec2-user]# curl localhost
<html lang="en">
<head>
<title>My Webpage</title>
<style>
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin-top: 20px;
}
h1 {
    color: #333;
}
</style>
</head>
<body>
<h1>DEVARADHAW KATTIMUTTATHU</h1><br>
<h2>CONESTOGA COLLEGE</h2>
</body>
</html>

[root@ip-10-30-0-214 ec2-user]# vi index.html
[root@ip-10-30-0-214 ec2-user]# vi /var/www/html/index.html
[root@ip-10-30-0-214 ec2-user]# curl localhost
<html lang="en">
<head>
<title>My Webpage</title>
<style>
body {
    background: Gray;
    font-family: Arial, sans-serif;
    text-align: center;
    margin-top: 20px;
}
h1 {
    color: #333;
}
</style>
</head>
<body>
<h1>DEVARADHAW KATTIMUTTATHU</h1><br>

```

i-07a11d0e368cec106 (dk5197-server)
PublicIP: 100.26.170.98 PrivateIP: 10.30.0.214

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 20 - Curl command to localhost

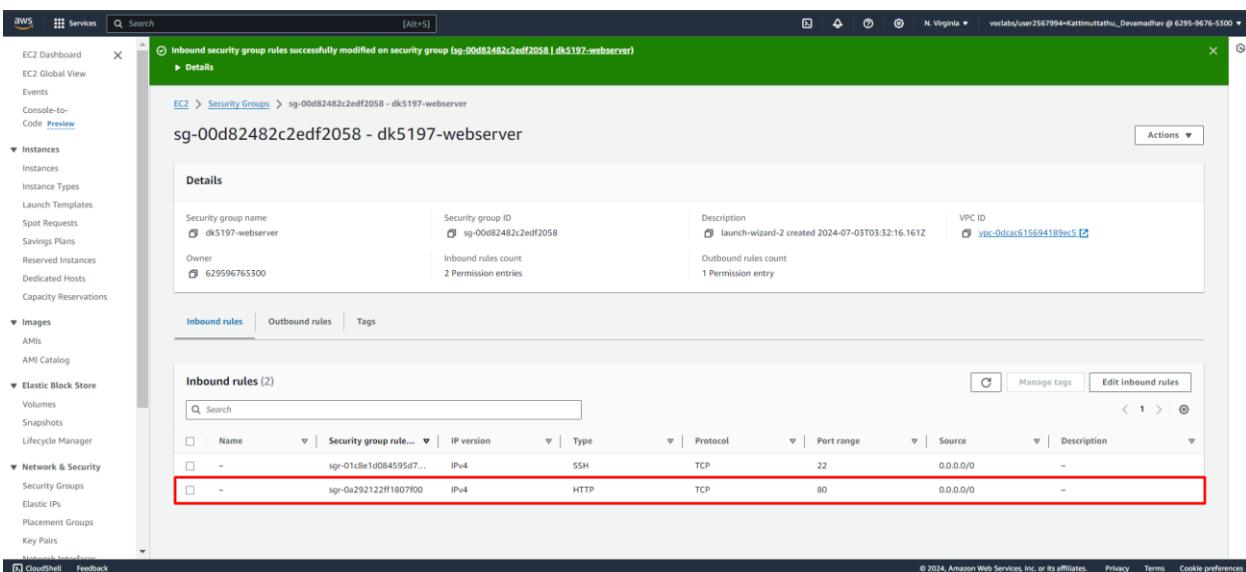


Figure 21 - Enabling HTTP traffic for our instance

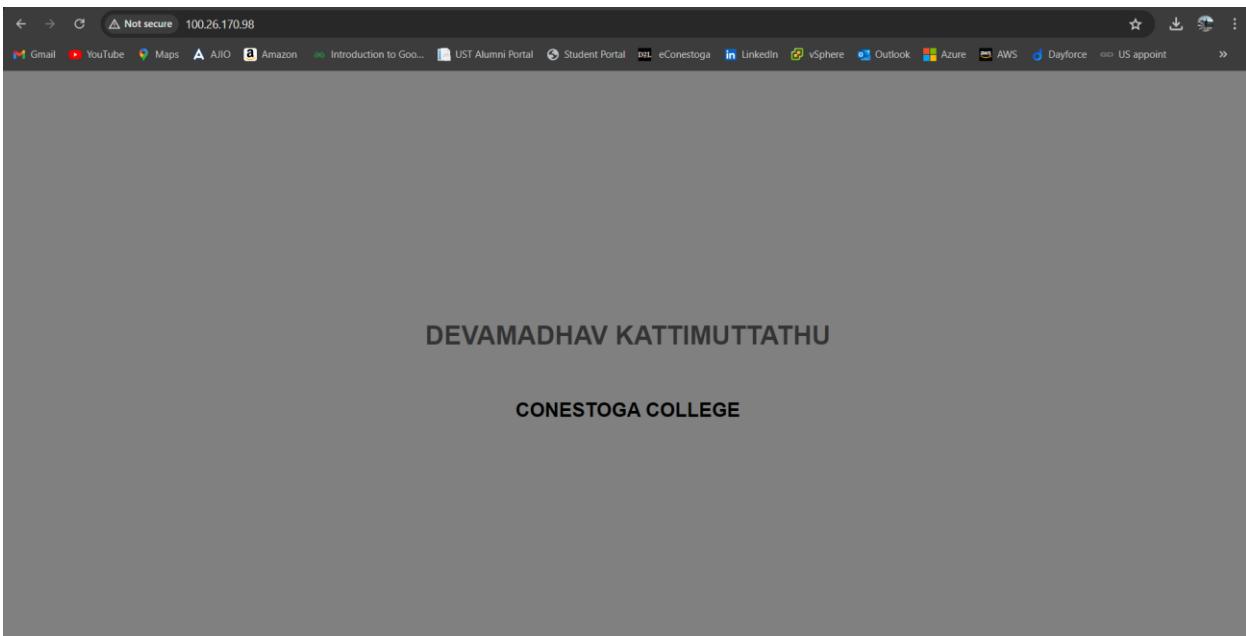


Figure 22 - Successfully access our webpage

Next, we navigated to the Google Cloud Platform (GCP) console to create a new Virtual Private Cloud (VPC). We named it dk5197-vpc2 and added two subnets. The first subnet, named sub1, was configured in the us-east1 region (matching the AWS region) with an IP range of 10.173.46.0/25. The second subnet, named sub2, was configured with an IP range of 10.173.46.128/25. After configuring the subnets, we set up firewall rules to allow custom ingress traffic for ICMP, RDP, and SSH. We also enabled global dynamic routing mode for the VPC before finalizing its creation. The VPC was successfully created.

Next, we navigated to Compute Engine and created a new instance named dk5197-gcp-server in the us-east1 region. We used the newly created VPC and sub1 for the instance, leaving the rest of the settings as default. After launching the instance, we attempted to SSH into it and ping the AWS instance using the command ping 10.30.0.214. However, there was no response because the security group in AWS had not yet been configured to allow ICMP traffic from the GCP network.

To address this, we navigated to the inbound rules in the AWS security group and added an ICMP rule for the GCP network. Although the rule was successfully added, we still couldn't get a response because AWS was not yet configured to connect to the GCP network. To complete the setup, we must establish a VPN connection for secure communication between the two cloud environments.

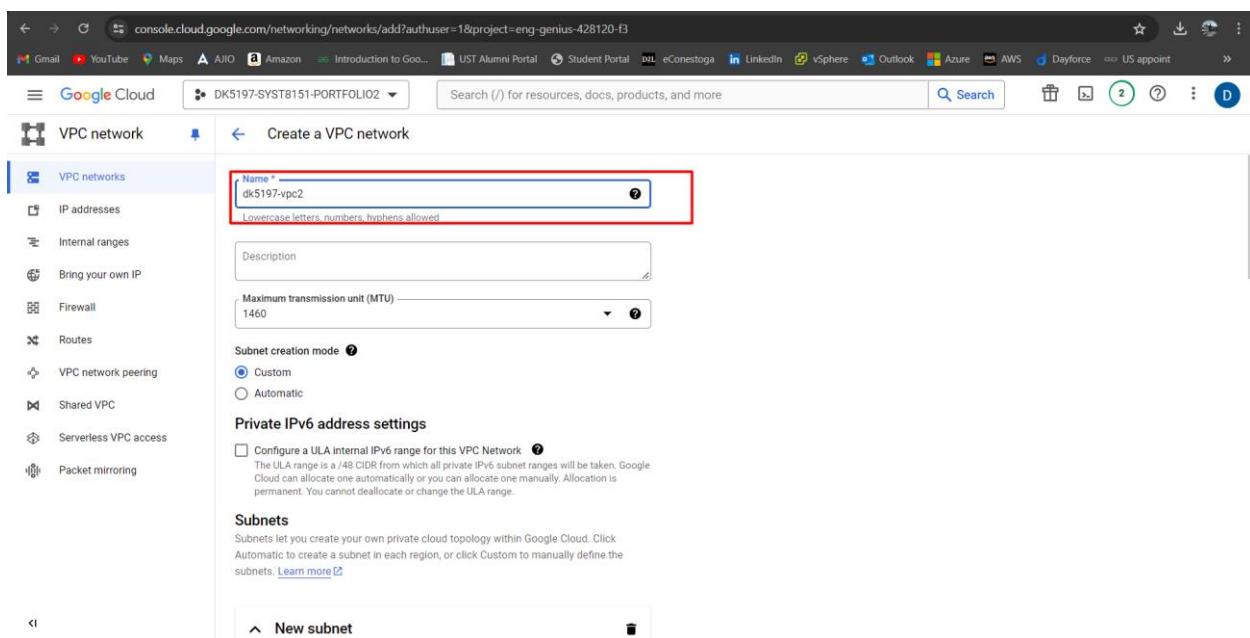


Figure 23 - Creating a new instance in gcp

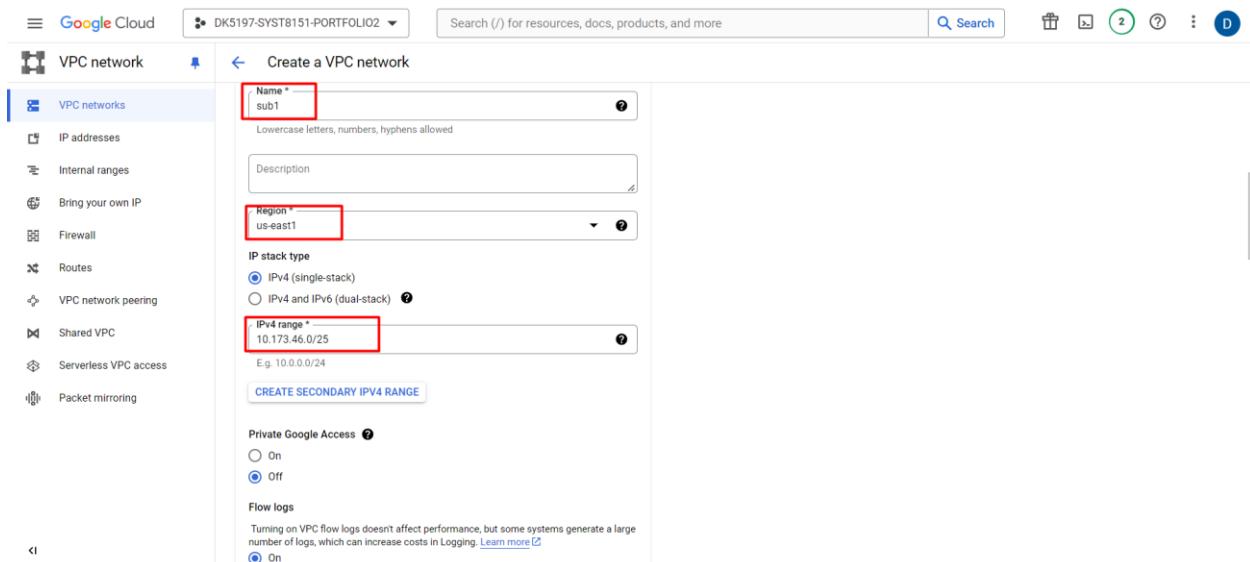


Figure 24 - Creating first subnet in our vpc

Google Cloud DK5197-SYST8151-PORTFOLIO2

Create a VPC network

VPC networks

Name * sub2

Description

Region * us-east1

IP stack type

- IPv4 (single-stack)
- IPv4 and IPv6 (dual-stack)

IPv4 range * 10.173.46.128/25

E.g. 10.0.0.0/24

CREATE SECONDARY IPV4 RANGE

Private Google Access

- On
- Off

Flow logs

Turning on VPC flow logs doesn't affect performance, but some systems generate a large number of logs, which can increase costs in Logging. [Learn more](#)

On

Figure 25 - Adding second subnet for our VPC

Google Cloud DK5197-SYST8151-PORTFOLIO2

Create a VPC network

VPC networks

IPV4 FIREWALL RULES

Name	Type	Targets	Filters	Protocols / ports	Action	Priority ↑	
dk5197-vpc2-allow-custom	Ingress	Apply to all	IP ranges: 10.173.46.0/25 10.173.46.128/25	all	Allow	65,534	EDIT
dk5197-vpc2-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65,534	
dk5197-vpc2-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65,534	
dk5197-vpc2-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65,534	
dk5197-vpc2-deny-all-ingress	Ingress	Apply to all	IP ranges: 0.0.0.0/0	all	Deny	65,535	
dk5197-vpc2-allow-all-egress	Egress	Apply to all	IP ranges: 0.0.0.0/0	all	Allow	65,535	

Advanced dynamic routing configuration

Dynamic routing mode

- Regional
Cloud Routers will learn routes only in the region in which they were created
- Global
Global routing lets you dynamically learn routes to and from all regions with a single VPN or interconnect and Cloud Router

DNS configuration (optional)

Figure 26 - Configuring the firewall for our VPC

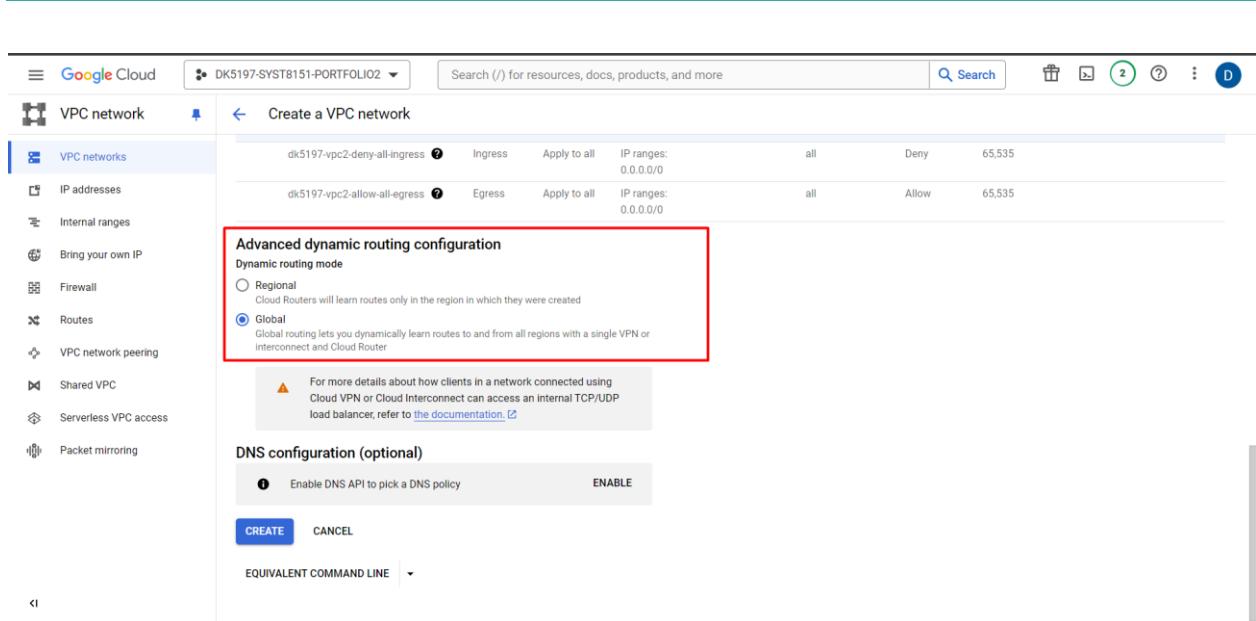


Figure 27 - Enabling global dynamic routing

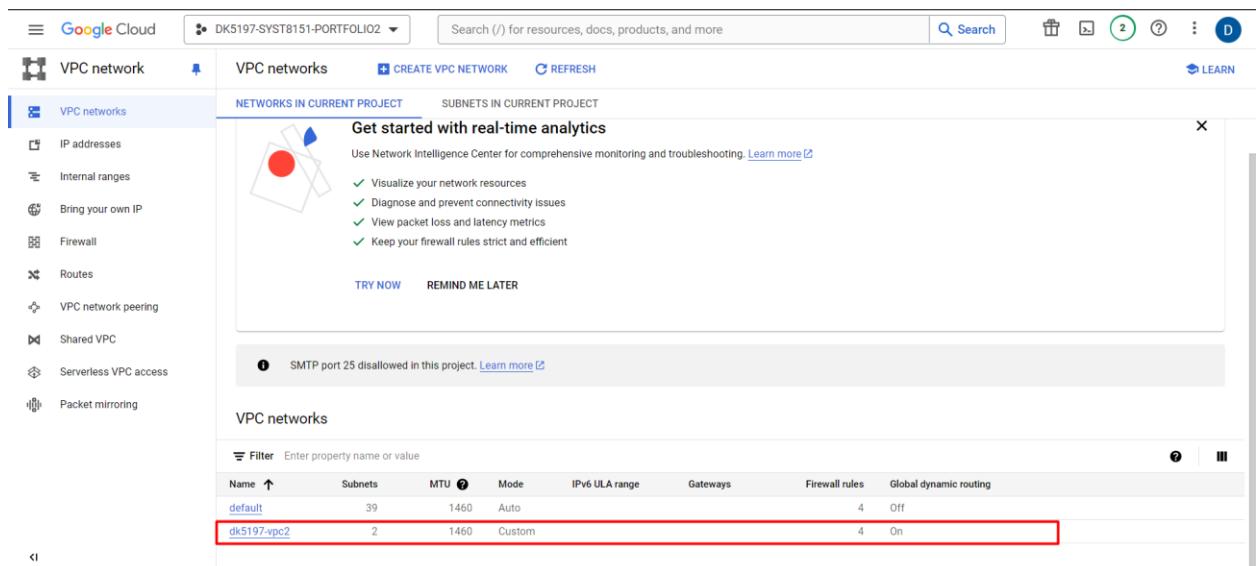


Figure 28 - Successfully created a new VPC in gcp

The screenshot shows the 'Create an instance' wizard in the Google Cloud Platform. The 'Name' field is set to 'dk5197-gcp-server'. The 'Region' is 'us-east1 (South Carolina)' and the 'Zone' is 'us-east1-b'. Under 'Machine configuration', the 'General purpose' tab is selected. A table lists various machine types (Series C4, N4, C3, C3D, E2, N2, N2D) with their descriptions, vCPUs, memory, and platform. The 'E2' row is highlighted with a blue background. At the bottom are 'CREATE', 'CANCEL', and 'EQUIVALENT CODE' buttons.

Figure 29 - Creating a new instance in GCP

The screenshot shows the 'Create an instance' wizard with 'CUSTOM' selected for 'Machine type'. The chosen machine type is 'e2-medium (2 vCPU, 1 core, 4 GB memory)'. Under 'ADVANCED CONFIGURATIONS', 'VM provisioning model' is set to 'Standard'. Under 'VM PROVISIONING MODEL ADVANCED SETTINGS', 'Display device' is enabled with the 'Enable display device' checkbox checked. At the bottom are 'CREATE', 'CANCEL', and 'EQUIVALENT CODE' buttons.

Figure 30 - Kept rest of the settings as default for the creation of instance

The screenshot shows the 'Create an instance' wizard in the Google Cloud Platform. The left sidebar lists options: 'New VM instance', 'New VM instance from template', 'New VM instance from machine image', and 'Marketplace'. The main configuration area is titled 'Boot disk' and includes fields for Name (dk5197-gcp-server), Type (New balanced persistent disk), Size (10 GB), License type (Free), and Image (Debian GNU/Linux 12 (bookworm)). A 'Monthly estimate' table shows costs for 2 vCPU + 4 GB memory (\$24.46), 10 GB balanced persistent disk (\$1.00), and a total of \$25.46. Below this is the 'Identity and API access' section, which includes a 'Service accounts' dropdown set to 'Compute Engine default service account' and an 'Access scopes' section with 'Allow default access' selected. At the bottom are 'CREATE', 'CANCEL', and 'EQUIVALENT CODE' buttons.

Figure 31 - leaving all other options as default for the creation of instance

This screenshot continues the 'Create an instance' wizard, focusing on the 'Edit network interface' step. It shows the selection of a 'Network' (dk5197-vpc2) and a 'Subnetwork' (sub1 IPv4 (10.173.46.0/25)). A note indicates that IPv6 subnet range is required. The 'IP stack type' is set to 'IPv4 (single-stack)'. Under 'Primary Internal IPv4 address', 'Ephemeral (Automatic)' is selected. In the 'Alias IP ranges' section, an 'External IPv4 address' is listed as 'Ephemeral'. The right side displays the same monthly cost breakdown as Figure 31. Buttons at the bottom include 'CREATE', 'CANCEL', and 'EQUIVALENT CODE'.

Figure 32 - Selecting our vpc and subnet1 to use as the insance network'

The screenshot shows the Google Cloud Compute Engine interface. On the left, a sidebar lists various services like Compute Engine, Storage, and Marketplace. The main area is titled 'VM instances' and shows a table of instances. One row is highlighted with a red border, corresponding to the instance name 'dk5197-gcp-server'. The table includes columns for Status, Name, Zone, Recommendations, In use by, Internal IP, External IP, and Connect (SSH). Below the table, there's a section titled 'Related actions' with links to 'Explore Backup and DR', 'View billing report', 'Monitor VMs', 'Explore VM logs', 'Set up firewall rules', 'Patch management', and 'Load balance between VMs'. To the right of the main content, there's a sidebar titled 'Get started with Compute Engine' containing several tutorial cards.

Figure 33 - Successfully created the new instance in gcp

This screenshot is similar to Figure 33, showing the Compute Engine VM instances page. However, a modal dialog box titled 'Authorize' is overlaid on the screen. The dialog asks 'Allow SSH-in-browser to connect to VMs.' and has two buttons: 'Authorize' and 'Cancel'. The background page shows the same list of VM instances, with 'dk5197-gcp-server' selected. The status bar at the bottom of the browser window indicates 'Establishing connection to SSH server...'.

Figure 34 - SSH connection to GCP instance

```

Linux dk5197-gcp-server 6.1.0-22-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.94-1 (2024-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

devamadhav0002@dk5197-gcp-server:~$ curl 10.30.0.214
^C
devamadhav0002@dk5197-gcp-server:~$ ping 10.30.0.214
PING 10.30.0.214 (10.30.0.214) 56(84) bytes of data.

```

Figure 35 - Tried to connect to the aws security group, but haven't received any response

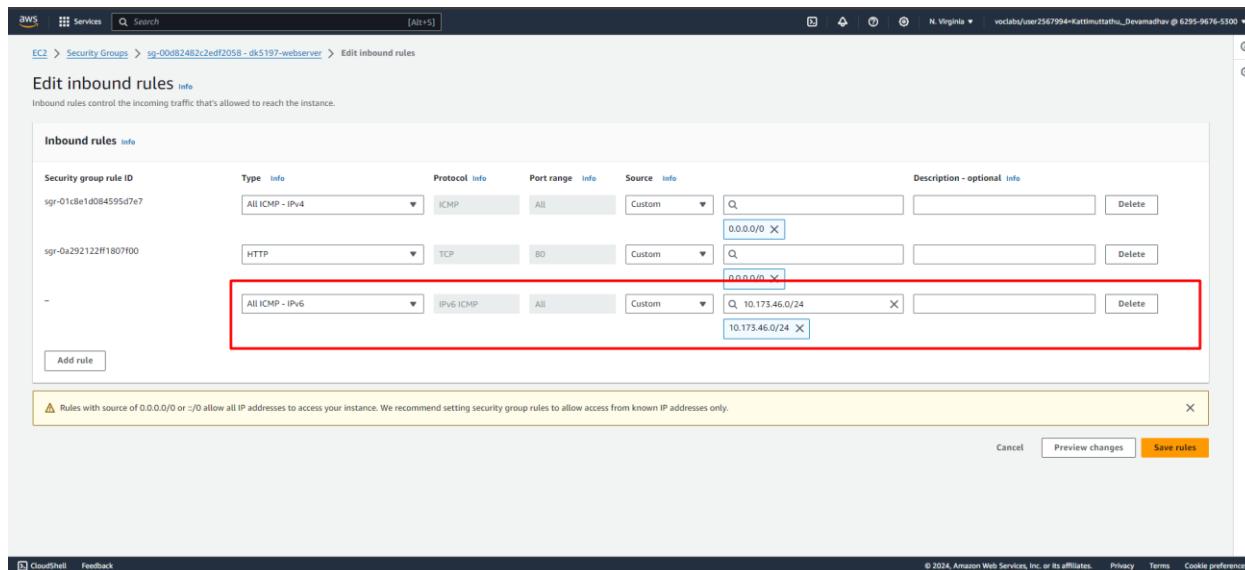


Figure 36 - Enabling the ICMP traffic in inbound rules for AWS

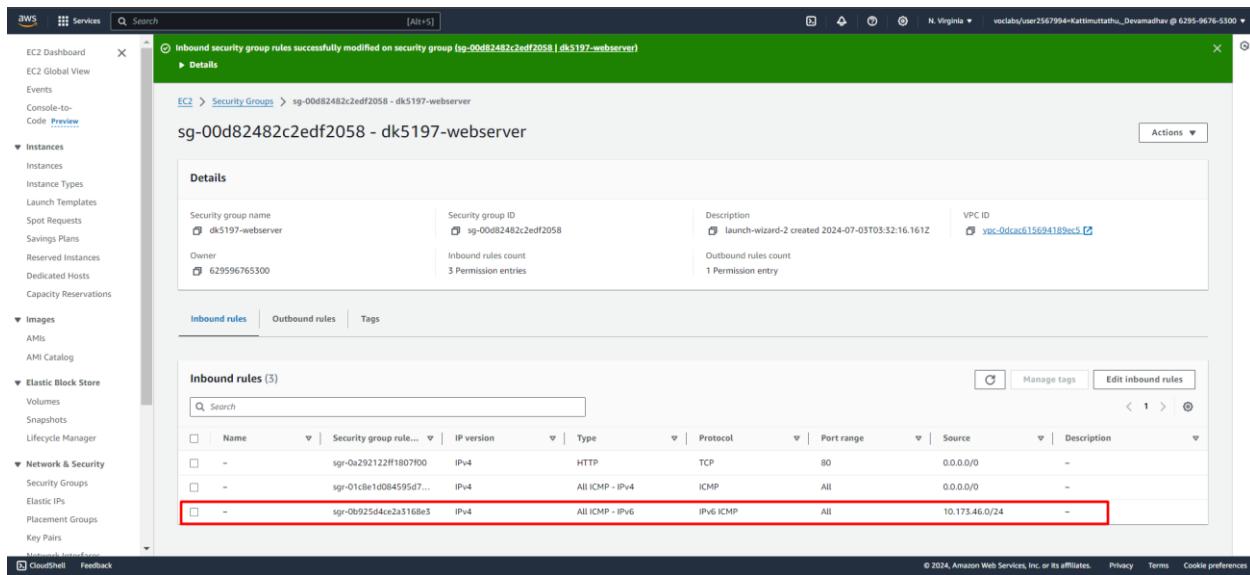


Figure 37 - Successfully added a new inbound rule in aws to allow ping

To establish the VPN connection, we first navigated to the GCP console and selected the Cloud Router option to create a new router for the VPN connection. We named the router gcp-router, selected our VPC network, and set the ASN (Autonomous System Number) value to 65000. The ASN is crucial for creating BGP (Border Gateway Protocol) sessions, which facilitate communication between different networks across providers. The router was successfully created.

Next, we navigated to the VPN section and chose the High Availability (HA) VPN option to create the tunnel. We created a new VPN gateway, associating it with our VPC and selecting the US-east1 region. Once the gateway was set up, our next task was to configure the VPN tunnel. GCP provides two public IP addresses for the VPN interfaces.

To complete the VPN setup, we also needed to define a non-Google Cloud peer VPN gateway, which meant configuring a similar VPN gateway on the AWS side. To do this, we navigated to the Virtual Private Gateway option in the AWS VPC console to begin setting up the corresponding gateway.

Google Cloud DK5197-SYST8151-PORTFOLIO02 vpn Search

Create a cloud router

Name * gcp-router

Description

Network * dk5197-vpc2

Region * us-east1 (South Carolina)

Google ASN * 65000

BGP peer keepalive interval seconds

BGP identifier E.g. 169.254.16.16/30. If not specified, Google will automatically-assign an IPv4 range.

Advertised routes

Routes

- Advertise all subnets visible to the Cloud Router (Default)
- Create custom routes

Figure 38 - Creating a router in gcp for VPN connection

Google Cloud DK5197-SYST8151-PORTFOLIO02 vpn Search

Cloud Router

Get started with real-time analytics

Use Network Intelligence Center for comprehensive monitoring and troubleshooting. [Learn more](#)

- ✓ Visualize your network resources
- ✓ Diagnose and prevent connectivity issues
- ✓ View packet loss and latency metrics
- ✓ Keep your firewall rules strict and efficient

Routers CREATE ROUTER REFRESH

Filter

<input type="checkbox"/>	Name ↑	Network	Region	Interconnect encryption	Google ASN	Interconnect / VPN gateway	Connection	BGP sessions	Logs
<input type="checkbox"/>	gcp-router	dk5197-vpc2	us-east1	Unencrypted	65000	None			View

Successfully created router "gcp-router".

Figure 39 - Successfully created a new router

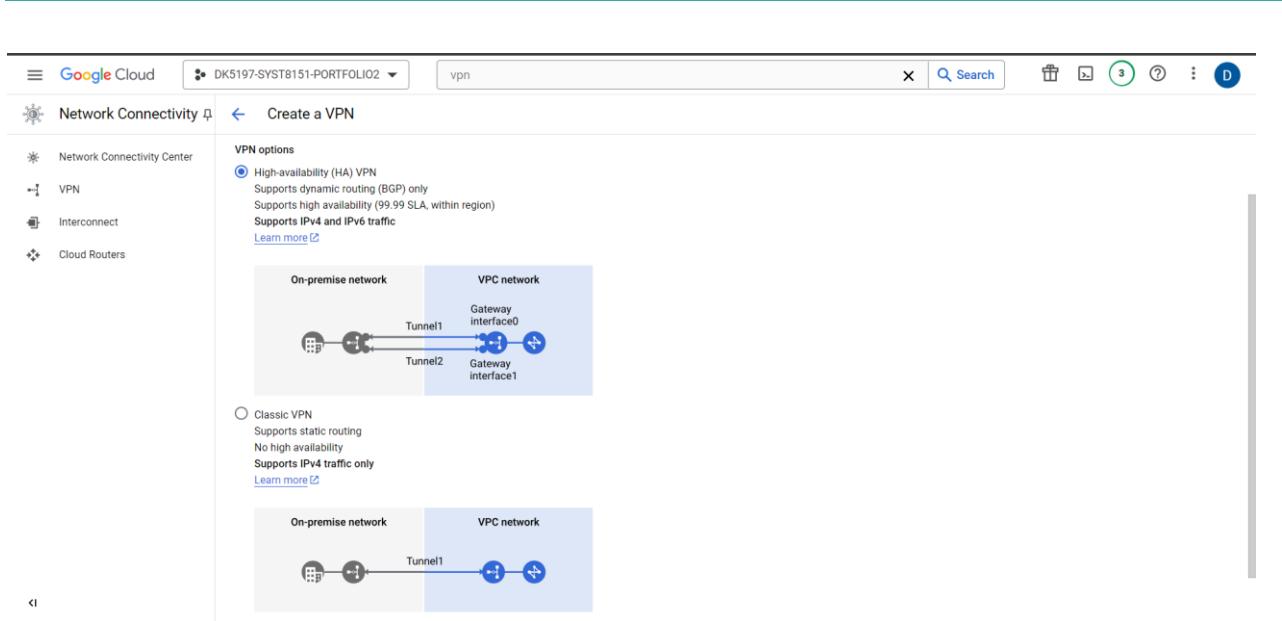


Figure 40 - Selecting HA VPN for the tunnelling in VPN

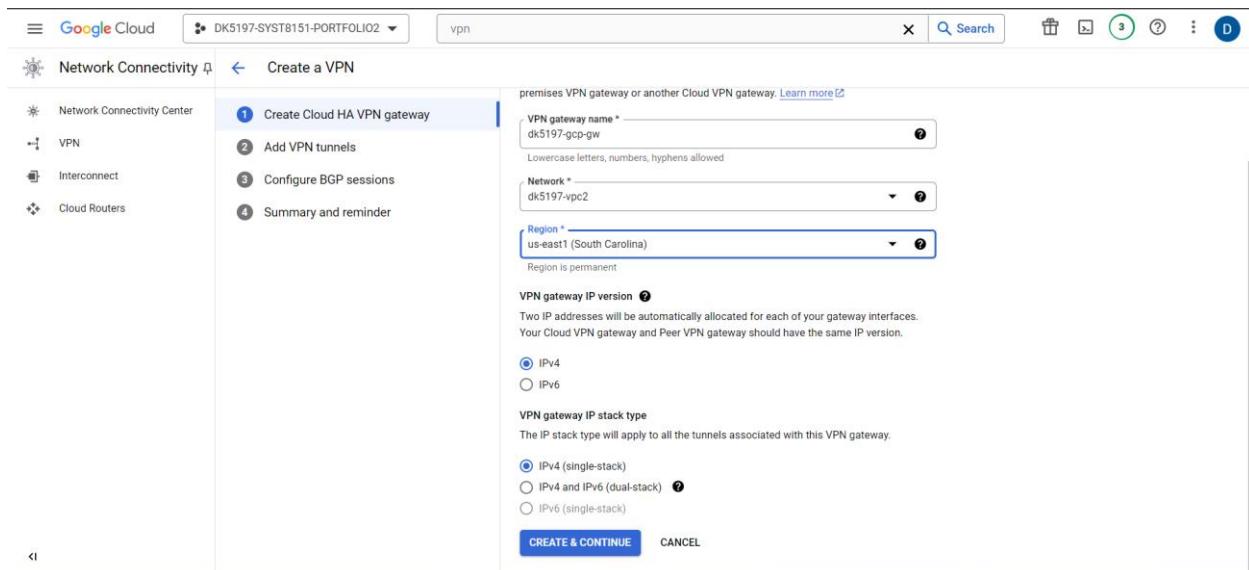


Figure 41 - Creating a new HA VPN gateway

In AWS, we created a new Virtual Private Gateway and named it aws-vpn, then attached it to our VPC, dk5197-VPC1. After setting up the Virtual Private Gateway, we needed to create Customer Gateways to represent the external network we wanted to connect to via the VPN. We named the first Customer Gateway dk5197-gcp-gw1, used 65000 as the ASN (matching Google Cloud's ASN), and set the IP address to the first interface IP of the Google Cloud router we had previously noted. Similarly, we created a second Customer Gateway named dk5197-gcp-gw2, using the second interface IP address.

With these configurations in place, we proceeded to create a new VPN connection in AWS. We named it aws-gcp-vpn, selected the transit gateway type as a Virtual Private Gateway, and linked it to the Customer Gateways. For both tunnel 1 and tunnel 2, we used the pre-shared key “hellocloud” for authentication between the Virtual Private Gateway and the Customer Gateways. The VPN connection was successfully created on the AWS side. Finally, we downloaded the VPN connection configuration, which included the Virtual Private Gateway addresses for each tunnel.

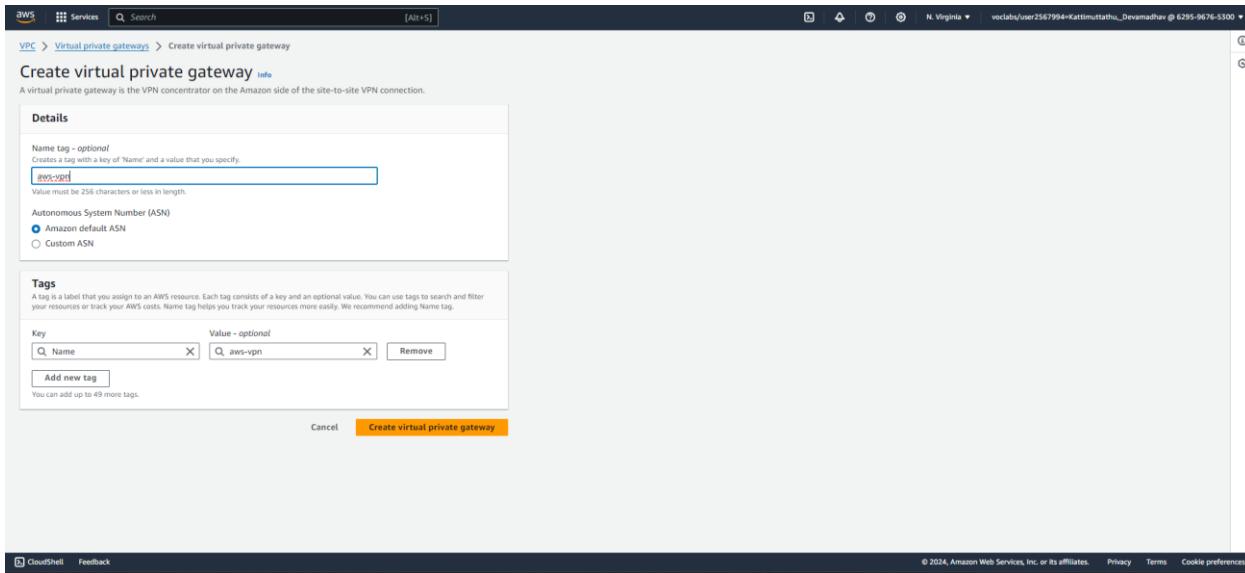


Figure 42 - Creating the virtual private gateway in AWS

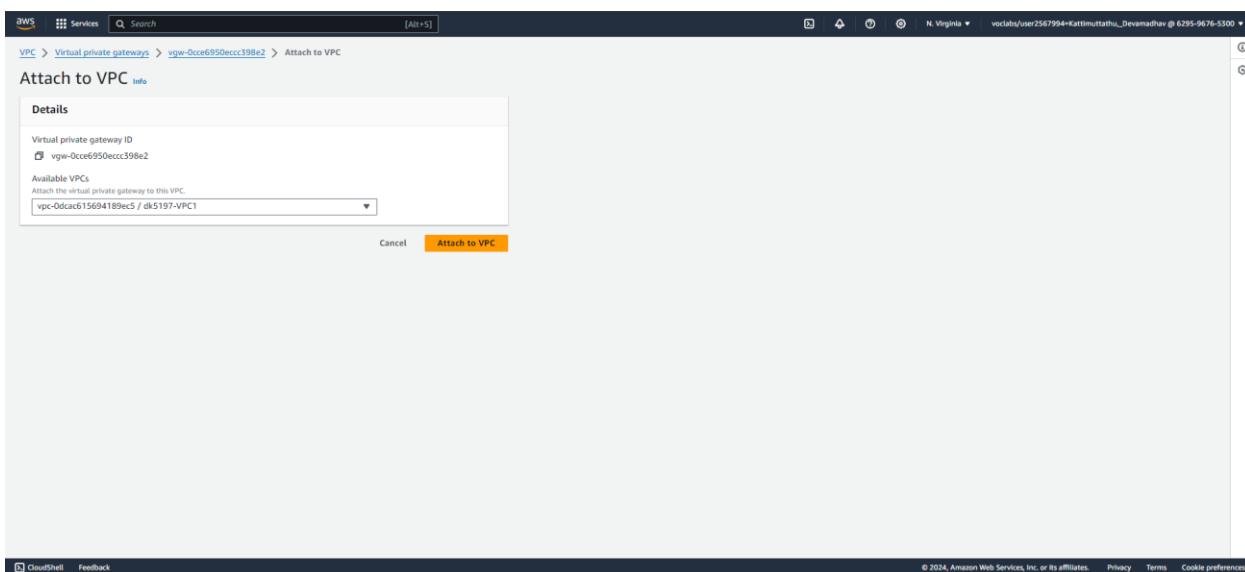


Figure 43 - Attaching the virtual private gateway to vpc

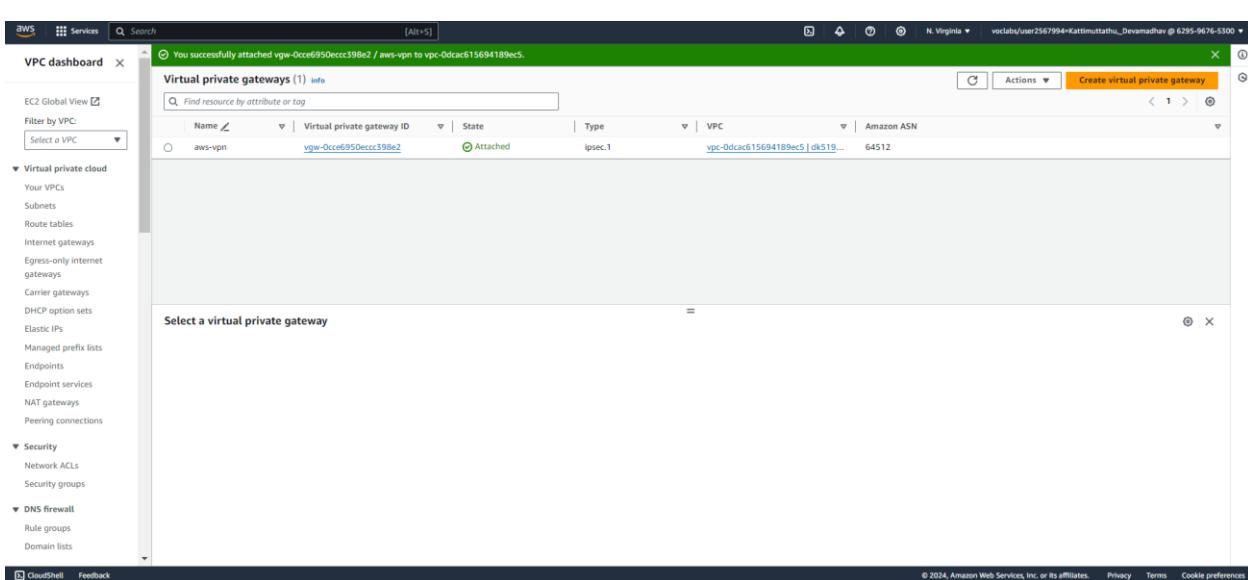


Figure 44 - Successfully attached the virtual private gateway to the VPC

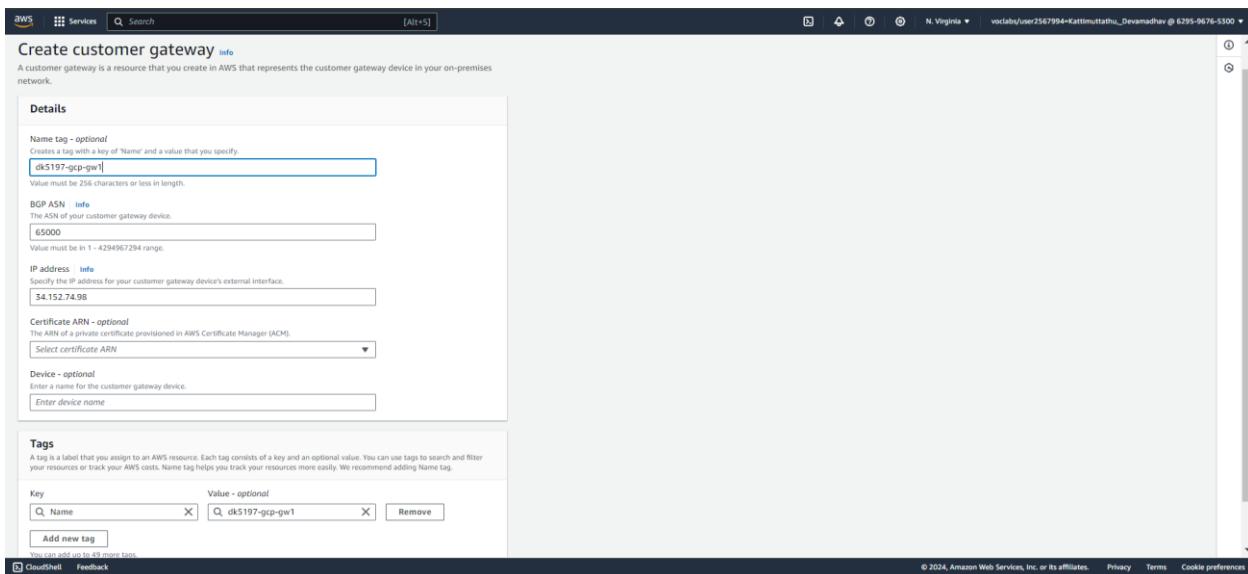


Figure 45 - Creating a new customer gateway with the interface0 of gcp as IP

Create customer gateway info

A customer gateway is a resource that you create in AWS that represents the customer gateway device in your on-premises network.

Details

Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.

Value must be 256 characters or less in length.

BGP ASN info
The ASN of your customer gateway device.

Value must be in 1 - 4294967294 range.

IP address info
Specify the IP address for your customer gateway device's external interface.

Certificate ARN - optional
The ARN of a private certificate provisioned in AWS Certificate Manager (ACM).

Device optional
Enter a name for the customer gateway device.

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs. Name tag helps you track your resources more easily. We recommend adding Name tag.

Key	Value - optional
<input type="text" value="Q_Name"/>	<input type="text" value="dk5197-gcp-gw2"/>
<input type="button" value="Remove"/>	

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 46 - Creating the second customer gateway with the interface1 of gcp as IP

Customer gateways (1/2) info

Name	Customer gateway ID	State	BGP ASN	IP address	Type	Certificate ARN
<input checked="" type="radio"/> dk5197-gcp-gw2	cgw-0d5615f3552108fda	Available	65000	35.220.15.77	ipsec.1	-
<input type="radio"/> dk5197-gcp-gw1	cgw-05776c899f94e51b4	Available	65000	34.152.74.98	ipsec.1	-

Customer gateway cgw-0d5615f3552108fda / dk5197-gcp-gw2

Details

Customer gateway ID <input type="text" value="cgw-0d5615f3552108fda"/>	State Available	Type <input type="text" value="ipsec.1"/>	IP address <input type="text" value="35.220.15.77"/>
BGP ASN <input type="text" value="65000"/>	Certificate ARN -	Device -	

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 47 - Successfully created two customer gateways

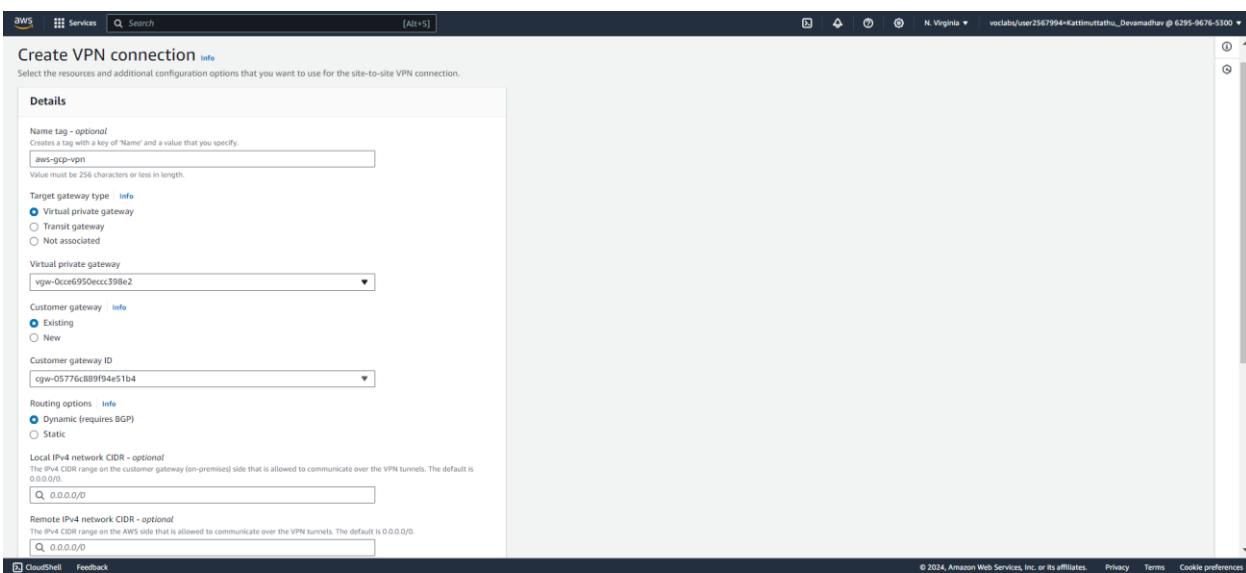


Figure 48 - Creating a new vpn connection in aws

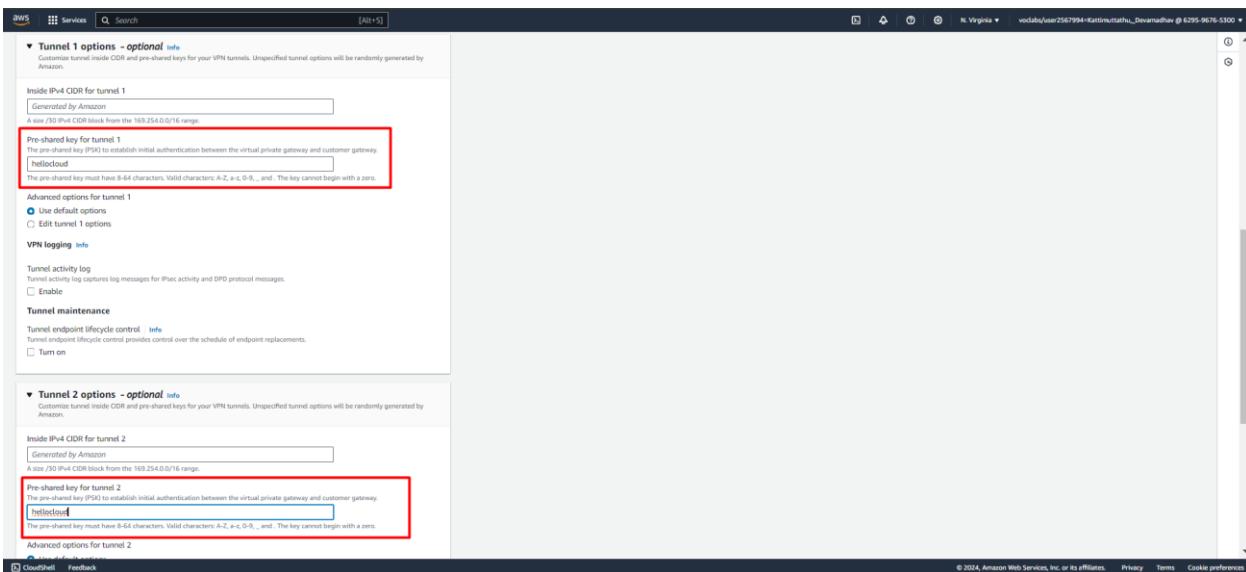


Figure 49 - Configuring tunneling in VPN connection and set hellocloud as preshared key for authentication

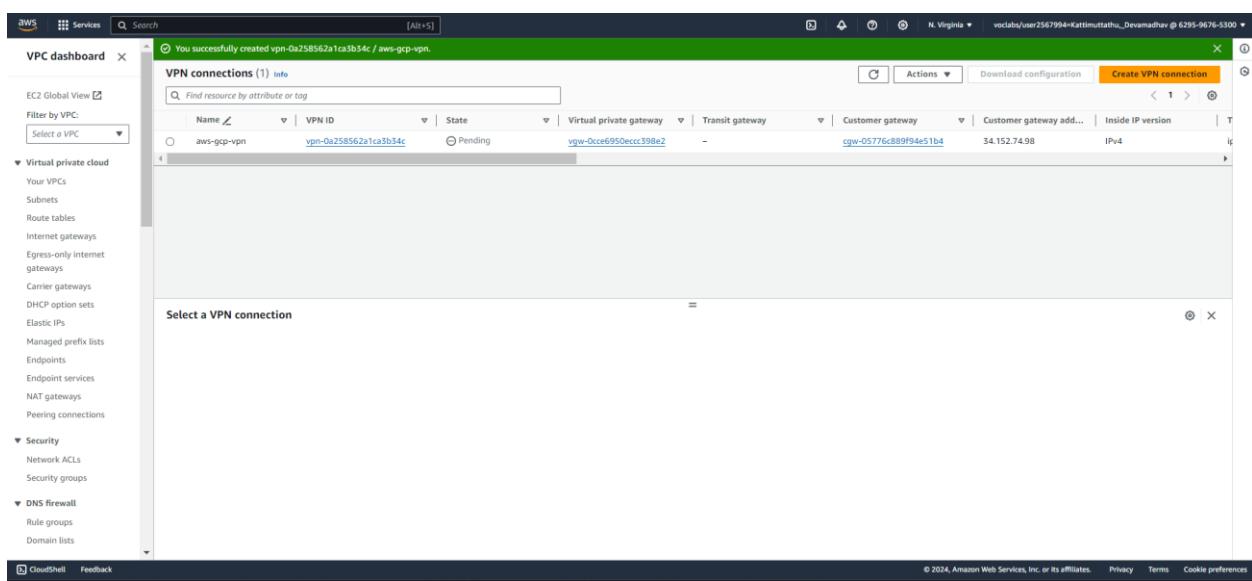


Figure 50 - Successfully created a new vpn connection

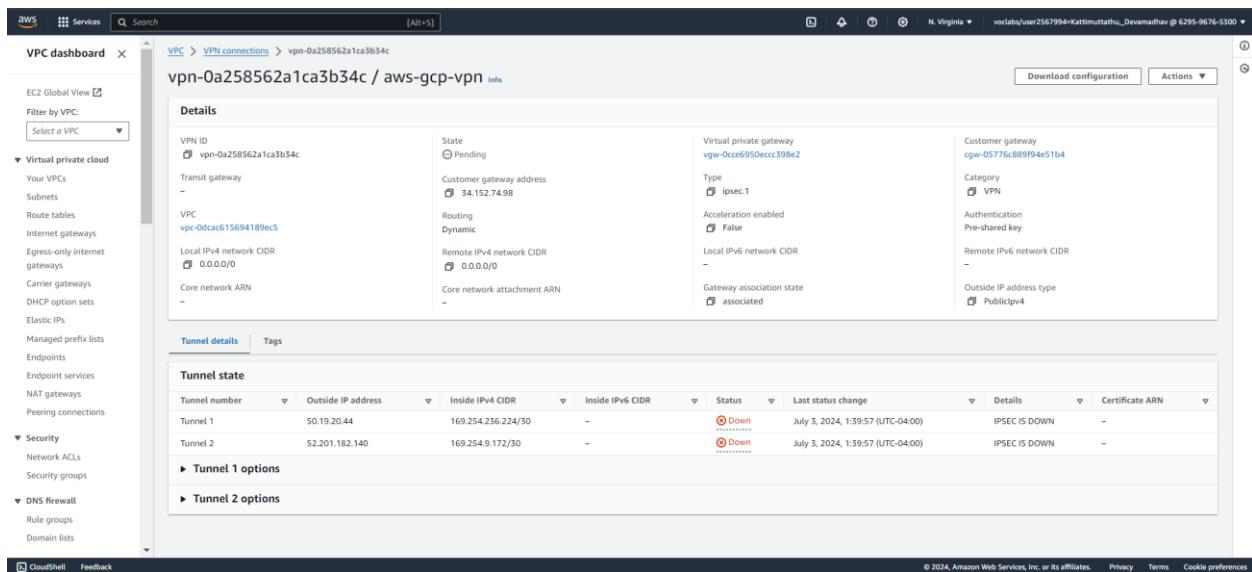


Figure 51 - Tunnels are in down state

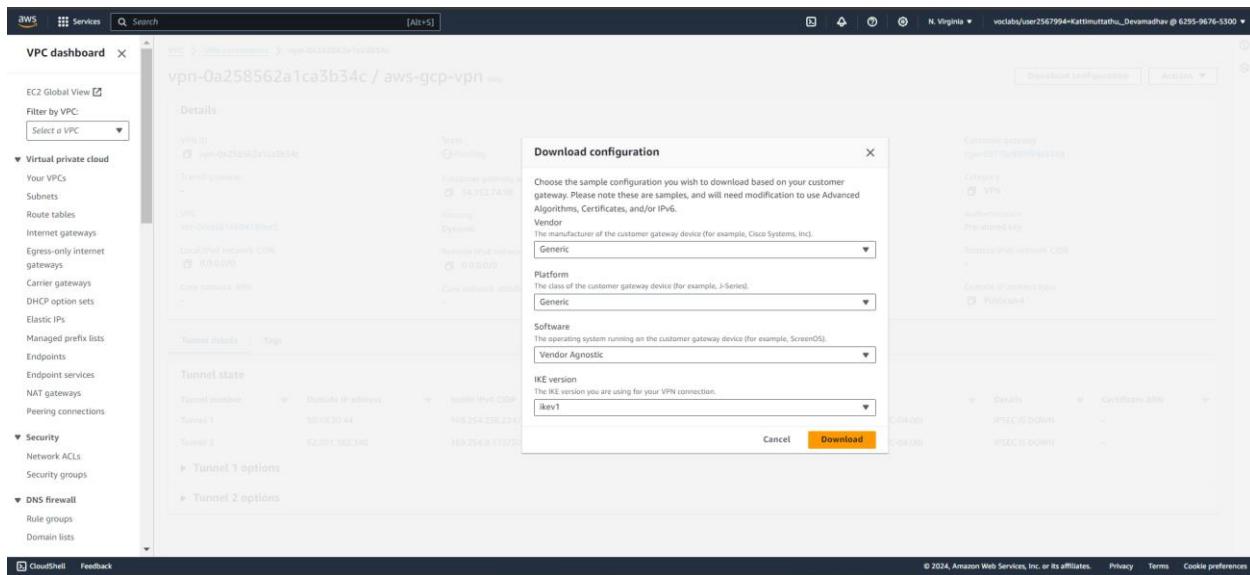


Figure 52 - Downloaded the vpn configuration file

```

vpn-0a258562a1ca3b34c.txt

File Edit View
traffic is exchanged. Each also contain an inside address associated with
the tunnel interface.

The Customer Gateway outside IP address was provided when the Customer Gateway
was created. Changing the IP address requires the creation of a new
Customer Gateway.

The Customer Gateway inside IP address should be configured on your tunnel
interface.

Outside IP Addresses:
- Customer Gateway : 34.152.74.98
- Virtual Private Gateway : 50.19.20.44

Inside IP Addresses
- Customer Gateway : 169.254.236.226/30
- Virtual Private Gateway : 169.254.236.225/30

Configure your tunnel to fragment at the optimal size:
- Tunnel interface MTU : 1436 bytes

#4: Border Gateway Protocol (BGP) Configuration:

The Border Gateway Protocol (BGPv4) is used within the tunnel, between the inside
IP addresses, to exchange routes from the VPC to your home network. Each
BGP router has an Autonomous System Number (ASN). Your ASN was provided
to AWS when the Customer Gateway was created.

BGP Configuration Options:
Ln 92, Col 22 | 10,999 characters

```

Figure 53 - Virtual private gateway details for tunnell1'

```
vpn-0a258562a1ca3b34c.txt
File Edit View
The Customer Gateway inside IP address should be configured on your tunnel interface.

Outside IP Addresses:
- Customer Gateway : 34.152.74.98
- Virtual Private Gateway : 52.201.182.140

Inside IP Addresses
- Customer Gateway : 169.254.9.174/30
- Virtual Private Gateway : 169.254.9.173/30

Configure your tunnel to fragment at the optimal size:
- Tunnel interface MTU : 1436 bytes

#4: Border Gateway Protocol (BGP) Configuration:
The Border Gateway Protocol (BGPv4) is used within the tunnel, between the inside IP addresses, to exchange routes from the VPC to your home network. Each BGP router has an Autonomous System Number (ASN). Your ASN was provided to AWS when the Customer Gateway was created.

BGP Configuration Options:
- Customer Gateway ASN : 65000
- Virtual Private Gateway ASN : 64512
- Neighbor IP Address : 169.254.9.173
- Neighbor Hold Time : 30

Configure BGP to announce routes to the Virtual Private Gateway. The gateway
Ln 92, Col 22 10,999 characters
100% Unix (LF)
```

Figure 54 - Virtual private gateway detail for tunnel2

After configuring the VPN tunnel in AWS, we returned to the Google Cloud console to set up the corresponding VPN gateway. We created a new VPN gateway in GCP using the interface address of the AWS Virtual Private Gateway we had just established. We selected our newly created router for the VPN and configured the VPN gateway interfaces, naming them dk5197-gcp-tunnel and dk5197-gcp-tunnel2. We used the same pre-shared key, “hellocloud,” for authentication between the AWS and GCP gateways. The VPN gateway was successfully created.

Next, we configured the BGP session for tunnel 1, naming it dk5197-bgp, with AWS’s ASN set to 64512. We used the inside IP address from the AWS VPN tunnel configuration. After setting up the BGP session, the tunnel was successfully established. We confirmed this by checking the AWS console, where the tunnel status had changed to "UP."

Additionally, we updated the AWS route table to include a new route directing traffic to the Google Cloud network (10.173.46.0/24) through the Virtual Private Gateway. Afterward, we successfully pinged the AWS instance from the Google Cloud instance, confirming the connection. We also tested the connection by pinging google.com from the instance.

With these steps, we successfully established communication between the two cloud networks across different platforms.

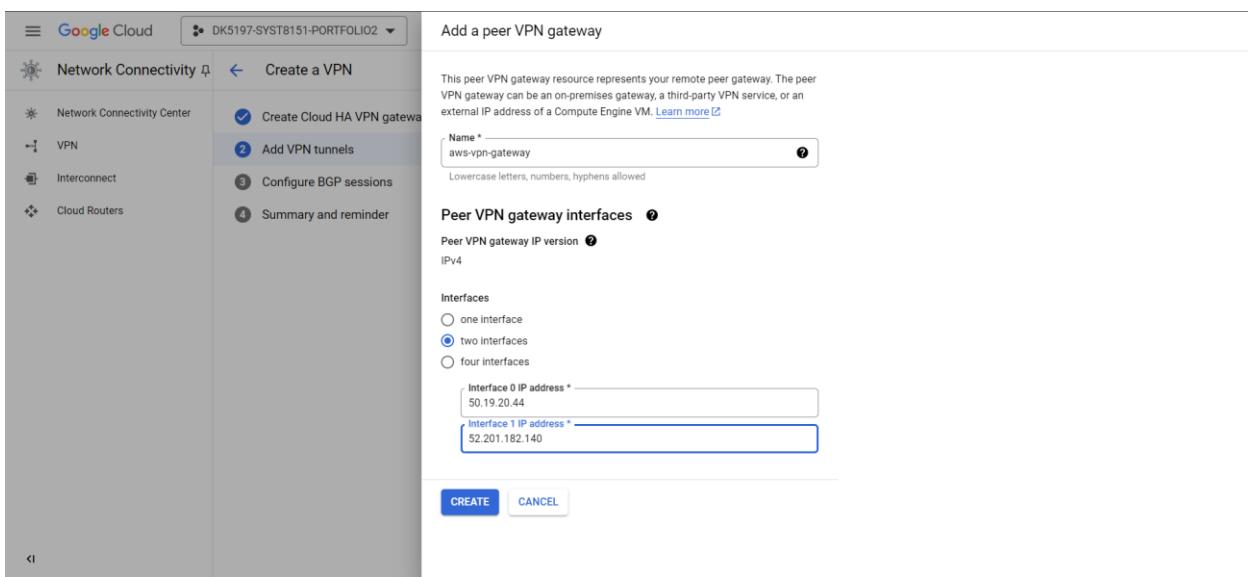


Figure 55 - Configuring the peer VPN gateway in gcp using the virtual private gateway from aws

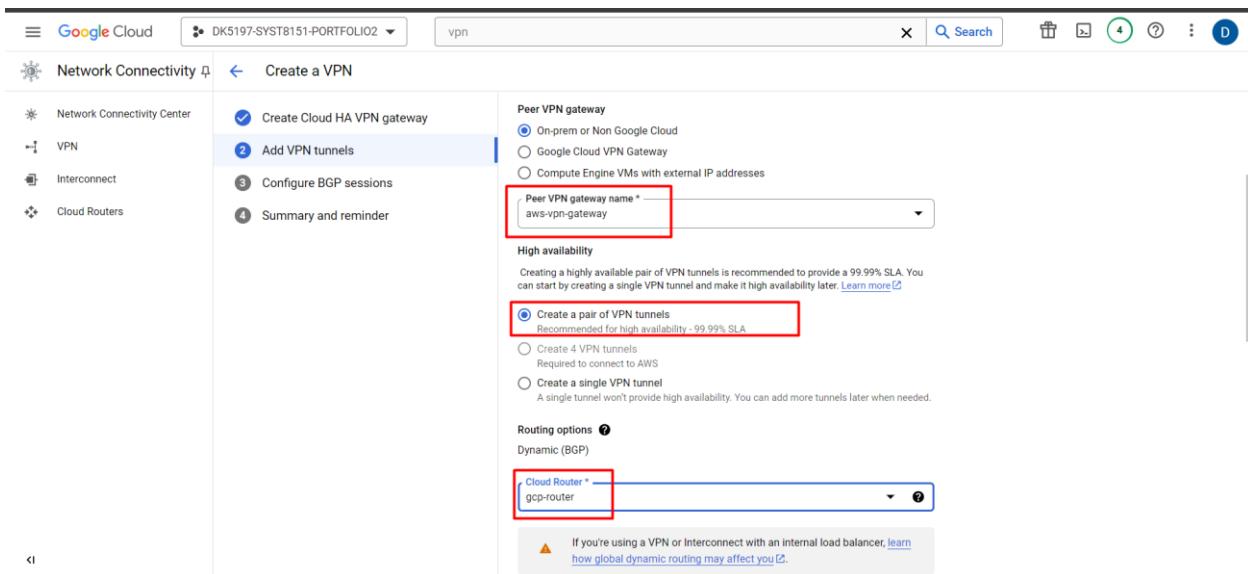


Figure 56 - Creating a vpn tunne in gcp

Associated Cloud VPN gateway interface
0:34.152.74.98

Associated peer VPN gateway interface *
0:50.19.20.44

Name *
dk5197-gcp-tunnel

Description

IKE version
IKEv2

IKE pre-shared key *
hellocloud GENERATE AND COPY

⚠ Make sure you record the pre-shared key in a secure location. The key can't be retrieved after this form is closed. [Learn more](#)

DONE

VPN tunnel 2 (not yet configured)

Figure 57 - Configuring tunnel1 details for vpn

Associated Cloud VPN gateway interface
1:35.220.15.77

Associated peer VPN gateway interface *
1:52.201.182.140

Name *
dk5197-gcp-tunnel2

Description

IKE version
IKEv2

IKE pre-shared key *
hellocloud GENERATE AND COPY

⚠ Make sure you record the pre-shared key in a secure location. The key can't be retrieved after this form is closed. [Learn more](#)

DONE

Figure 58 - Configuring tunnel2 details in vpn

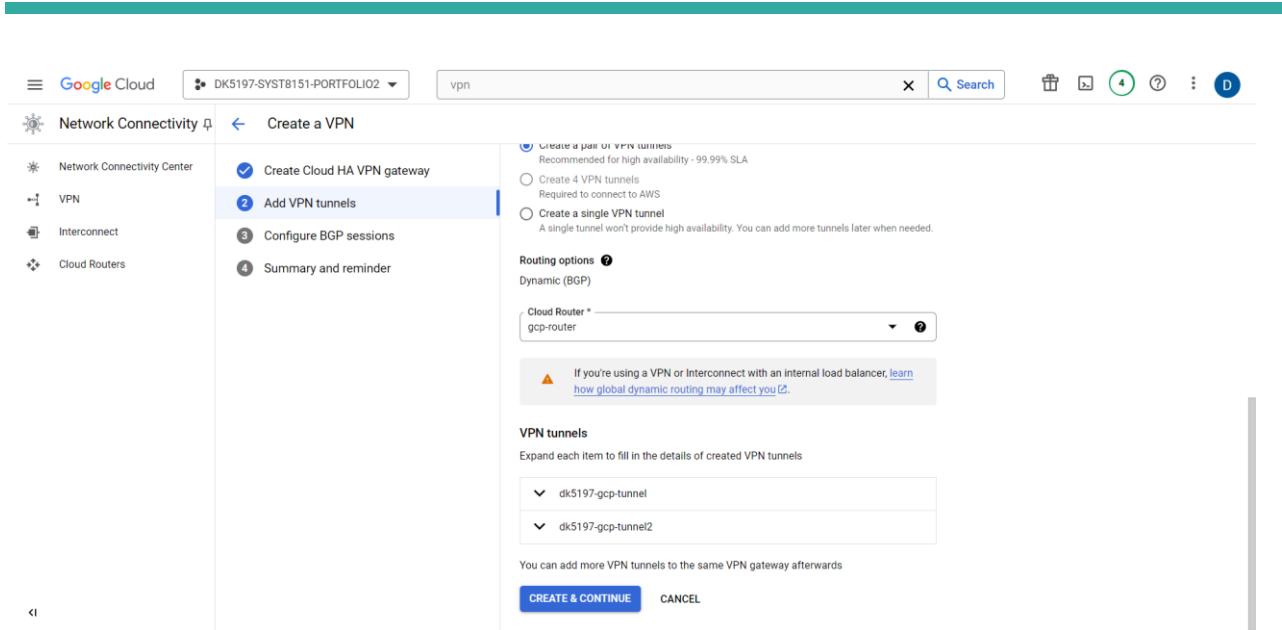


Figure 59 - Successfully configured all details for vpn tunnel

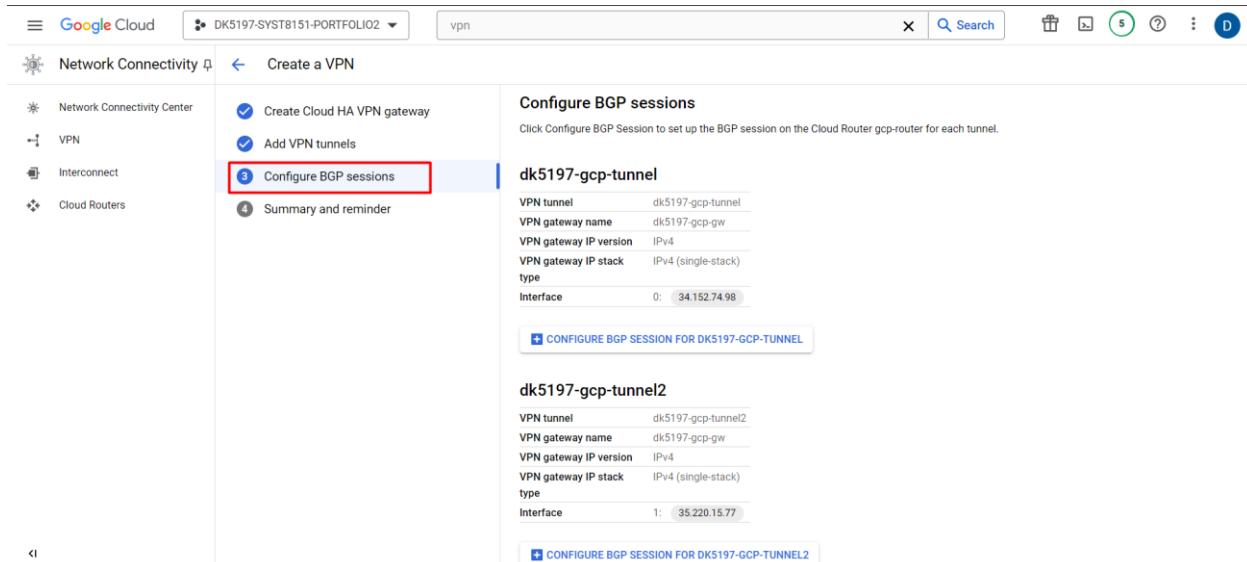


Figure 60 - Configuring the BGP sessions for the VPN

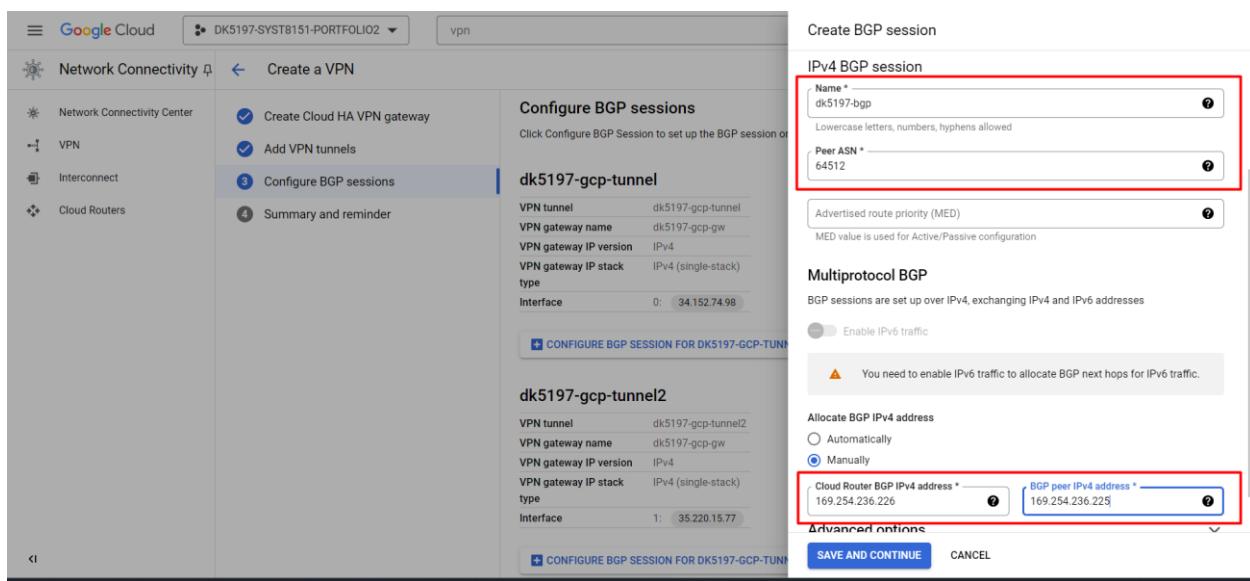


Figure 61 - Creating a new BGP connection for tunnel1

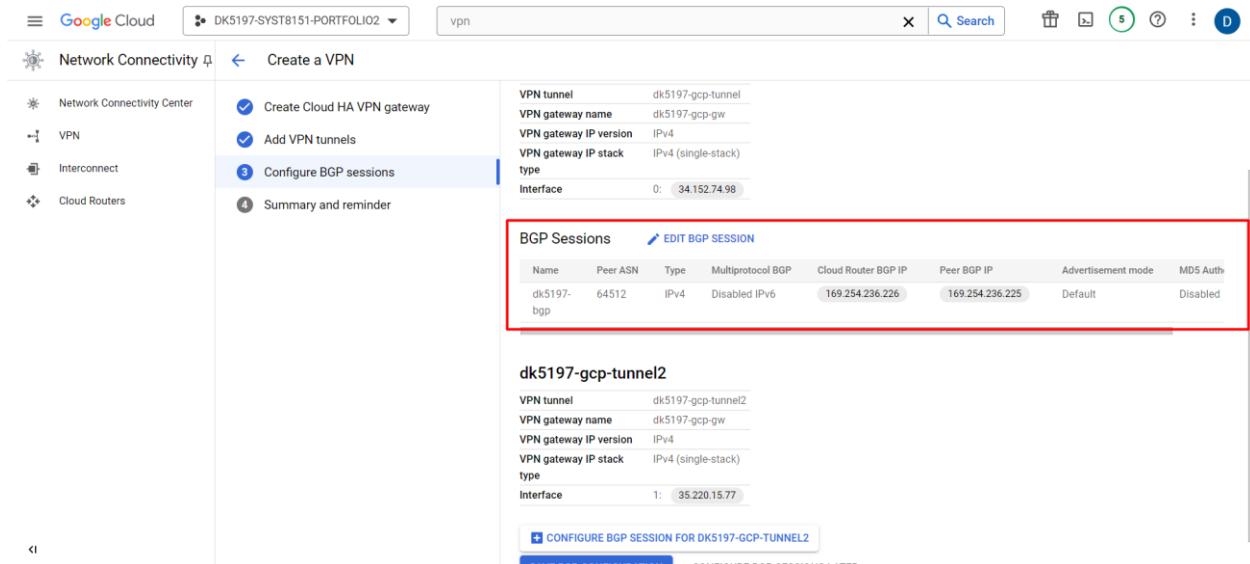


Figure 62 - Successfully created a new bgp session

The screenshot shows the Google Cloud Network Connectivity VPN interface. On the left sidebar, 'VPN' is selected. The main area displays a 'Get started with real-time analytics' section and a table of 'CLOUD VPN TUNNELS'. The table has columns: Name, Cloud VPN gateway (IP), Peer VPN gateway (IP), Cloud Router BGP, and Actions. Two rows are listed:

Name	Cloud VPN gateway (IP)	Peer VPN gateway (IP)	Cloud Router BGP	Actions
dk5197-gcp-gw	34.152.74.98	aws-vpn-gateway	50.19.20.44	⋮
dk5197-gcp-gw	35.220.15.77	aws-vpn-gateway	52.201.182.140	CONFIGURE BGP SESSION ⋮

A modal window titled 'Select a VPN tunnel' is open on the right, showing a message: 'No VPN tunnels selected.'

Figure 63 - Successfully created two vpn tunnels

The screenshot shows the same Google Cloud Network Connectivity VPN interface as Figure 63. The 'CLOUD VPN TUNNELS' table now includes a column 'VPN tunnel status' which is highlighted with a red box. The first row's status is 'Established'.

P IP address	Peer BGP IP address	VPN tunnel status	BGP session status	VPC n	Actions
26	169.254.236.225	Established	BGP established	dk519	⋮
—	—	No incoming packets	BGP not configured	dk519	CONFIGURE BGP SESSION ⋮

The 'Select a VPN tunnel' modal window is still present on the right.

Figure 64 - Tunnel1 status is in established state

VPC dashboard

VPN connections > **vpn-0a258562a1ca3b34c** / **aws-gcp-vpn**

Tunnel number	Outside IP address	Inside IPv4 CIDR	Inside IPv6 CIDR	Status	Last status change	Details	Certificate ARN
Tunnel 1	50.19.20.44	169.254.236.224/30	-	Up	July 3, 2024, 2:01:55 (UTC-04:00)	2 BGP ROUTES	-
Tunnel 2	52.201.182.140	169.254.9.172/30	-	Down	July 3, 2024, 1:42:29 (UTC-04:00)	IPSEC IS DOWN	-

Tunnel 1 options

Tunnel 2 options

Figure 65 - Tunnel status in aws has changed to UP .Thereby enabling communication

Edit routes

Destination	Target	Status	Propagated
10.30.0.0/20	local	Active	No
0.0.0.0/0	Internet Gateway	Active	No
0.0.0.0/0	vgw-0cce6950ecc398e2	-	No
10.173.46.0/24	Virtual Private Gateway	-	No

Add route

Save changes

Figure 66 - Creating a new route to the gcp through virtual private gateway

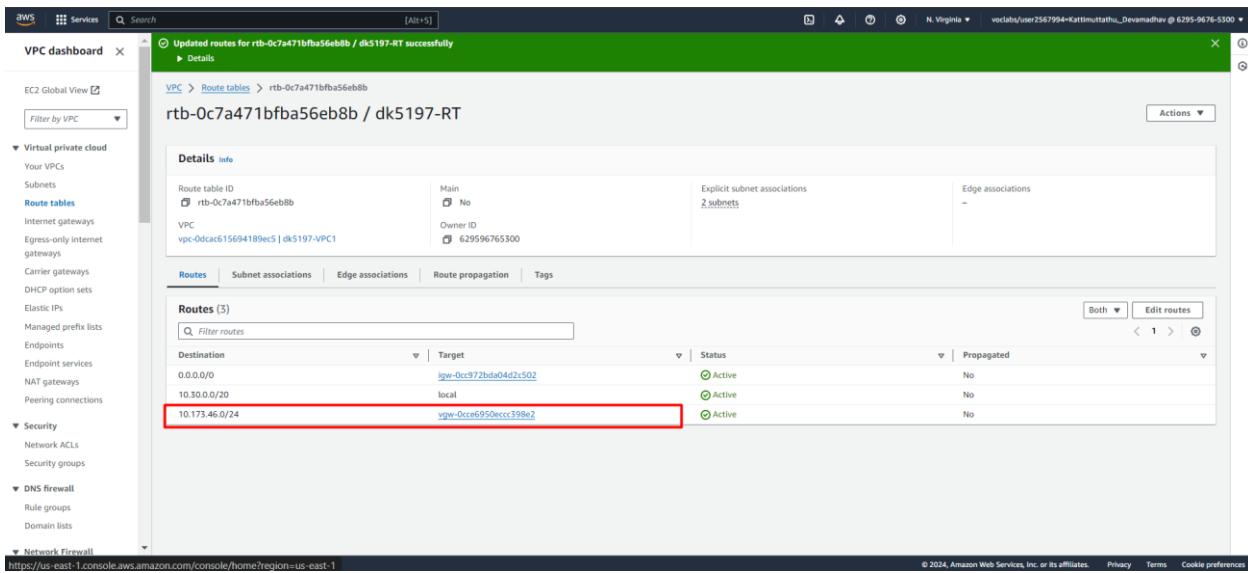


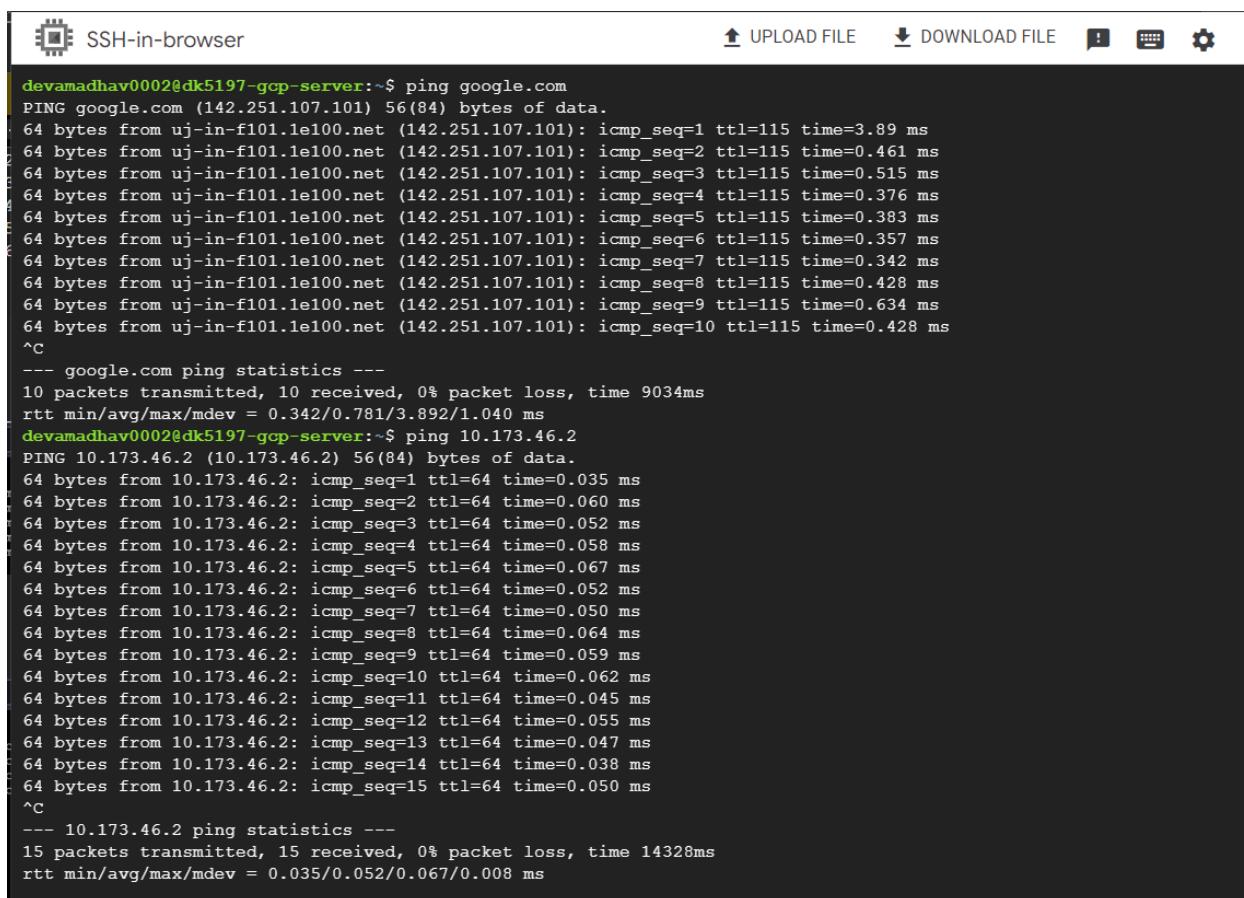
Figure 67 - Successfully created a new route

```

devamadhav0002@dk5197-gcp-server:~$ ping 10.30.0.214
PING 10.30.0.214 (10.30.0.214) 56(84) bytes of data.
64 bytes from 10.30.0.214: icmp_seq=4385 ttl=126 time=14.7 ms
64 bytes from 10.30.0.214: icmp_seq=4386 ttl=126 time=14.2 ms
64 bytes from 10.30.0.214: icmp_seq=4387 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4388 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4389 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4390 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4391 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4392 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4393 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4394 ttl=126 time=14.2 ms
64 bytes from 10.30.0.214: icmp_seq=4395 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4396 ttl=126 time=14.5 ms
64 bytes from 10.30.0.214: icmp_seq=4397 ttl=126 time=14.2 ms
64 bytes from 10.30.0.214: icmp_seq=4398 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4399 ttl=126 time=14.6 ms
64 bytes from 10.30.0.214: icmp_seq=4400 ttl=126 time=14.3 ms
64 bytes from 10.30.0.214: icmp_seq=4401 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4402 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4403 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4404 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4405 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4406 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4407 ttl=126 time=14.1 ms
64 bytes from 10.30.0.214: icmp_seq=4408 ttl=126 time=14.4 ms
64 bytes from 10.30.0.214: icmp_seq=4409 ttl=126 time=14.6 ms
64 bytes from 10.30.0.214: icmp_seq=4410 ttl=126 time=14.4 ms

```

Figure 68 - Successful communication with aws instance from gcp instance



The screenshot shows a terminal window titled "SSH-in-browser". At the top, there are buttons for "UPLOAD FILE" and "DOWNLOAD FILE", along with icons for file operations. The terminal output is as follows:

```
devamadhav0002@dk5197-gcp-server:~$ ping google.com
PING google.com (142.251.107.101) 56(84) bytes of data.
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=1 ttl=115 time=3.89 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=2 ttl=115 time=0.461 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=3 ttl=115 time=0.515 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=4 ttl=115 time=0.376 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=5 ttl=115 time=0.383 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=6 ttl=115 time=0.357 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=7 ttl=115 time=0.342 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=8 ttl=115 time=0.428 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=9 ttl=115 time=0.634 ms
64 bytes from uj-in-f101.1e100.net (142.251.107.101): icmp_seq=10 ttl=115 time=0.428 ms
^C
--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9034ms
rtt min/avg/max/mdev = 0.342/0.781/3.892/1.040 ms
devamadhav0002@dk5197-gcp-server:~$ ping 10.173.46.2
PING 10.173.46.2 (10.173.46.2) 56(84) bytes of data.
64 bytes from 10.173.46.2: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 10.173.46.2: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from 10.173.46.2: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 10.173.46.2: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 10.173.46.2: icmp_seq=5 ttl=64 time=0.067 ms
64 bytes from 10.173.46.2: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 10.173.46.2: icmp_seq=7 ttl=64 time=0.050 ms
64 bytes from 10.173.46.2: icmp_seq=8 ttl=64 time=0.064 ms
64 bytes from 10.173.46.2: icmp_seq=9 ttl=64 time=0.059 ms
64 bytes from 10.173.46.2: icmp_seq=10 ttl=64 time=0.062 ms
64 bytes from 10.173.46.2: icmp_seq=11 ttl=64 time=0.045 ms
64 bytes from 10.173.46.2: icmp_seq=12 ttl=64 time=0.055 ms
64 bytes from 10.173.46.2: icmp_seq=13 ttl=64 time=0.047 ms
64 bytes from 10.173.46.2: icmp_seq=14 ttl=64 time=0.038 ms
64 bytes from 10.173.46.2: icmp_seq=15 ttl=64 time=0.050 ms
^C
--- 10.173.46.2 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14328ms
rtt min/avg/max/mdev = 0.035/0.052/0.067/0.008 ms
```

Figure 69 - successfully verified the connection