

The Network Times
Handbook Series – Part VII

AWS Networking Fundamentals

A Practical Guide to Understand How to Build a Virtual Datacenter into the AWS Cloud



AWS consists of interconnected **Regions**, each having multiple **Availability Zones (AZ)**. AZ is an isolated area that consists of one or more physical Datacenters. **Virtual Private Cloud (VPC)** within a Region is your Virtual Datacenter. AZ-specific subnets in your VPC are either **Public** with **Internet GW** or **Private** with or without **NAT GW**. You can build a VPN between your on-prem DC to AWS VPC using a **Virtual Private GW**, or **Transit GW**. Using **AWS Direct Connect**, you can establish a dedicated connection from on-prem to VPC. You can Inter-connect VPCs with **VPC Peering**, **Private Link**, or **Transit GW**. **NACL** and **Network FW** protect your Subnets while **Security Groups** guard your **EC2 instances**. This book is your guide to AWS Networking.

Toni Pasanen, CCIE#28158

AWS Networking Fundamentals

A Practical Guide to Understand How to Build a Virtual Datacenter into the AWS Cloud

Toni Pasanen, CCIE 28158

Copyright © Toni Pasanen, all rights reserved.

Published - 1 January 2022:

About the Author:

Toni Pasanen. CCIE No. 28158 (RS), Distinguished Engineer at Fujitsu Finland. Toni started his IT carrier in 1998 at Tieto, where he worked as a Service Desk Specialist moving via the LAN team to the Data Center team as a 3rd. Level Network Specialist. Toni joined Teleware (Cisco Learning partner) in 2004, where he spent two years teaching network technologies focusing on routing/switching and MPLS technologies. Toni joined Tieto again in 2006, where he spent the next six years as a Network Architect before joining Fujitsu. In his current role, Toni works closely with customers helping them in selecting the right network solutions not only from the technology perspective but also from the business perspective.

He is also the author of books:

Virtual Extensible LAN – VXLAN:

The Practical Guide to VXLAN Solution
(August 2019)

LISP Control-Plane in Campus Fabric:

A Practical Guide to Understand the Operation of Campus Fabric
(February 2020)

VXLAN Fabric with BGP EVPN Control-Plane:

Design Considerations
(September 2020)

Object-Based Approach to Cisco ACI:

The Logic Behind the Application Centric Infrastructure
(December 2020)

Cisco SD-WAN:

A Practical Guide to Understand the Basics of Cisco Viptela Based SD-WAN Solution (May 2021)

Network Virtualization:

LISP, OMP, and BGP EVPN Operation and Interaction
(August 2021)

About This Book

Moving from traditional Datacenter networking to AWS VPC Networking might feel like a giant leap. For me, however, Cloud networking is a perfect example of an Intent-Based System that makes the life of a network engineer easier. For instance, let us say that the application team needs a closed tenant on AWS Stockholm Region for their development servers. To do that, I tell the AWS management system that I want a new VPC-X with CIDR 10.10.0.0/16 in London Region. When the VPC is up and running, I ask the system to launch a subnet 10.10.10.0/24 in one of VPC-X's Availability Zone, and that's it. Now the development team has a closed tenant. I didn't have to do anything else than describe what I wanted, and the system took care of the implementation.

Chapter 1 - Virtual Private Cloud – VPC: The first chapter explains how you can start your virtual Datacenter, Virtual Private Cloud (VPC). You will learn what Regions and Availability Zones (AZ) are and how you create subnets within AZ. Subnets in VPC are not broadcast domains. They are networks for workloads having equal connectivity requirements. This chapter explains how you create subnet-specific route tables and Network Access Control Lists (NACL).

Chapter 2 - VPC Control-Plane and Data-Plane: This chapter introduces the Mapping-Service, the Control-Plane of AWS. It also discusses Data-Plane operation and VPC encapsulation.

Chapter 3 - VPC Internet Gateway: AWS classifies subnets as either public or private. A public subnet is a subnet with an Internet Gateway (IGW) attached to it. Using IGW, we make instances visible from the Internet. This chapter explains how to create an Internet Gateway and how to route traffic to it. In this chapter, we launch our first EC2 instance. Besides, we modify instance-specific Security Group (SG) to allow ssh from the test PC on the Internet side.

Chapter 4 - VPC Nat Gateway: This chapter introduces a NAT Gateway (NAT GW), which offers egress-only Internet access for EC2 Instances in the Private subnet. EC2 instances in Private subnets have the Internet connection via NAT GW. However, hosts on the Internet can not initiate connections to EC2 instances behind a NAT GW.

Chapter 5 - Virtual Private Gateway: Chapter five introduces how we enable a Hybrid Datacenter architecture by establishing a VPN connection over the Internet between VGW (Virtual Private Gateway on VPC) and CGW (Customer Gateway on on-prem site).

Chapter 6 - Transit Gateway: The following three chapter introduces AWS Transit Gateway. This chapter explains how we can provide an Inter-VPC connection within AWS Region by using Transit Gateway. It also explains how we can set up a VPN connection between VPC and on-prem Datacenter using TGW.

Chapter 7 - VPC Segmentation with Transit Gateway: This chapter explains how you can implement Inter-regional VPC segmentation policy by using VPC-specific TGW route tables.

Chapter 8 - Transit Gateway Peering: We can build an Inter-Region backbone for or VPCs using Transit Gateway Peering connections. This chapter explains how you create the TGW peering connection between two TGWs. It also explains what are the necessary routing requirements to enable traffic flows over TGW peering connection.

Chapter 9 – VPC Peering: This chapter introduces VPC peering model that is a simple Inter-VPC connection between two VPCs. It requires subnet route table modification to allow traffic flows between VPCs.

Chapter 10 – AWS PrivateLink: This chapter explains how we build application-specific, unidirectional Inter-VPC connections. PrivateLink solution use three main components, Network Load Balancer, Endpoint Service, and Endpoints. Service Providers can publish their services to one or many VPCs. That said, a PrivateLink makes it possible to publish shared services to many customers.

Chapter 11 – Dedicated Direct Connect: This is the first of the four AWS Direct Connect connection chapters. It explains how to order a cross-connect between AWS devices and the customer device located in AWS Direct Connect Location. This chapter also introduces how to create AWS Direct Connect Gateway (DXGW) and what is needed to create BGP peering between DXGW and customer devices. You will also learn how to attach Transit Gateway to DXGW.

Chapter 12 – Hosted Direct Connect: This chapter introduces AWS Hosted Direct Connect. It explains how AWS Direct Connect Partners offers a cross-connection from the AWS Direct Connect Partner using their infrastructure. Because some AWS DCPs used BGP EVPN/VXLAN Fabric for their Hosted Direct Connect service, this chapter also introduces the BGP EVPN based MAC address learning process and explains how the VXLAN Data-Plane traffic forwarding works.

Chapter 13 – Direct Connect BGP Policy: This chapter explains how we can affect to egress path selection process on AWS Direct Connect Gateway (DXGW). DXGW are AWS-managed services, so we can't change their configuration. However, we can use BGP built-in tools (BGP route aggregation, BGP AS-Path Prepending, and BGP Communities) for affecting to egress path selection process.

Chapter 14 – AWS Direct Connect SiteLink: This chapter introduces the AWS Direct Connect SiteLink, which enables site-to-site traffic over Direct Connect connections terminated into the same DXGW. It also explains how to migrate an existing Corporate inter-site WAN connection to AWS BackBone.

Chapter 15 – AWS Direct Connect – Public VIF: This chapter introduces how to use an AWS Direct Connect connection (DX) and a Public Virtual Interface (P-VIF) for accessing AWS Public Service.

Disclaimers

The content of this book is based only on the author's own experience and testing results. Its content is neither validated nor accepted by Amazon or any other organization or person. This book is meant to be neither design nor an implementation guide. After reading this book, readers should do their technology validation before using it in a production environment.

Table of Contents

About This Book v

Chapter 1: Virtual Private Cloud - VPC 1

VPC 1	
<i>VPC Introduction</i> 1	
<i>The Structure of Availability Zone</i> 2	
Create VPC - AWS Console 4	
<i>Select Region</i> 4	
<i>Create VPC</i> 7	
<i>DHCP Options Set</i> 9	
<i>Main Route Table</i> 10	
<i>VPC Verification Using AWS CLI</i> 12	
Create VPC - AWS CloudFormation 16	
<i>Create Template</i> 17	
<i>Upload Template</i> 17	
<i>Verification Using AWS Console</i> 18	
<i>VPC Verification using AWS CLI</i> 21	
Create Subnets - AWS Console 23	
<i>Create Subnets</i> 24	
Route Tables 29	
Create Subnets – AWS Console 30	
Create Subnets - AWS CloudFormation 37	
Create Network ACL 40	

Chapter 2: VPC Control-Plane 43

VPC Control-Plane – Mapping Service 43	
<i>Introduction</i> 43	
<i>Mapping Register</i> 43	
<i>Mapping Request - Reply</i> 44	
<i>Data-Plane Operation</i> 45	
References 46	

Chapter 3: VPC Internet Gateway 47

Introduction 47	
Allow Internet Access from Subnet 48	
<i>Create Internet Gateway</i> 49	
<i>Update Subnet Route Table</i> 54	
<i>Network Access Control List</i> 57	
Associate SG and Elastic-IP with EC2 59	
<i>Create Security Group</i> 59	
<i>Launch an EC2 Instance</i> 65	
<i>Allocate Elastic IP address from Amazon Ipv4 Pool</i> 71	
Reachability Analyzer 81	
Billing 85	

Chapter 4: VPC NAT Gateway 87

- Introduction 87
- Create NAT Gateway and Allocate Elastic IP 89
- Add Route to NGW on Private Subnet Route Table 94
- Test Connections 97
- Billing 101

Chapter 5: Virtual Private Gateway - VGW 103

- Introduction 103
- Customer Gateway (CGW) 105
 - Create CGW* 106
- Virtual Gateway (VGW) 109
 - Create CGW* 109
 - Attach CGW to VPC* 110
- Route Table Propagation 113
 - Edit Route Table Route Propagation* 113
- VPN Connection 115
 - Edit Route Table Route Propagation* 115
- CGW Configuration 119
 - Download CFG File* 119
 - Configure CGW Device* 126
- Tunnel Verification 128
- Control-Plane Verification 132
- Data-Plane Verification 134
- Billing 135

Chapter 6: Transit Gateway 137

- Introduction 137
- Create Transit Gateway 139
 - Launch TGW* 140
 - Create Transit Gateway Attachment* 144
 - Update Subnet Route Tables* 150
 - Data-Plane Testing* 152
- Create VPN Connection 153
 - Configure VPN on TGW* 154
 - Configure VPN on CGW* 159
 - Control-Plane and Data-Plane Verification* 160
 - Transit Gateway Pricing* 165

Chapter 7: VPC Segmentation with Transit Gateway 167

Introduction 167
Create Route Table for Attachments 173
 Create TGW Route Table 174
 Detach Attachments from the Default RT 176
 Associate Attachments with RT 178
Route Table Propagation 180
 Create Propagation 181
Summary 191

Chapter 8: Transit Gateway Peering 193

Introduction 193
Create TGW Peering 195
 TGW Peering Connection Request (Stockholm-TGW) 195
 TGW Attachment - London: Accept 199
RT of Stockholm-TGW 201
RT of London-TGW 203
RT of TGW-London-VPC-RT 205
RT of TGW-London-VPN-RT 205
RT of Stockholm-EC2-RT 206
RT of NWKT-Prod-Public 207
Verify IP Connection 207
TGW Peering Pricing 208
Summary 209

Chapter 9: VPC Peering 211

Introduction 211
Configure VPC Peering 213
Update Route Tables 218
Test Connectivity 223

Chapter 10: AWS PrivateLink 225

Introduction 225
Create Network Load Balancer 226
Create Endpoint Service 237
Create Endpoint 241
Connection Verification 249
Billing 253

Chapter 11: Dedicated Direct Connect & Transit VIF 255

- Introduction 255
- Dedicated Direct Connect Connection 255
 - Direct Connect Ordering Process* 256
 - Create Direct Connect Gateway* 264
 - Create Transit Virtual Interface* 266
 - Configure BGP Peering Between Routers* 271
 - Associate TGW with Direct Connect GW* 272
 - Direct Connect Gateway – Traffic Flow* 277

Chapter 12: Hosted Direct Connect 279

- Introduction 279
- Network Edge 280
 - BGP EVPN Control Plane Operation* 280
 - VXLAN Data Plane* 283

Chapter 13: Direct Connect BGP Policy 287

- Introduction 287
 - BGP Route Selection Process* 287
 - DXGW Egress Policy - BGP Summary Route* 289
 - DXGW Egress Policy – BGP AS-Path Prepend* 290
 - DXGW Egress Policy - BGP Communities* 292
 - On-Prem DC Egress Policy* 294

Chapter 14: Direct Connect SiteLink 297

- Introduction 297
 - SiteLink Disabled on VIFs 297
 - SiteLink Enabled on VIFs 298
 - Enabling SiteLink 299
 - SiteLink Implementation 300
 - References 302

Chapter 15: Direct Connect - Public Virtual Interface 303

- Introduction 303
 - BGP Updates from on-Prem to AWS 303
 - BGP Updates from AWS to on-Prem 305
 - Create Public VIF 306
 - References 309

Chapter 1: Virtual Private Cloud - VPC

VPC

VPC Introduction

AWS Virtual Private Cloud (VPC) is a virtual Datacenter for Amazon Elastic Cloud Compute instances (EC2) within AWS Region. AWS Regions, in turn, belongs to the global AWS Cloud environment. Each AWS Region consists of Availability Zones (AZ), which are isolated locations. AZ, in turn, has two more physical Datacenters. At the time of writing, Seoul and Tokyo have four, and Northern Virginia has six AZs. All other AWS regions have three AZs. VPC spans over regional AZs but not between AWS Regions. In other words, VPCs are region-specific virtual networks.

A VPC has to have a CIDR (Classless Interdomain Routing) IP block attached to it. The VPC CIDR defines the IP range, which we can use when creating subnets to VPC. CIDR range is VPC specific and can overlap with other VPC's CIDR range. If you create a VPC-to-VPC connection, you need a unique CIDR range per VPC.

We can allocate subnets for EC2 instances from the VPC's CIDR range. Subnets are AZ-specific, and they can't be span from one AZ to another. Subnets are classified either as Public Subnets or Private Subnets. Public Subnet has a route to Internet GW (Internet Gateway) in its Routing Table (RT). EC2 instances launched in a Public Subnet have to have a public IPv4 address in order to have an Internet connection. Note that IPv6 addresses are always assigned from the public address space. EC2 launched in a Private Subnet doesn't have a public IPv4 address. Their Internet connection goes via NAT GW. To allow Internet connection to EC2 instances in Private Subnet, we need to add a route to NAT GW into the Private Subnet Routing Table. We can use a stateful, egress-Only Internet GW for EC2 using an IPv6 address. This way, IPv6 EC2 instances have an Internet connection, but hosts on the internet can't initiate a connection to EC2. IP connectivity between EC2 instances within VPC is established between private IP address even if one of the EC2s is attached to Public Subnet and has a Public IP address. VPC has a main Routing Table (RT) and subnet-specific RTs.

Each VPC also has a default Network Access Control List (NACL). The default NACL is bind to all subnets in VPC by default. NACL is stateless by nature, traffic to and from the subnet has to be allowed in both inbound and outbound directions. The default NACL allows all ingress/egress traffic.

Figure 1-1 illustrates our example VPC and its relationship to AWS Availability Zones, AWS Regions, and AWS Account. When we create VPC, we first have to log on to our AWS account. Next, we select an AWS Region, in our case Europe (London) eu-west-2. Then we choose Availability Zones for subnets. In our case, network 10.10.0.0/24 is a Public Subnet in the AZ eu-west-2c, and network 10.10.1.0/24 is a Private Subnet in the AZ eu-west-2a. As the last step, we create subnet-specific Routing Tables where we can later add subnet-specific routes.

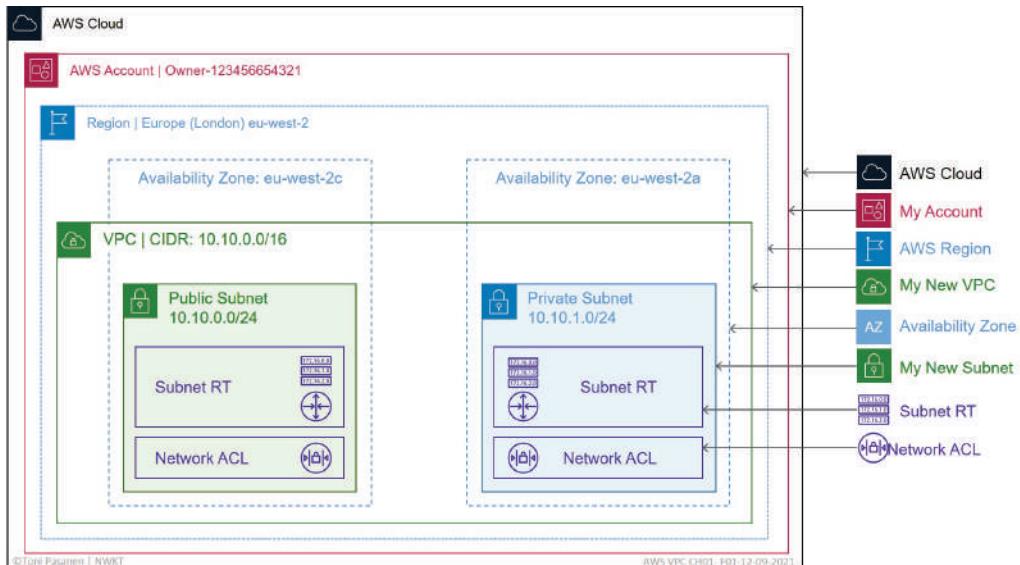


Figure 1-1: Virtual Private Cloud (VPC) Basic Building Blocks.

The Structure of Availability Zone

Figure 1-2 is an overview of an AWS Region. The physical design of network devices within the Availability Zone is based on a routed 3-tier Clos topology. The primary task of Underlay Network infrastructure is to provide fast and resilient IP connections between Hosts, Mapping services, Gateways like Internet GW, and Endpoints like S3 Endpoint.

There is a software router within each host. The virtual router's vNIC uses the physical NIC of a host as an uplink to switching fabric. Virtual routers are responsible for the encapsulation/decapsulation process (add/remove VPC header) of data traffic. The underlying network infrastructure is unaware of VPCs, and switches make a forwarding decision for encapsulated data packets based on the destination IP address in the outer tunnel IP header. The Control-Plane solution for EC2 instance reachability information relies on Mapping services. Each host registers its locally running EC2 instances to Mapping Service. The mapping message includes information about EC2's IP/MAC addresses and its VPC in addition (identifier) to the IP address of a physical host (location). The Mapping Service publishes the information when requested by hosts. Hosts, in turn, caches the mapping information to minimize the latency.

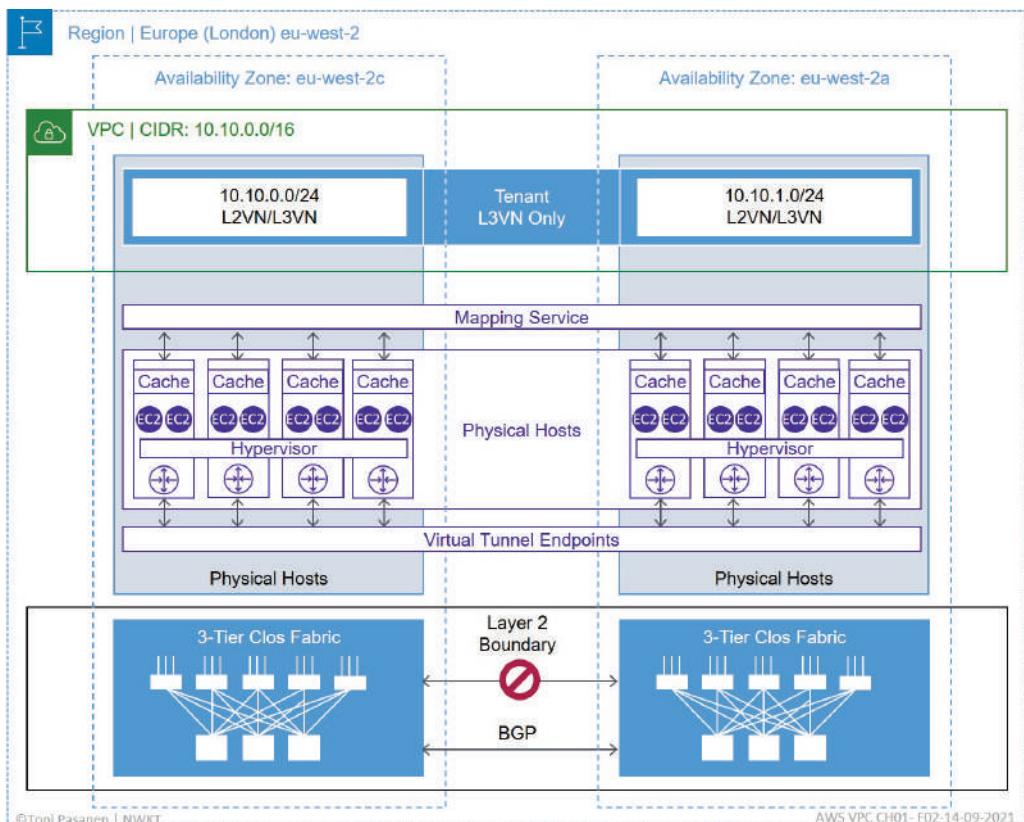


Figure 1-2: Overview of Availability Zones.

Create VPC - AWS Console

The first thing to do when we create a VPC is to log in to the AWS console. Then we select the AWS Region where we want to launch our VPC. We are going to use VPC Region *Europe (London) eu-west-2*. As the last step, we give the name to VPC and associate a CIDR block 10.10.0.0/16 to it.

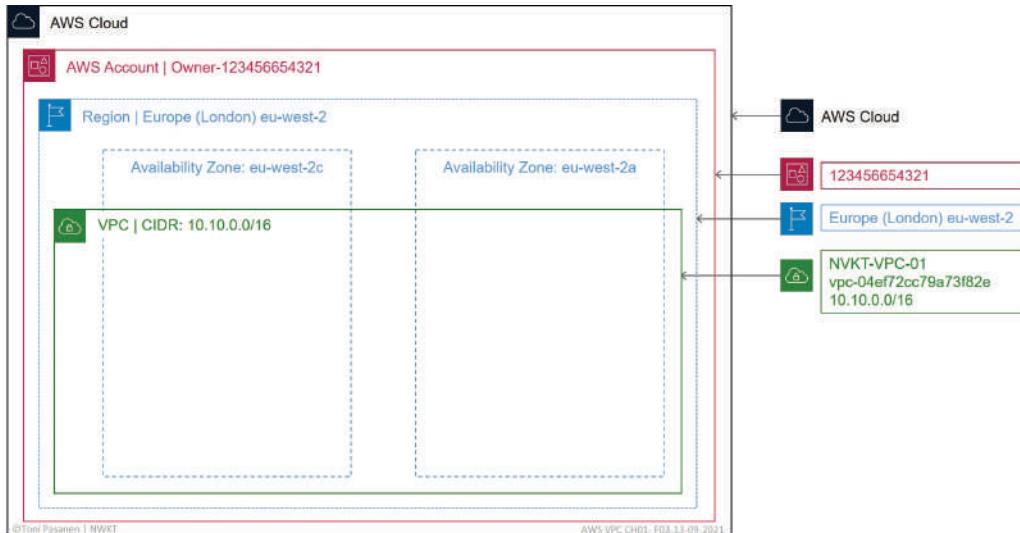


Figure 1-3: Virtual Private Cloud (VPC) – Example VPC.

Select Region

Figure 1-4 shows the AWS Management Console main window. We are going to create our VPC to the AWS Region Europe (London) eu-west-2. As a first step, we need to pick up the selected region from the drop-down menu. Then we click the VPC hyperlink from the *AWS services* window. VPC hyperlink leads us to the VPC Dashboard.

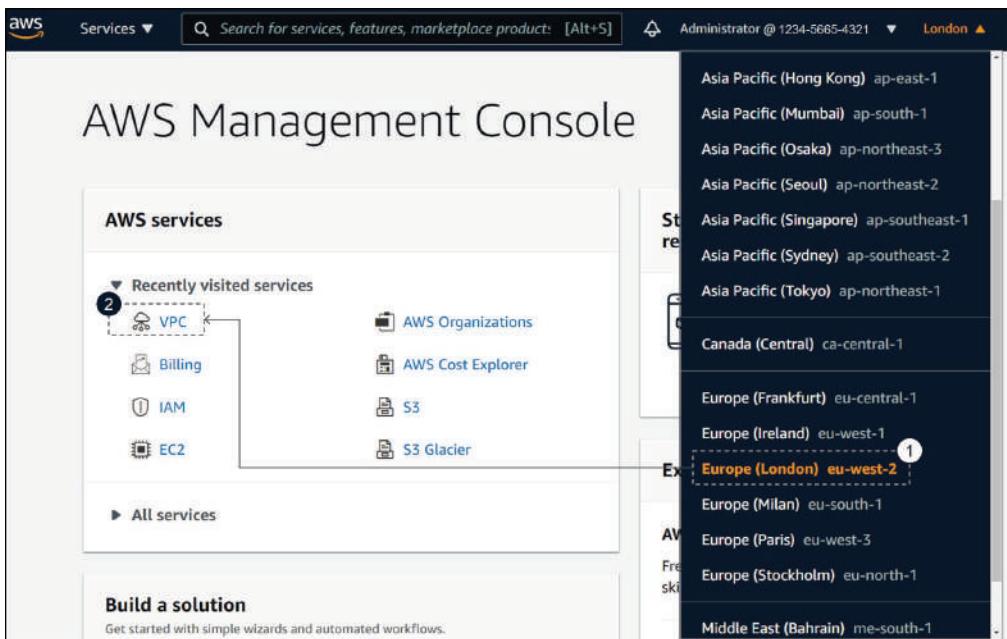


Figure 1-4: Default VPC Components.

The section on the left in the VPC Dashboard window lists all resources based on their categories (VPC, Security, Reachability, etc.). The right column lists the same resources (with some expectation) and the count of launched resources and related AWS region. The right column shows that there is one VPC, three subnets, a route table, an Internet Gateway, DHCP Options Sets, Network ACL, and Security Group already in place. All of these are related to pre-defined *Default VPC*. Using the Default VPC is the simplest way to start using EC2 instances. The Default VPC has three subnets with /20 mask from the CIDR 172.31.0.0/16. Subnet 172.31.0.0/20 is attached to AZ eu-west-2c. Subnet 172.31.16.0/20 is bind to AZ eu-west-2a, and the third subnet 172.16.32.0/20 is attached in AZ eu-west-2b. Default VPC also has an Internet Gateway and a default route (0.0.0.0/0) towards it in the main routing table. If we launch an EC2 instance in default VPC, we only need to assign a public IP address to EC2 to allow an Internet connection. Instead of using the default VPC, we are going to create a new VPC. Even if you don't use the default VPC, do NOT delete it. If you accidentally delete it, you can re-recreate it, but it is better not to do it. The default VPC, like any other default resource, doesn't have a name by default. I have named all default resources using the DFLT prefix and the suffix based on the resource. As an example, I have named the default routing table DFLT-RTBL and the default VPC DFLT-VPC. If we don't name resources, they are shown only with their resource id in drop-down menus.

For example, when attaching a subnet to the VPC, the available VPCs are listed with their relative VPC-Ids, like `vpca-cfbac1a7`. When adding a name to VPC, it is shown like `vpca-cfbac1a7 | DFLT-VPC`. To proceed, click either *Your VPC* or *VPCs* hyperlinks (3).

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar with various navigation links under categories like VIRTUAL PRIVATE CLOUD, SECURITY, REACHABILITY, and DNS FIREWALL. A callout bubble labeled '3' points from the 'Your VPCs' link in the VPC section to the 'VPCs' resource card in the main content area. The main content area is titled 'Resources by Region' and shows a grid of Amazon VPC resources. Each resource card includes the resource type, count (e.g., London 1 or London 0), and a 'See all regions' link.

Resources by Region	
VPCs	London 1 See all regions ▾
NAT Gateways	London 0 See all regions ▾
Subnets	London 3 See all regions ▾
Route Tables	London 1 See all regions ▾
VPC Peering Connections	London 0 See all regions ▾
Internet Gateways	London 1 See all regions ▾
Network ACLs	London 1 See all regions ▾
Egress Only Internet Gateways	London 0 See all regions ▾
Security Groups	London 1 See all regions ▾
Carrier Gateways	London 0 See all regions ▾
Customer Gateways	London 0 See all regions ▾
DHCP Options Sets	London 1 See all regions ▾
Virtual Private Gateways	London 0 See all regions ▾
Elastic IPs	London 0 See all regions ▾
Site-to-Site VPN Connections	London 0 See all regions ▾
Managed Prefix Lists	London 0 See all regions ▾
Endpoints	London 0 See all regions ▾
Running Instances	London 0 See all regions ▾
Endpoints	London 0 See all regions ▾
Endpoint Services	London 0 See all regions ▾

AWS VPC CH01-F05-16-09-2021

Figure 1-5: AWS Resources.

Create VPC

Figure 1-6 shows the pre-defined Default VPC and its details. We can create a new VPC by clicking the *Create VPC* button (4).

The screenshot shows the AWS VPC console interface. At the top, there is a header bar with the title 'Your VPCs (1/1) Info' and a 'Create VPC' button. A large orange circle labeled '4' is positioned over the 'Create VPC' button. Below the header is a search bar with the placeholder 'Filter VPCs'. The main table lists one VPC entry:

Name	VPC ID	State	IPv4 CIDR
DFLT-VPC	vpc-cfbac1a7	Available	172.31.0.0/16

Below the table, a modal window is open for the VPC 'vpc-cfbac1a7 / DFLT-VPC'. The modal has tabs for 'Details', 'CIDs', 'Flow logs', and 'Tags', with 'Details' selected. The 'Details' section contains the following information:

Setting	Value	Setting	Value
VPC ID	vpc-cfbac1a7	State	Available
Tenancy	DHCP options set	DNS hostnames	Main route table
Default	dopt-09217361 / DFLT-DOPT	RTB	Main network ACL
Default VPC	IPv4 CIDR	IPv6 pool	acl-57cc963f / DFLT-NACL
Yes	172.31.0.0/16	-	
Route 53 Resolver DNS	Owner ID		IPv6 CIDR (Network border group)
Firewall rule groups	123456654321	-	-

A vertical watermark on the right side of the modal reads 'AWS VPC CH01_F06-16-09-2021'.

Figure 1-6: Default VPC Components.

We give the name NVKT-VPC-01 for our VPC (5). The *Name Tag* is a Key/Value pair where the Key=Name, and the Value=NVKT-VPC-01 (8). We are going to use the CIDR block 10.10.0.0/16 (6). We don't use IPv6 addressing in this VPC. We do not need dedicated hardware for our EC2 instances, and we leave the Tenancy value to *Default* (7).

If you need dedicated hardware to run EC2 instances on, select the *Dedicated* option from the drop-down menu. Next, click the *Create VPC* button (9).

The screenshot shows the 'Create VPC' wizard in the AWS Management Console. The steps are numbered 5 through 9.

- Step 5:** VPC settings. A name tag 'NVKT-VPC-01' is entered.
- Step 6:** IPv4 CIDR block. The CIDR block '10.10.0.0/16' is specified.
- Step 7:** Tenancy. The 'No IPv6 CIDR block' option is selected.
- Step 8:** Tags. A tag 'Name: NVKT-VPC-01' is added.
- Step 9:** Create VPC button.

Callouts provide additional context for the IPv4 CIDR block and Tenancy fields.

IPv4 CIDR block

You must specify an IPv4 address range for your VPC. Specify the IPv4 address range as a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. A CIDR block size must be between a /16 netmask and /28 netmask.

Tenancy

You can run instances in your VPC on single-tenant, dedicated hardware. Select Dedicated to ensure that instances launched in this VPC are dedicated tenancy instances, regardless of the tenancy attribute specified at launch. Select Default to ensure that instances launched in this VPC use the tenancy attribute specified at launch.

AWS VPC CH01- F07-16-09-2021

Figure 1-7: Create VPC.

Figure 1-8 shows our new VPC `vpc-04ef72cc79a73f82e` / NVKT-VPC-01. Under the *DHCP options set*, *Main route table* and *Main network ACL* are hyperlinks for each service.

The screenshot shows the AWS VPC console interface. At the top, a green banner displays the message: "You successfully created vpc-04ef72cc79a73f82e / NVKT-VPC-01". Below this, the breadcrumb navigation shows: VPC > Your VPCs > vpc-04ef72cc79a73f82e. The main title is "vpc-04ef72cc79a73f82e / NVKT-VPC-01". On the right, there is an "Actions" dropdown menu. The "Details" tab is selected, showing the following configuration:

VPC ID vpc-04ef72cc79a73f82e	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP options set dopt-09217361 / DFLT-DOPT	Main route table rtb-069ac98ac692271fe	Main network ACL acl-0dfc4c4ef28ae6491
Default VPC No	IPv4 CIDR 10.10.0.0/16	IPv6 pool -	IPv6 CIDR (Network border group) -
Route 53 Resolver DNS Firewall rule groups	Owner ID 123456654321		

Below the details, there are tabs for "CIDRs", "Flow logs", and "Tags". The "CIDRs" tab is active, showing the IPv4 CIDR information:

CIDR	Status
10.10.0.0/16	Associated

The "IPv6 CIDRs" section shows the message: "You have no IPv6 CIDR blocks associated with your VPC."

AWS VPC CH01 F08-16-09 2021

Figure 1-8: Default VPC Components.

DHCP Options Set

DHCP Options Sets defines the Domain name and DNS server. You can also define the NTP server. Note that we don't have to specify DHCP pools for EC2 instances. The IP address for EC2 using DHCP will get the IP address and default GW from the subnet range where we launched it.

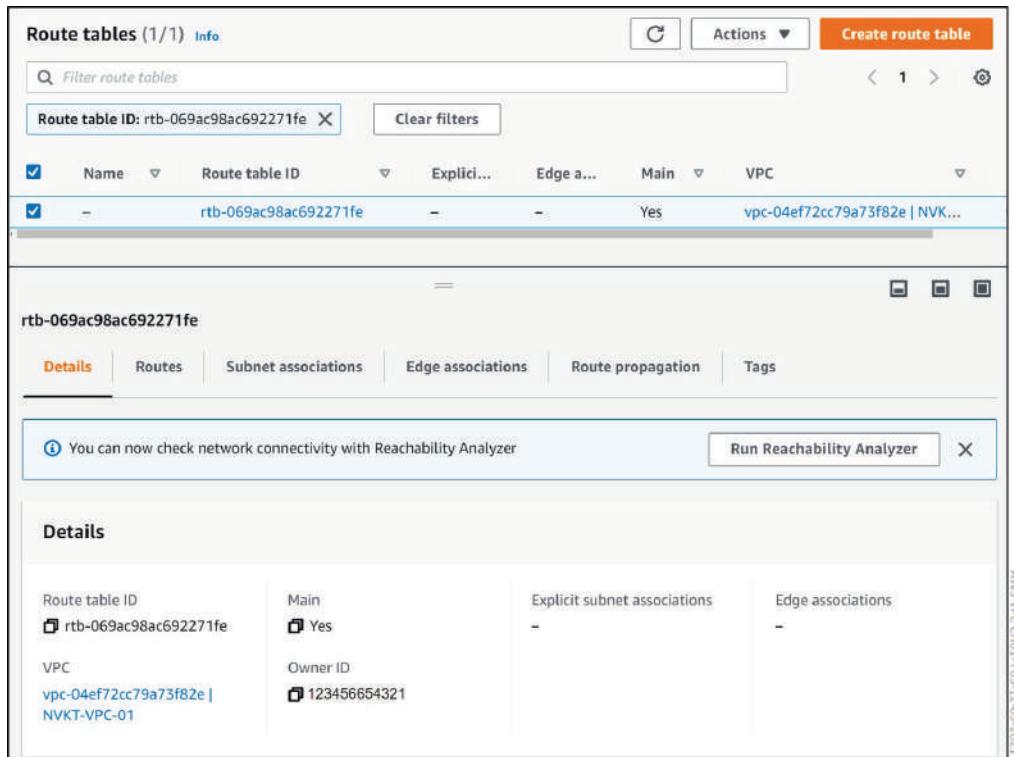


Figure 1-9: DHCP Options Set.

Main Route Table

The *Details* tab in figure 1-10 shows us that this is the main route table for VPC NVKT-VPC-01. It also shows that we haven't associate any subnets with it. When creating a new subnet, we will assign a dedicated, subnet-specific routing table. The reason for the dedicated route table is that we are going to use Internet GW for bi-directional Internet Connection in Public Subnet and NAT GW for uni-directional (egress-only) Internet access for Private Subnets. If we associate both subnets with the VPC main route table, we can only add one default route that we can use in all associated subnets. The *Routes* sheet show the actual routes. There is only one route, the local route 10.10.0.0/16 (VPC CIDR block). This route is for Inter-VPC connection, and it makes sure that all EC2 instances within VPC have IP reachability between each other.

The screenshot shows the AWS VPC Route Tables Details page. At the top, there is a search bar with the placeholder "Filter route tables" and a "Create route table" button. Below the search bar, a table lists one route table: NWKT-MAIN-RT with Route table ID rtb-069ac98ac692271fe. The table columns include Name, Route table ID, Explicit..., Edge a..., Main, and VPC. The "Main" column for this route table shows a checkmark and the value "Yes". The "VPC" column shows "vpc-04ef72cc79a73f82e | NVK...". Below the table, the route table ID "rtb-069ac98ac692271fe" is displayed. A navigation bar below the ID includes tabs for "Details", "Routes", "Subnet associations", "Edge associations", "Route propagation", and "Tags". A message box indicates that network connectivity can be checked using the Reachability Analyzer, with a "Run Reachability Analyzer" button and a close button. The "Details" section contains information about the route table, including its ID, being the main route table ("Main Yes"), and its association with a specific VPC and owner ID. A dashed box highlights the "Explicit subnet associations" and "Edge associations" fields, both of which are currently empty. The bottom right corner of the page has a watermark: "AWS VPC CH01 - F01.11.09-2021".

Figure 1-10: VPC Route Table: Details.

The screenshot shows the AWS VPC Route Table: Routes page for the same route table. At the top, there is a search bar with the placeholder "Filter routes" and a dropdown menu set to "Both". Below the search bar, a table lists one route entry: "10.10.0.0/16" with a target of "local". The table columns include Destination, Target, Status, and Propagated. The "Status" column shows "Active" with a green checkmark, and the "Propagated" column shows "No". The bottom right corner of the page has a watermark: "AWS VPC CH01 - F01.11.09-2021".

Figure 1-11: VPC Route Table: Routes.

VPC Verification Using AWS CLI

We can verify our VPC configuration by using AWS CLI. Example 1-1 shows the output for command **aws ec2 describe-vpc** in JSON format. This command lists all our VPC resources with their properties. The first one is the newest VPC NVKT-VPC-01, and the second one is the default VPC which I have named DFLT-VPC. The first VPC gets ordinal zero [0], and the second VPC gets number one [1]. Note that ordinal numbers are not shown in the output. VPC properties describe the VPC-specific CIDR Block, DHCP Options, VPC Identifier, Owner Id, CIDR Block Association, and Tags.

```
aws ec2 describe-vpcs
{
    "Vpcs": [
        {
            "CidrBlock": "10.10.0.0/16",
            "DhcpOptionsId": "dopt-09217361",
            "State": "available",
            "VpcId": "vpc-04ef72cc79a73f82e",
            "OwnerId": "123456654321",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-0379c0e3e854f43ff",
                    "CidrBlock": "10.10.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "NVKT-VPC-01"
                }
            ]
        },
        {
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "dopt-09217361",
            "State": "available",
            "VpcId": "vpc-cfbac1a7",
            "OwnerId": "123456654321",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-89d487e1",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ]
        }
    ]
}
```

```

        }
    ],
    "IsDefault": true,
    "Tags": [
        {
            "Key": "Name",
            "Value": "DFLT-VPC"
        }
    ]
}
]

```

Example 1-1: AWS CLI: Retrieve VPC Information.

We can use filters for retrieving information only from some specific resources. The command **aws ec2 describe-vpcs --filters Name=tag:Name,Values=NVKT-VPC-01** shows VPCs where we have attached the Key/Value pair Name/NVKT-VPC-01.

```

aws ec2 describe-vpcs --filters Name=tag:Name,Values=NVKT-VPC-01
{
    "Vpcs": [
        {
            "CidrBlock": "10.10.0.0/16",
            "DhcpOptionsId": "dopt-09217361",
            "State": "available",
            "VpcId": "vpc-04ef72cc79a73f82e",
            "OwnerId": "123456654321",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-0379c0e3e854f43ff",
                    "CidrBlock": "10.10.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "NVKT-VPC-01"
                }
            ]
        }
    ]
}

```

Example 1-2: AWS CLI: Retrieve VPC Information.

We can also query resource-specific information using the command **aws ec2 describe-vpcs --query "Vpcs[0]"**. The zero within square brackets after the resource Vpcs identifies the ordinal number of a resource. In our example, VPC NVKT-VPC-01 is the first one, and it has an ordinal number zero.

```
aws ec2 describe-vpcs --query "Vpcs[0]"
{
    "CidrBlock": "10.10.0.0/16",
    "DhcpOptionsId": "dopt-09217361",
    "State": "available",
    "VpcId": "vpc-04ef72cc79a73f82e",
    "OwnerId": "123456654321",
    "InstanceTenancy": "default",
    "CidrBlockAssociationSet": [
        {
            "AssociationId": "vpc-cidr-assoc-0379c0e3e854f43ff",
            "CidrBlock": "10.10.0.0/16",
            "CidrBlockState": {
                "State": "associated"
            }
        }
    ],
    "IsDefault": false,
    "Tags": [
        {
            "Key": "Name",
            "Value": "NVKT-VPC-01"
        }
    ]
}
```

Example 1-3: AWS CLI: Retrieve VPC Information.

If we want to see only some specific resource properties, we can add the properties after the resource, separated by a dot. Example 1-4 shows how we can see the CIDR Block Association for VPC NVKT-VPC-01 (ordinal zero).

```
aws ec2 describe-vpcs --query "Vpcs[0].CidrBlockAssociationSet"
[
    {
        "AssociationId": "vpc-cidr-assoc-0379c0e3e854f43ff",
        "CidrBlock": "10.10.0.0/16",
        "CidrBlockState": {
            "State": "associated"
        }
    }
]
```

Example 1-4: AWS CLI: Retrieve CIDR (Properties) Association to VPC (Resource).

We can change the output representation from the JSON to table by using the option --output table. The table output is a good choice when we create documentation about VPCs. Note that you can use this option with all other commands too.

```
aws ec2 describe-vpcs --query "Vpcs[0].CidrBlockAssociationSet" --output table
```

DescribeVpcs	
AssociationId	CidrBlock
vpc-cidr-assoc-0379c0e3e854f43ff	10.10.0.0/16
CidrBlockState	
State	associated

Example 1-5: AWS CLI: Retrieve CIDR Association to VPC – Table Output.

Create VPC - AWS CloudFormation

The focus of this section is to show how we can create a VPC using AWS *CloudFormation* service. Figure 1-12 shows our example AWS *CloudFormation* Templates. Its first section, *AWSTemplateFormatVersion*, specifies the template language format. At the time of writing, 2010-09-09 is the latest and only valid version. We can use the *Description* section to describe our template. Note that it must follow the *AWSTemplateFormatVersion* Section. *AWSTemplateFormatVersion* and *Description* are optional sections. The *Resources* section specifies the actual AWS resources and their properties. Each AWS resource is identified with a *logical name*. I have given the logical name *NwktVPC* for our example VPC. We can use resource-specific logical names for defining dependencies between resources. For example, when adding the *AWS::EC2::Subnet* resource to our template, we assign the *VpcId* value by calling it from the *AWS::EC2::VPC* resource using *!REF* *intrinsic function*. I will explain the process in the Subnet section later. The resources and their properties are defined under logical names. The *Resources* section is the only required section in AWS CloudFormation-Template. AWS CloudFormation Templates are identified by using Stack Names in AWS Cloud Formation. Our example Stack Name is *MyNetworkStack*.

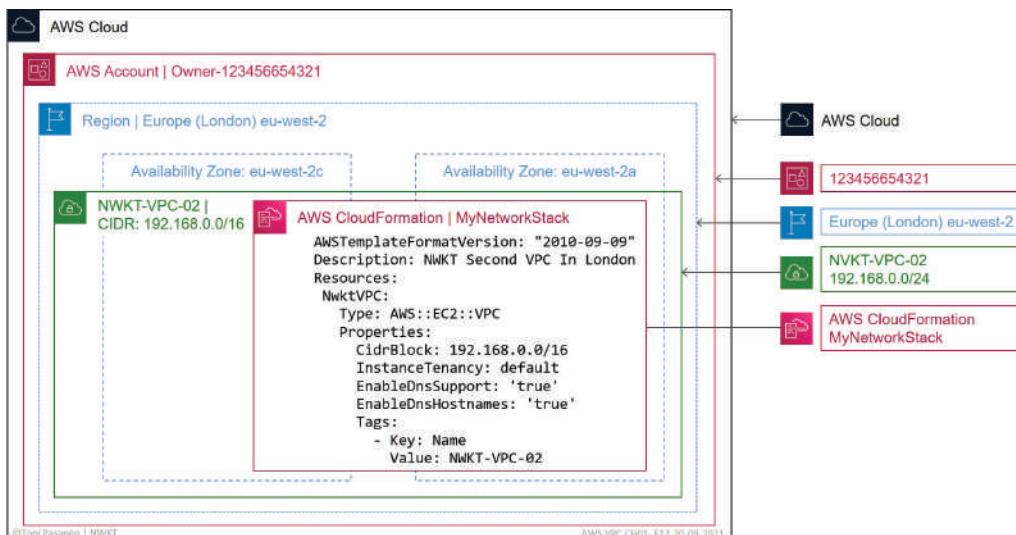


Figure 1-12: AWS *CloudFormation*: VPC.

Create Template

I have created a text file using the YAML format. The file defines the set of properties for our new VPC named NWKT-VPC-02. After writing the file, I saved it to my computer using the name MyVpc.yml. You can use your favorite repository for storing the template.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: NWKT Second VPC In London
Resources:
  NwktVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 192.168.0.0/16
      InstanceTenancy: default
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
    Tags:
      - Key: Name
        Value: NWKT-VPC-02
```

Example 1-6: AWS Cloud Formation Template for VPC.

Upload Template

After saving the file, I downloaded it to AWS using the Stack Name MyNetworkStack. We will receive a notification when the stack is downloaded.

```
aws cloudformation create-stack --stack-name MyNetworkStack --template-body
file://C:\Toni\AWS-CF-Templates\MyVPC.yml

{
  "StackId": "arn:aws:cloudformation:eu-west-
2:123456654321:stack/MyNetworkStack/8d42ac70-1939-11ec-81f9-06cf091d9f40"
```

Example 1-7: AWS CLI: Upload Template to AWS.

Verification Using AWS Console

We can verify that our Stack MyNetworkStack is created into AWS from the AWS Management Console. First, we select CloudFormation from the Service section.

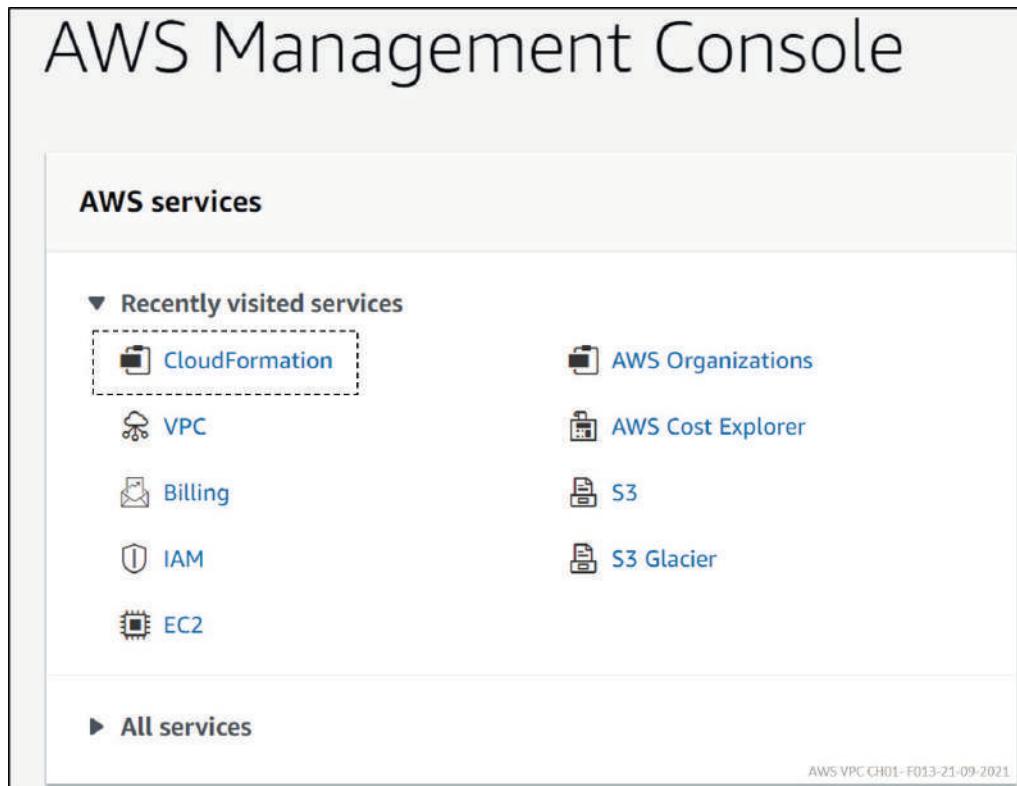


Figure 1-13: AWS management Console – CloudFormation Service.

Next, we select the Stacks option from the CloudFormation section (figure 1-14). We can see that we have one stack, MyNetworkStack, and its status is Create Complete. We can observe the stack by clicking the Stacks details option from the CloudFormation section.

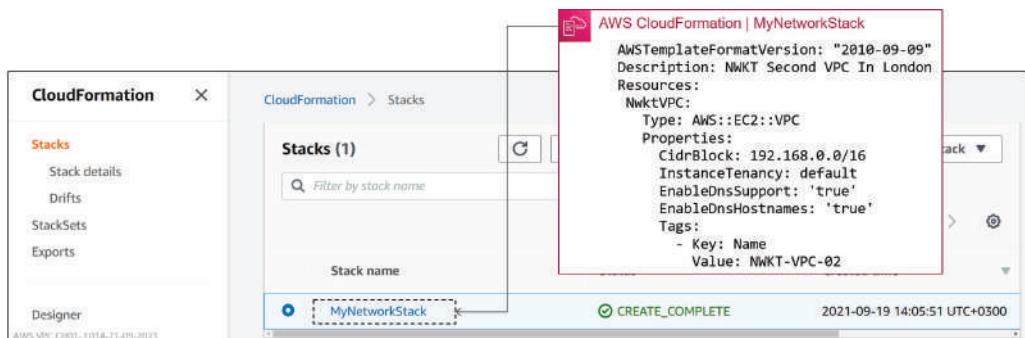


Figure 1-14: CloudFormation Stacks.

The *Stack info* tab in figure 1-15 shows the AWS assigned Amazon Resource Name for the stack and the Description we used in our template.

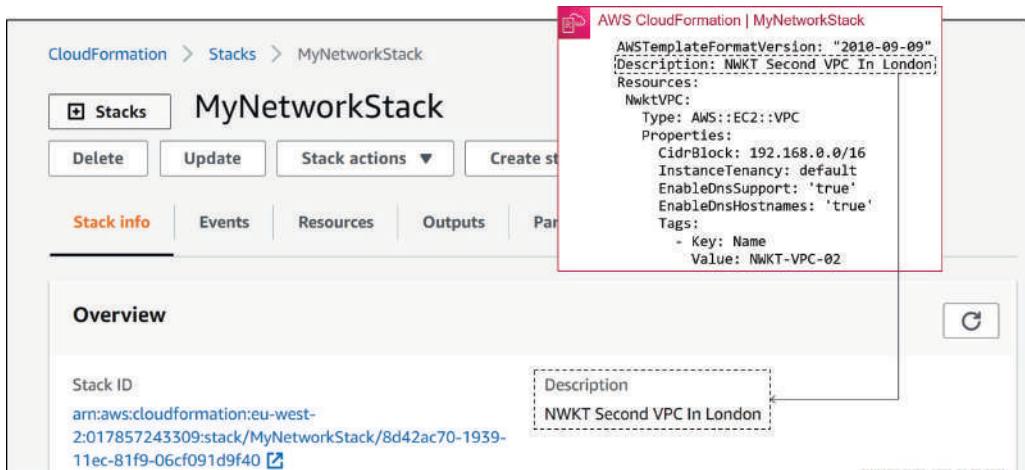


Figure 1-15: MyNetworkStack - Stack Info.

The *Resources* tab lists type of resources along with their Logical and Physical Identifier.

The screenshot shows the AWS CloudFormation console with the stack named "MyNetworkStack". The "Resources" tab is selected. A single resource, "NwktVPC", is listed. A red box highlights this resource, showing its details: Type: AWS::EC2::VPC, Properties: CidrBlock: 192.168.0.0/16, InstanceTenancy: default, EnableDnsSupport: 'true', EnableDnsHostnames: 'true', and Tags: Name: NwKT-VPC-02. The logical ID is "NwktVPC" and the physical ID is "vpc-0687dedcf950d0de". The status is "CREATE_COMPLETE".

Logical ID	Physical ID	Type	Status
NwktVPC	vpc-0687dedcf950d0de	AWS::EC2::VPC	CREATE_COMPLETE

Figure 1-16: MyNetworkStack - Resources.

The *Template* tab in figure 1-17 shows the actual template which we previously upload to AWS.

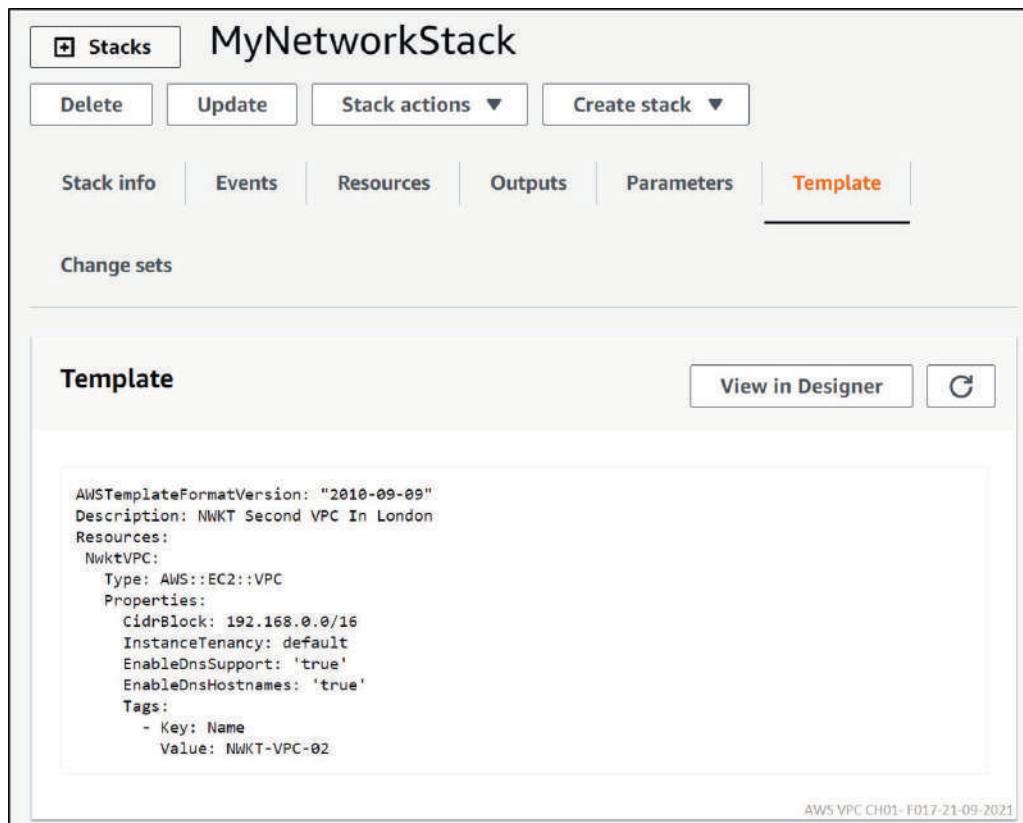


Figure 1-17: MyNetworkStack – Template.

VPC Verification using AWS CLI

We can retrieve the stack-specific information from AWS by using the command **aws cloudformation describe-stacks** (example 1-8).

```
aws cloudformation describe-stacks
{
  "Stacks": [
    {
      "StackId": "arn:aws:cloudformation:eu-west-2:123456654321:stack/MyNetworkStack/8d42ac70-1939-11ec-81f9-06cf091d9f40",
      "StackName": "MyNetworkStack",
      "Description": "NWKT Second VPC In London",
      "CreationTime": "2021-09-19T11:05:51.593Z",
      "RollbackConfiguration": {},
      "StackStatus": "CREATE_COMPLETE",
      "DisableRollback": false,
      "NotificationARNs": [],
      "Tags": [],
      "DriftInformation": {
        "StackDriftStatus": "NOT_CHECKED"
      }
    }
  ]
}
```

Example 1-8: AWS CLI: Retrieve VPC Information.

We can verify that the VPC defined in our AWS CloudFormation template is created using the AWS CLI command `aws ec2 describe-vpcs --filters Name=tag:Name,Values=NWKT-VPC-02`. Note that our VPC has three AWS assigned tags in addition to the Name tag. They describe a) the Stack-Id, b) the logical name of the VPC resource, and c) the Stack Name.

```
aws ec2 describe-vpcs --filters Name=tag:Name,Values=NWKT-VPC-02
{
  "Vpcs": [
    {
      "CidrBlock": "192.168.0.0/16",
      "DhcpOptionsId": "dopt-09217361",
      "State": "available",
      "VpcId": "vpc-0687dedcf950d0de",
      "OwnerId": "123456654321",
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "AssociationId": "vpc-cidr-assoc-05f07968b6ac29e7d",
          "CidrBlock": "192.168.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ],
      "IsDefault": false,
      "Tags": [
        {
          "Key": "Name",
          "Value": "NWKT-VPC-02"
        }
      ]
    }
  ]
}
```

```

    },
    {
        "Key": "aws:cloudformation:stack-id",
        "Value": "arn:aws:cloudformation:eu-west-
2:123456654321:stack/MyNetworkStack/8d42ac70-1939-11ec-81f9-06cf091d9f40"
    },
    {
        "Key": "aws:cloudformation:logical-id",
        "Value": "NwktVPC"
    },
    {
        "Key": "aws:cloudformation:stack-name",
        "Value": "MyNetworkStack"
    }
]
}
}

```

Example 1-9: AWS CLI: Retrieve VPC Information.

Create Subnets - AWS Console

When we have created a new VPC, we can start adding subnets to it. We are going to create two subnets. Subnet 10.10.0.0/24 is a Public Subnet in Availability Zone eu-west2c, where we later add an Internet GW. Subnet 10.10.1.0/24 is a Private Subnet in Availability Zone eu-west2a that will use a NAT GW for uni-directional Internet access. Note that subnets in AWS Cloud are not broadcast domains. Rather, they are places for instances with common routing requirements, like route to Internet.

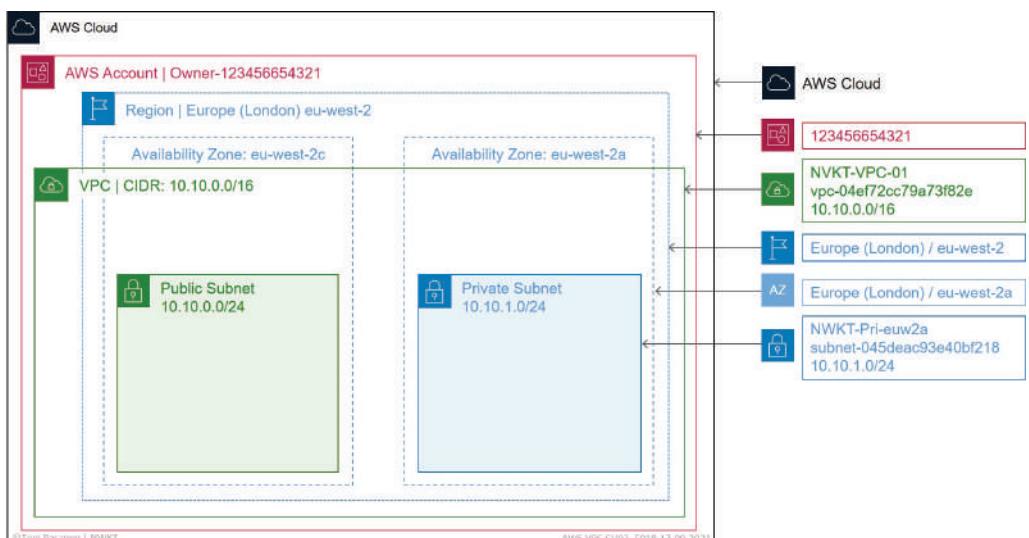


Figure 1-18: VPC Route Table: Routes.

Create Subnets

Navigate back to VPC Dashboard and click the *Subnets* hyperlink either from the left or right columns.

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under the 'VIRTUAL PRIVATE CLOUD' section, the 'Subnets' link is highlighted with a large black circle containing the number 10. The main content area displays 'Resources by Region' for the Europe (London) region. A callout box highlights the 'Subnets' resource, which shows 3 items. Other resources listed include VPCs, NAT Gateways, VPC Peering Connections, Route Tables, Network ACLs, Security Groups, Customer Gateways, DHCP options sets, Virtual Private Gateways, Site-to-Site VPN Connections, Elastic IPs, Endpoints, and Endpoint Services. Each resource item includes a 'See all regions' link.

Resource Type	Count	Region	Action
VPCs	1	London	See all regions
Subnets	3	London	See all regions
Route Tables	1	London	See all regions
Internet Gateways	1	London	See all regions
Egress Only Internet Gateways	0	London	See all regions
Carrier Gateways	0	London	See all regions
DHCP Options Sets	0	London	See all regions
Elastic IPs	0	London	See all regions
Managed Prefix Lists	0	London	See all regions
Endpoints	0	London	See all regions
Endpoint Services	0	London	See all regions
NAT Gateways	0	London	See all regions
Peering Connections	0	London	See all regions
Network ACLs	1	London	See all regions
Security Groups	1	London	See all regions
Customer Gateways	0	London	See all regions
DHCP options sets	1	London	See all regions
Virtual Private Gateways	0	London	See all regions
Site-to-Site VPN Connections	0	London	See all regions
Elastic IPs	0	London	See all regions
Endpoints	0	London	See all regions
Endpoint Services	0	London	See all regions
Running Instances	0	London	See all regions

AWS VPC CH01- F19-17-09-2021

Figure 1-19: VPC Dashboard: Subnets.

Figure 1-20 shows current subnets. All three subnets belong to default VPC. Note that I have added a name to each subnet. For example, subnet 172.31.0.0/20 is in Availability Zone eu-west-2c (euw-az1), and that is why I have used the 2c suffix after the DFLT-SUBN prefix. The *Default Subnet = Yes* in the first column in Details window verifies that the subnet 172.31.0.0/20 (subnet-039ed36a) is the default subnet. We can also see that the Route Table rtb-8edeeae6 | DFLT-RTBL and Network ACL acl-57cc963f | DFLT-NACL are associated with the subnet. We are going to create new subnets, which we then add to our VPC NVKT-VPC-01. Click the *Create subnet* button to proceed.

The screenshot shows the AWS VPC Subnets list and a detailed view of a specific subnet. The top navigation bar has tabs for 'Subnets (1/3)' and 'Info'. On the right, there are 'Actions' and a 'Create subnet' button. A circular badge with the number '11' is visible above the actions. Below the table, there's a 'Details' section for the selected subnet.

Subnets (1/3) Info

	Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 ...	Avi...
<input checked="" type="checkbox"/>	DFLT-SUBN-2c	subnet-039ed36a	Available	vpc-cfbac1a7 DFLT-VPC	172.31.0.0/20	-	4091
<input type="checkbox"/>	DFLT-SUBN-2a	subnet-cab53cb0	Available	vpc-cfbac1a7 DFLT-VPC	172.31.16.0/20	-	4091
<input type="checkbox"/>	DFLT-SUBN-2b	subnet-8d6fb5c1	Available	vpc-cfbac1a7 DFLT-VPC	172.31.32.0/20	-	4091

Details

Subnet ID <input type="text" value="subnet-039ed36a"/>	Subnet ARN <input type="text" value="arn:aws:ec2:eu-west-2:017857243309:subnet/subnet-039ed36a"/>	State Available	IPv4 CIDR <input type="text" value="172.31.0.0/20"/>
Available IPv4 addresses <input type="text" value="4091"/>	Availability Zone <input type="text" value="eu-west-2c"/>	Availability Zone ID <input type="text" value="euw2-az1"/>	Network ACL <input type="text" value="acl-57cc963f DFLT-NACL"/>
Network border group <input type="text" value="eu-west-2"/>	VPC <input type="text" value="rtb-8edeeae6 DFLT-RTBL"/>	Route table <input type="text" value="rtb-8edeeae6 DFLT-RTBL"/>	Auto-assign IPv6 address <input type="text" value="No"/>
Default subnet: <input checked="" type="checkbox" value="Yes"/>	Auto-assign public IPv4 address: <input checked="" type="checkbox" value="Yes"/>	Auto-assign customer-owned IPv4 address: <input checked="" type="checkbox" value="No"/>	Customer-owned IPv4 pool <input type="text" value=""/>
Customer-owned IPv4 pool <input type="text" value=""/>	IPv4 CIDR reservations <input type="text" value="-"/>	IPv6 CIDR reservations <input type="text" value="-"/>	Owner <input type="text" value="123456654321"/>

AWS VPC CH01- F20-17-09-2021

Figure 1-20: VPC Route Table: Routes.

We start the subnet creation process by selecting the VPC, where we are adding our new subnet. Open the VPC Id drop-down menu. There's two VPCs, the default one (vpc-cfbac1a7 | DFLT-VPC) and the one we have created (vpc-04ef72cc79a73f82 | NVKT-VPC). At this phase, we can see why naming is such an important thing to do for each AWS resource and its properties. If we have several VPCs without the name tag, it would be hard to select the right one based on just autogenerated identifier (number/letter). When we have selected the right VPC, more properties appear.

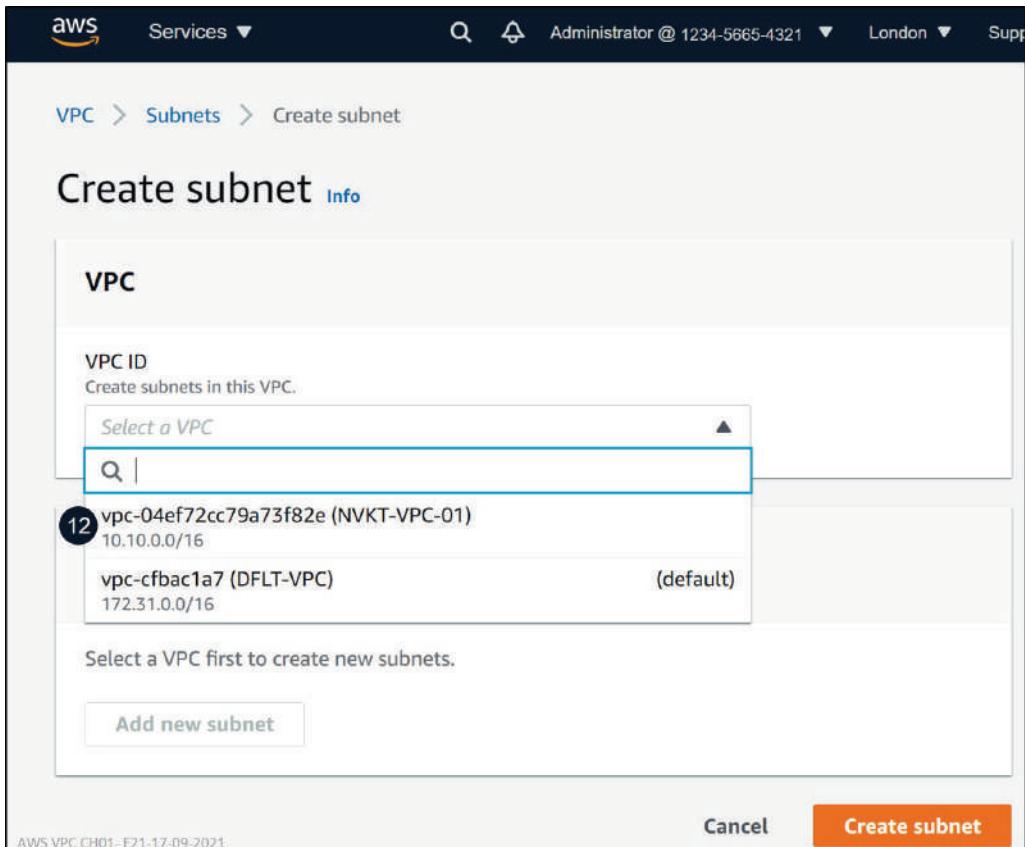


Figure 1-21: VPC Subnets: Select VPC.

After selecting the correct VPC, we need to specify the subnet. We are creating a subnet 10.10.0.0/24. We are going to add it to Availability Zone eu-west-2c. If there is no need for a specific AZ, select the No Preference option from the drop-down menu, and Amazon will randomly choose the AZ. Manual AZ selection gives you the control to decide subnet location policy based on application/business needs. In the worst-case scenario, all your critical EC2 instances might end up in the same Availability Zone (=Physical Datacenter). The subnet name I have given describes that this is a Public subnet in AZ eu-west-2c. Note that the name tag is a Key/Value pair, where the Key is the "Name", and the value is the actual name NWKT-Pub-euw2c. I will show later how we can use these tags when retrieving information from AWS using an AWS CLI.

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.
vpc-04ef72cc79a73f82e (NVKT-VPC-01)

Associated VPC CIDRs
IPv4 CIDRs
10.10.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
NWKT-Pub-euw2c
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
Europe (London) / eu-west-2c

IPv4 CIDR block Info
Q 10.10.0.0/24 X

▼ Tags - optional

Key Q Name X	Value - optional Q NWKT-Pub-euw2c X Remove
---------------------	---

Add new tag
You can add 49 more tags.

Remove

Add new subnet

AWS VPC CH01- F22-17-09-2021
Cancel
Create subnet

Figure 1-22: VPC Subnets: Select VPC.

Figure 1-23 shows that we have successfully created a new subnet. Note that there is a filter that shows only the subnet we just created. Subnet uses the main routing table (rtb-069ac98ac692271fe) of VPC NVKT-VPC-01 by default. Also, VPC's default Network ACL (acl-0dfc4c4ef28ae6491) is used with the subnet.

The screenshot shows the AWS VPC Subnets page. At the top, a green success message says "You have successfully created 1 subnet: subnet-04af160d1d0aee071". Below this, the "Subnets (1/1)" table lists one subnet:

<input checked="" type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR
<input checked="" type="checkbox"/>	NWKT-Pub-euw2c	subnet-04af160d1d0aee071	Available	vpc-04ef72cc79a73f82e NVK...	10.10.0.0/24

Below the table, the subnet details are shown in a card:

subnet-04af160d1d0aee071 / NWKT-Pub-euw2c

- Details** (selected)
- Flow logs
- Route table
- Network ACL
- CIDR reservations
- Sharing
- Tags

Details

Subnet ID subnet-04af160d1d0aee071	Subnet ARN arn:aws:ec2:eu-west-2:017857243309:subnet/subnet-04af160d1d0aee071	State Available	IPv4 CIDR 10.10.0.0/24
Available IPv4 addresses 251	IPv6 CIDR -	Availability Zone eu-west-2c	Availability Zone ID euw2-az1
Network border group eu-west-2	VPC vpc-04ef72cc79a73f82e NVKT-VPC-01	Route table rtb-069ac98ac692271fe	Network ACL acl-0dfc4c4ef28ae6491
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	Auto-assign public IPv4 address No	IPv4 CIDR reservations -	IPv6 CIDR reservations -

Figure 1-23: VPC Subnets: Select VPC.

I have created Private subnet 10.10.1.0/24 in the same way as Public subnet 10.10.0.0/24 but on the Availability Zone eu-west-2a. Figure 1-24 shows both Public subnet 10.10.0.0/24 and Private subnet 10.10.1.0/24.

Subnets (1/5) Info						
	Name	Subnet ID	State	VPC	IPv4 CIDR	Actions Create subnet
<input checked="" type="checkbox"/>	NWKT-Pub-euw2c	subnet-04af160d1d0ae071	Available	vpc-04ef72cc79a73f82e ...	10.10.0.0/24	
<input type="checkbox"/>	NWKT-Pri-euw2a	subnet-045deac93e40bf218	Available	vpc-04ef72cc79a73f82e ...	10.10.1.0/24	
<input type="checkbox"/>	DFLT-SUBN-2c	subnet-039ed36a	Available	vpc-cfbac1a7 DFLT-VPC	172.31.0.0/20	
<input type="checkbox"/>	DFLT-SUBN-2a	subnet-cab53cb0	Available	vpc-cfbac1a7 DFLT-VPC	172.31.16.0/20	
<input type="checkbox"/>	DFLT-SUBN-2b	subnet-8d6fb3c1	Available	vpc-cfbac1a7 DFLT-VPC	172.31.32.0/20	

Figure 1-24: VPC Subnets: Select VPC.

Route Tables

At this phase, we have attached subnets to their respective Availability Zones. Next, we will create subnet-specific route tables for both Public and Private subnets.

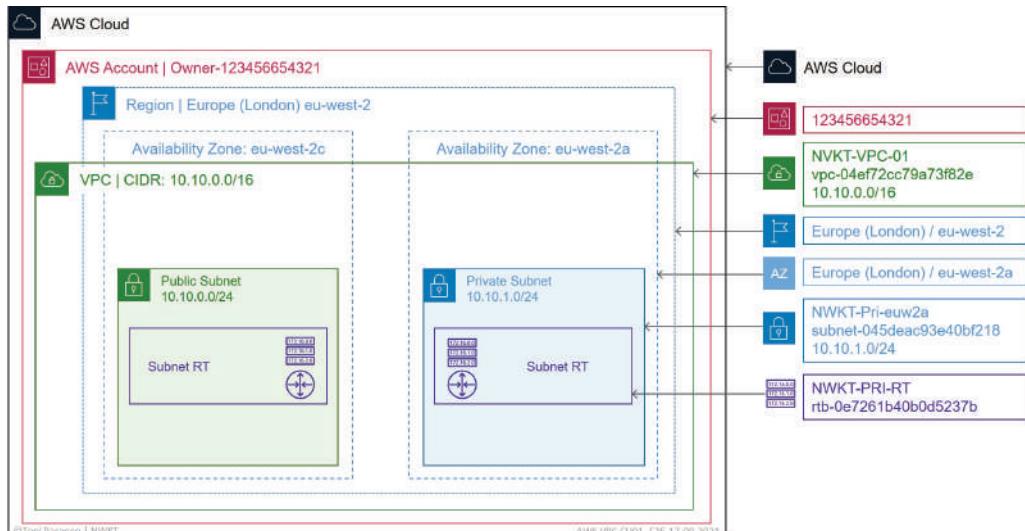


Figure 1-25: VPC Subnets: Select VPC.

Create Route Table – AWS Console

Navigate back to VPC Dashboard and click the *Route Tables* hyperlink either from the left or right columns.

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under the 'VIRTUAL PRIVATE CLOUD' section, the 'Route Tables' link is highlighted with a blue dashed box and a large number '17' indicating it is the current step. The main content area displays 'Resources by Region' for the Europe (London) region. A callout arrow points from the highlighted 'Route Tables' link to the 'Route Tables' resource card in the center of the dashboard.

Resources by Region	
Note: Your Instances will launch in the Europe (London) region.	
Launch VPC Wizard Launch EC2 Instances	
Refresh Resources	
You are using the following Amazon VPC resources.	
VPCs London 1 See all regions ▾	NAT Gateways London 0 See all regions ▾
Subnets London 3 See all regions ▾	VPC Peering Connections London 0 See all regions ▾
Route Tables London 1 See all regions ▾	Network ACLs London 1 See all regions ▾
Internet Gateways London 1 See all regions ▾	Security Groups London 1 See all regions ▾
Egress Only Internet Gateways London 0 See all regions ▾	Customer Gateways London 0 See all regions ▾
Carrier Gateways	Virtual Private Gateways London 0 See all regions ▾
DHCP Options Sets	Site-to-Site VPN Connections London 0 See all regions ▾
Elastic IPs	Running Instances London 0 See all regions ▾
Managed Prefix Lists	
Endpoints	
Endpoint Services New	
NAT Gateways	
Peering Connections New	
AWS VPC CH01- F26-13-09-2021	

Figure 1-26: Subnet Route Table.

Figure 1-27 shows our current route tables. I have given the name NWKT-MAIN-RT (rtb-069ac98ac692271fe) for the auto generated main RT of VPC-VPC-01. The figure shows that there are no subnets associated with it. To proceed, click the *Create route table button* (18).

The screenshot shows the AWS Route Tables page. At the top, there is a table titled "Route tables (1/2)" with columns: Name, Route table ID, Explicit..., Edge a..., Main, and VPC. Two rows are listed: "DFLT-RTBL" (rtb-Bedeaae6) and "NWKT-MAIN-RT" (rtb-069ac98ac692271fe). A red circle labeled "18" is placed over the "Create route table" button in the top right corner of the table header.

Below the table, a specific route table is selected: "rtb-069ac98ac692271fe / NWKT-MAIN-RT". The "Subnet associations" tab is active. It displays a section titled "Explicit subnet associations (0)" with a "Find subnet association" search bar and an "Edit subnet associations" button. Below this, a message states "No subnet associations" and "You do not have any subnet associations.".

Further down, a section titled "Subnets without explicit associations (2)" is shown. It includes a "Find subnet association" search bar and an "Edit subnet associations" button. It lists two subnets: "subnet-04af160d1d0aee071 / NWKT-Pub-euw2c" (IPv4 CIDR: 10.10.0.0/24) and "subnet-045deac93e40bf218 / NWKT-Pri-euw2a" (IPv4 CIDR: 10.10.1.0/24). A vertical watermark "AWS VPC CHAT-RT-13-09-2021" is visible on the right side of the interface.

Figure 1-27: Subnet Route Table.

First, give the name to the route table. Then, select the correct VPC from the drop-down menu. After choosing the VPC, click the *Create route table* button. You can also create new tags for subnets. For example, the tag can describe the purpose of the route table. As an example, we can add Key = Purpose, Value = NWKT-WEB-SRV-RT. Note that each tag can have only one value. The maximum length for key/value pairs is 128/256 characters which are case-sensitive.

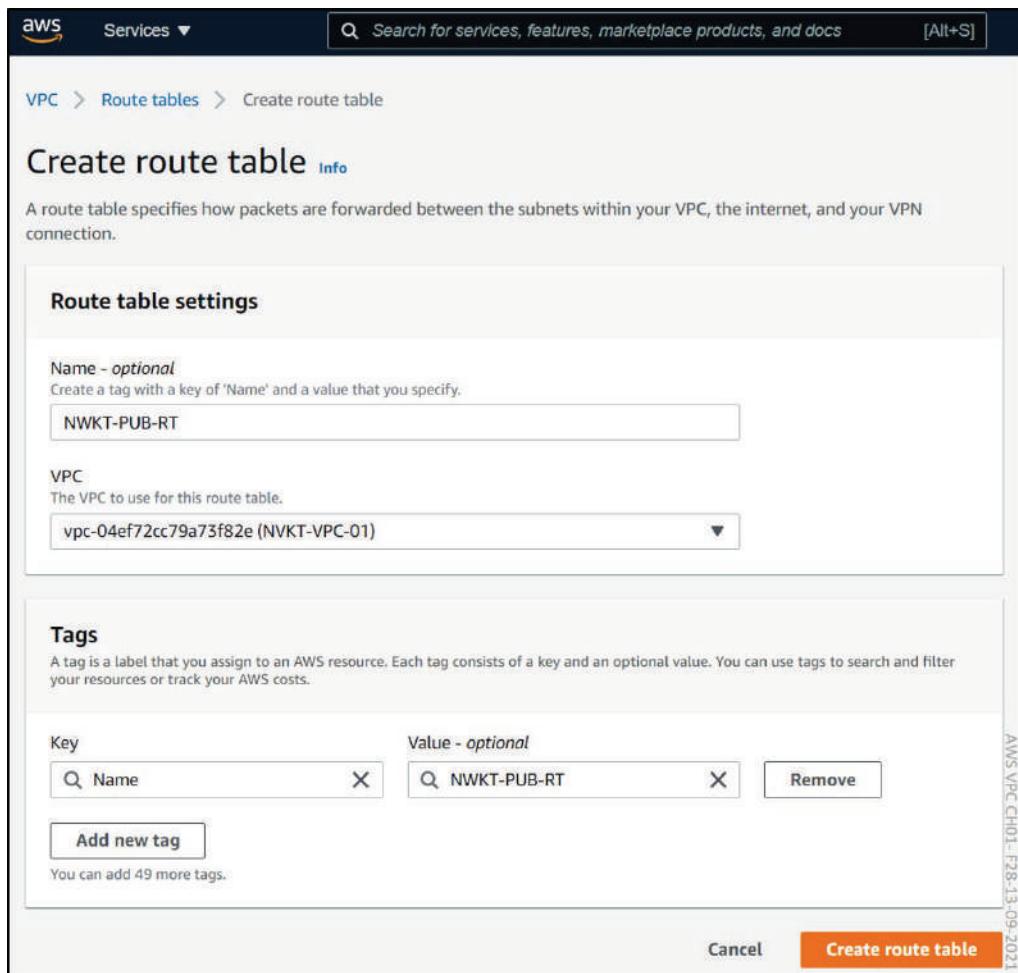


Figure 1-28: Subnet Route Table.

Figure 1-29 shows that we have successfully created a new route table rtb-0fd4639034844c3ea | NWKT-PUB-RT. Now we can do the subnet association. At the moment, the route table has only one local route.

The screenshot shows the AWS Route Tables page. At the top, a green banner indicates: "Route table rtb-0fd4639034844c3ea | NWKT-PUB-RT was created successfully." Below the banner, the navigation path is "VPC > Route tables > rtb-0fd4639034844c3ea". The main title is "rtb-0fd4639034844c3ea / NWKT-PUB-RT". On the right, there is an "Actions" dropdown menu. A callout box says, "You can now check network connectivity with Reachability Analyzer", with a "Run Reachability Analyzer" button. Below this, there are two tabs: "Details" and "Info". Under "Details", there are four sections: "Route table ID" (rtb-0fd4639034844c3ea), "Main" (No), "Explicit subnet associations" (empty), and "Edge associations" (empty). The "VPC" section shows "Owner ID" (123456654321) and "VPC-04ef72cc79a73f82e | NVKT-VPC-01". At the bottom of the "Details" section, there are tabs for "Routes", "Subnet associations", "Edge associations", "Route propagation", and "Tags", with "Routes" being the active tab. The "Routes" table has one entry: Destination 10.10.0.0/16, Target local, Status Active, and Propagated No. There is also a "Filter routes" search bar and a "Edit routes" button. On the far right edge of the screenshot, there is a vertical watermark: "AWS VPC CH1-F19-13-05-2021".

Figure 1-29: Subnet Route Table.

The Subnet association tab in the route table window (figure 1-30) shows that we haven't associate subnets to the route table. The next step is to associate our Public subnet 10.10.0.0/24 with this route table. Click the *Edit subnet association* button.

VPC > Route tables > rtb-0fd4639034844c3ea

rtb-0fd4639034844c3ea / NWKT-PUB-RT

Actions ▾

Details <small>Info</small>	Route table ID: rtb-0fd4639034844c3ea	Main: No	Explicit subnet associations: -	Edge associations: -
VPC: vpc-04ef72cc79a73f82e NVKT-VPC-01	Owner ID: 123456654321			

Routes | **Subnet associations** | Edge associations | Route propagation | Tags

Explicit subnet associations (0)

Edit subnet associations

Find subnet association

Subnet ID	IPv4 CIDR	IPv6 CIDR
No subnet associations		
You do not have any subnet associations.		

Subnets without explicit associations (2)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Edit subnet associations

Find subnet association

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-04af160d1d0aee071 / NWKT-Pub-euw2c	10.10.0.0/24	-
subnet-045deac93e40bf218 / NWKT-Pri-euw2a	10.10.1.0/24	-

AWS VPC CLOUD LEARNER

Figure 1-30: Subnet Route Table.

Select the subnet NWKT-Pub-euw2c and click the Save association button.

The screenshot shows the AWS VPC Route Tables interface. The top navigation bar includes 'VPC', 'Route tables', 'rtb-0fd4639034844c3ea', and 'Edit subnet associations'. The main title is 'Edit subnet associations' with the sub-instruction 'Change which subnets are associated with this route table.' Below this is a table titled 'Available subnets (1/2)' with two rows:

Name	Subnet ID	IPv4 CIDR	Route table ID
<input checked="" type="checkbox"/> NWKT-Pub-euw2c	subnet-04af160d1d0aee071	10.10.0.0/24	rtb-069ac98ac692271fe / NWKT-MAIN-RT
<input type="checkbox"/> NWKT-Pri-euw2a	subnet-045deac93e40bf218	10.10.1.0/24	Main (rtb-069ac98ac692271fe / NWKT-MAIN-RT)

Below the table is a section titled 'Selected subnets' containing a single item: 'subnet-04af160d1d0aee071 / NWKT-Pub-euw2c' with a delete icon (X). At the bottom right are 'Cancel' and 'Save associations' buttons. A small footer note at the bottom left reads 'AWS VPC CH01 - F11-13-09-2021'.

Figure 1-31: Subnet Route Table.

Figure 1-32 shows that now the subnet NWKT-Pub-euw2c (10.10.0.0/24) is associated with route table NWKT-PUB-RT. The Private subnet NWKT-Pri-euw2a is still without explicit association. As a next step, we associate it with the route table NWKT-PRI-RT. Figure 1-33 shows the route table association after associating both subnets with correct route tables.

The screenshot shows the AWS VPC Route Tables page. At the top, there is a search bar labeled "Filter route tables" and a "Create route table" button. Below the search bar, a table lists one route table:

Name	Route table ID	Explicit subnet associations	Main
NWKT-PUB-RT	rtb-0fd4639034844c3ea	subnet-04af160d1d0aee071 / NWKT-Pub-euw2c	No

Below the table, the route table details are shown. The "Subnet associations" tab is selected, displaying one explicit subnet association:

Explicit subnet associations (1)

Subnet ID	IPv6 CIDR
subnet-04af160d1d0aee071 / NWKT-Pub-euw2c	-

Below this, the "Subnets without explicit associations (1)" section is shown, listing one subnet:

Subnets without explicit associations (1)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Subnet ID	IPv6 CIDR
subnet-045deac93e40bf218 / NWKT-Pri-euw2a	AWS VPC CH01- F32-19-09-2021

Figure 1-32: Subnet Route Table.

The screenshot shows the AWS VPC Route Tables page with four route tables listed:

Name	Route table ID	Explicit subnet associations	Edge associations
DFLT-RTBL	rtb-8edeeae6	-	-
NWKT-MAIN-RT	rtb-069ac98ac692271fe	-	-
NWKT-PRI-RT	rtb-0e7261b40b0d5237b	subnet-045deac93e40bf218 / NWKT-Pri-euw2a	-
NWKT-PUB-RT	rtb-0fd4639034844c3ea	subnet-04af160d1d0aee071 / NWKT-Pub-euw2c	-

AWS VPC CH01- F33-13-09-2021

Figure 1-33: Subnet Route Table.

Create VPC, Subnets, and RT - AWS CloudFormation

In this section, we create a Subnet with the set of properties and attach it to VPC. We also specify a Route Table, which we associate with the Subnet using association.

In our YAML template (figure 1-34), we have four AWS resources (logical name within parenthesis):

- 1) AWS::EC2::VPC (NwktVPC)
- 2) AWS::EC2::Subnet (NwktSubnet)
- 3) AWS::EC2::RouteTable (NwktPUB2RouteTable)
- 4) AWS::EC2::SubnetRouteTableAssociation (NwktRouteTableAssociation)

We are using a Ref function for defining the dependencies between AWS resources when the actual AWS resource Identifier is unknown. For example, the Ref function in AWS::EC2::Subnet resource [2] refers to the resource AWS::EC2::VPC's logical name NwktVPC (A). We have to use an intrinsic function because we don't know which VPC Identifier AWS generates to VPC. After creating the subnet, we specify the subnet-specific Route Table [3]. First, we need to bind it to VPC using the Ref function value NwktVPC (B). Next, we "glue" the Route Table to Subnet using RouteTableAssociation, where we use two Ref functions. The first one refers to Route Table (C), and the second to Subnet (D).

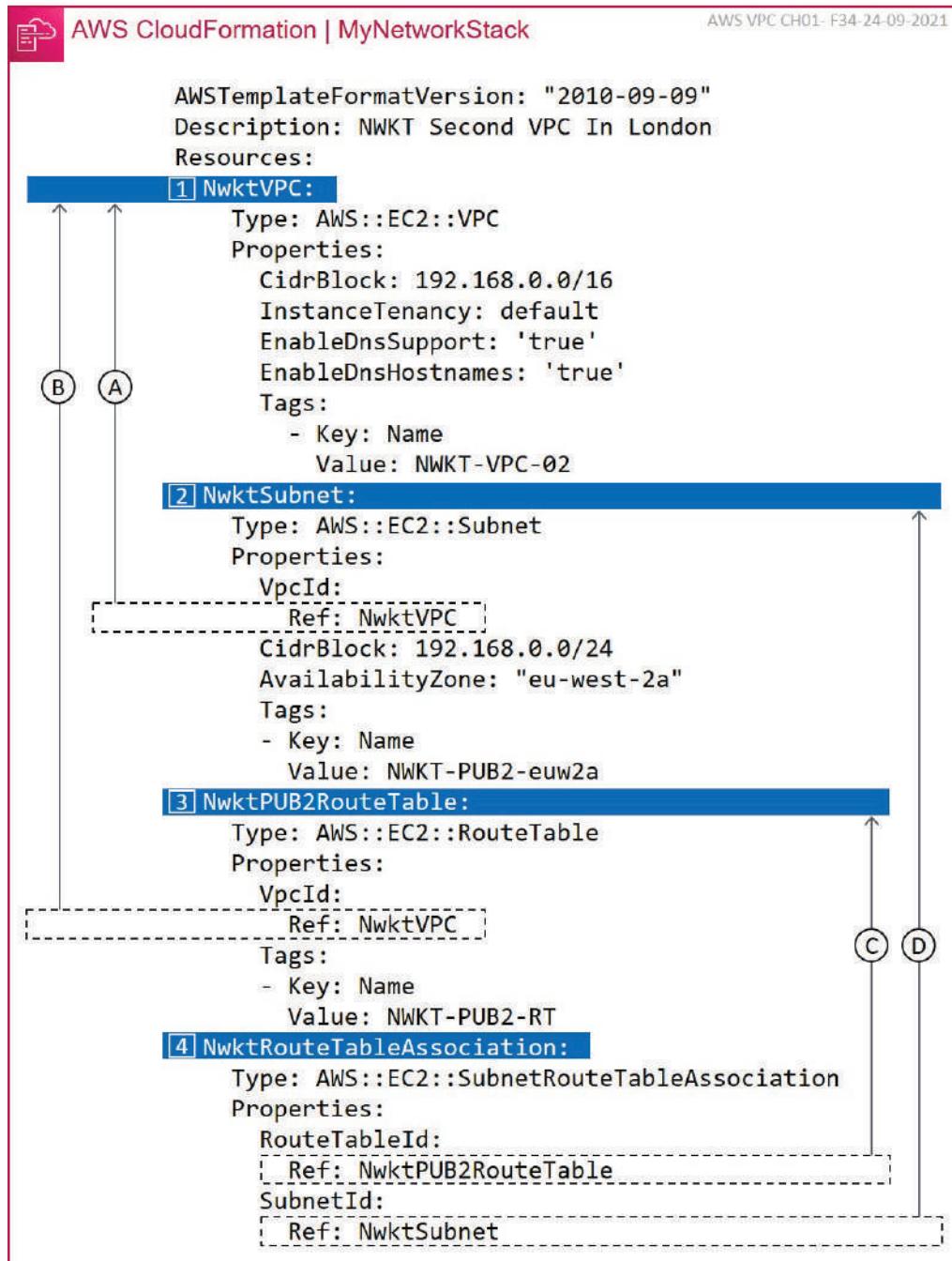


Figure 1-34: Subnet Route Table.

Example 1-10 shows how we upload a locally stored YAML template *MyVPC-Subnet-RT.yml* to AWS using the Stack Name *My NetworkStack*.

```
aws cloudformation create-stack --stack-name MyNetworkStack --template-body file://C:\Toni\AWS-CF-Templates\MyVPC-Subnet-RT.yml
{
  "StackId": "arn:aws:cloudformation:eu-west-2:123456654321:stack/MyNetworkStack/623ffce0-1d07-11ec-a276-02619afca8e"
}
```

Example 1-10: AWS CloudFormation – Create-Stack.

We can see our new stack by navigating to AWS CloudFormation/Stack window and by selecting *MyNetworkStack*. The resources tab shows all of the *MyNetworkStack* resources and their physical/logical Identifiers.

Resources (4)					
Logical ID	Physical ID	Type	Status		
NwktPUB2RouteTable	rtb-01552a97d0e687854	AWS::EC2::RouteTable	CREATE_COMPLETE		
NwktRouteTableAssociation	rtbassoc-0a5b18301c2eb2806	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE		
NwktSubnet	subnet-08eee681493045f37	AWS::EC2::Subnet	CREATE_COMPLETE		
NwktVPC	vpc-02c642ab4358b04d9	AWS::EC2::VPC	CREATE_COMPLETE		

Figure 1-35: Subnet Route Table.

By clicking the *subnet-08eee681493045f37* hyperlink, we can see its IPv4 CIDR address, Availability Zone, VPC, Route Table, and Network ACL. Note that the NACL is the default one.

Subnet ID	Subnet ARN	State	AWS VPC CH03 - F36-2-09-2021	IPv4 CIDR
subnet-08eee681493045f37	arn:aws:ec2:eu-west-2:017857243309:subnet/subnet-08eee681493045f37	Available		192.168.0.0/24
Available IPv4 addresses			Availability Zone	Availability Zone ID
251			eu-west-2a	euw2-az2
Network border group	-		Route table	Network ACL
eu-west-2	VPC		rtb-01552a97d0e687854 NWKT-PUB2-RT	acl-07951d33f7872b02f
Default subnet	vpc-02c642ab4358b04d9 NWKT-VPC-02		Auto-assign IPv6 address	Auto-assign customer-owned IPv4 address
No				

Figure 1-36: Subnet Route Table.

Create Network ACL

In this section, I am going to introduce the default Network ACL for subnets in VPC NVKT-VPC-01.

Figure 1-37 shows the complete structure of our VPC NVKT-VPC-01. We have a Public subnet 10.10.0.0/24 in AZ eu-west-2c a Private subnet 10.10.1.0/24 in AZ eu-west-2a. Both subnets are protected by the default VPC's NACL named NWKT-NACL. NACL allows all traffic to and from the subnet by default.

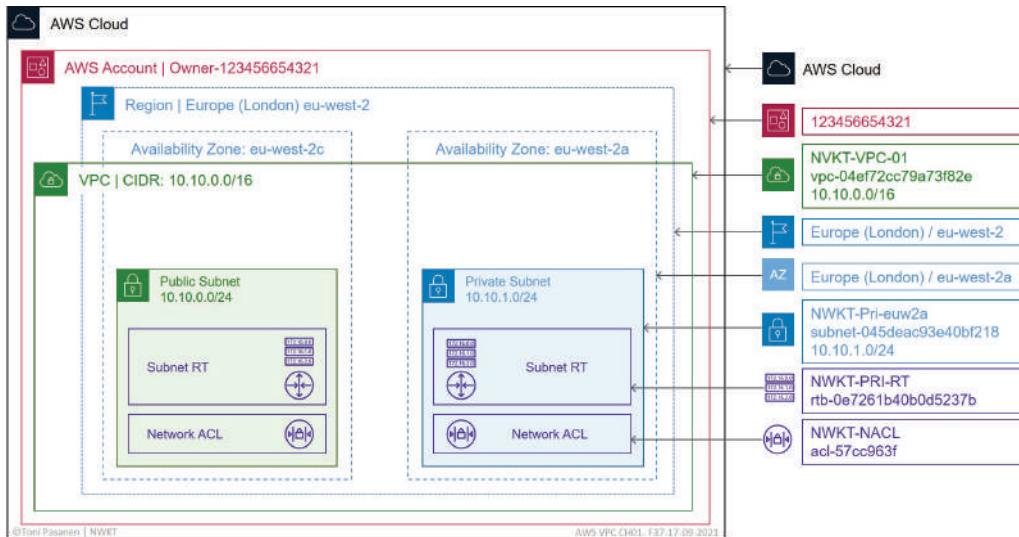


Figure 1-37: Complete VPC Stack.

Navigate back to VPC Dashboard and click the *Network ACL* hyperlink either from the left or right columns.

Inbound (figure 1-38) and outbound rules (figure 1-39) permit all traffic to and from the associated subnet. There is also an implicit deny policy rule at the end of the policy rule. The NACL is processed from the first row until there is a match.

Name	Network ACL ID	Associated Subnets	Default	VPC ID
DFLT-NACL	acl-57cc963f	3 Subnets	Yes	vpc-cfbac1a7 / DFLT-VPC
NWKT-NACL	acl-0dfc4c4ef28ae6491	2 Subnets	Yes	vpc-04ef72cc79a73f82e / NVKT-VPC-01

Inbound rules (2)						
Rule number	Type	Protocol	Port range	Source	Allow/Deny	
100	All traffic	All	All	0.0.0.0/0	Allow	
*	All traffic	All	All	0.0.0.0/0	Deny	

Figure 1-38: Default NACL: Inbound Rule.

Outbound rules (2)						
Rule number	Type	Protocol	Port range	Destination	Allow/Deny	
100	All traffic	All	All	0.0.0.0/0	Allow	
*	All traffic	All	All	0.0.0.0/0	Deny	

Figure 1-39: Default NACL: Outbound Rule.

Figure 1-40 shows the Subnet Association. Both our subnets are associated with NACL by default.

acl-0dfc4c4ef28ae6491 / NWKT-NACL					
Details	Inbound rules	Outbound rules	Subnet associations	Tags	AWS VPC CH01 - 140-19-09-2021
Subnet associations (2)					
Edit subnet associations					
<input type="text"/> Filter subnet associations					
Name	Subnet ID	Associated with	Availability Zone	IPv4 CIDR	
NWKT-Pri-euw2a	subnet-045deac93e40bf218	acl-0dfc4c4ef28ae6491 / NWKT-NACL	eu-west-2a	10.10.1.0/24	
NWKT-Pub-euw2c	subnet-04af160d1d0aee071	acl-0dfc4c4ef28ae6491 / NWKT-NACL	eu-west-2c	10.10.0.0/24	

Figure 1-40: Default NACL: Subnet Association.

At this phase, the VPC is a closed tenant without Internet GW, NAT GW, or any other route out from the VPC. However, both subnets have a local route 10.10.0.0/16 in their RT and the NACL that allows all traffic.

The next chapter introduces the Control-Plane (how EC2 instance reachability is registered to Mapping Service) and Data-Plane operation (the Data-Plane encapsulation) within VPC.

Chapter 2: VPC Control-Plane

VPC Control-Plane – Mapping Service

Introduction

This chapter explains the VPC Control-Plane operation when two EC2 instances within the same subnet initiate TCP session between themselves. In our example, EC2 instances are launched in two different physical servers. Both instances have an Elastic Network Interface (ENI) card. The left-hand side EC2's ENI has MAC/IP addresses `cafe:0001:0001/10.10.1.11` and the right-hand side EC2's ENI has MAC/IP addresses `beef:0001:0001/10.10.1.22`. Each physical server hosting EC2 instances has a Nitro Card for VPC [NC4VPC]. It is responsible for routing, data packets encapsulation/decapsulation, and Traffic limiting. In addition, Security Groups (SGs) are implemented in hardware on the Nitro card for VPC. AWS Control-Plane relies on the *Mapping Service* system decoupled from the network devices. It means that switches are unaware of Overlay Networks having no state information related to VPC's, Subnets, EC2 Instances, or any other Overlay Network components. From the Control-Plane perspective, physical network switches participate in the Underlay Network routing process by advertising the reachability information of physical hosts, Mapping Service, and so on. From the Data-Plane point of view, they forward packet based on the outer IP header.

Mapping Register

Starting an EC2 instance triggers the Control-Plane process on a host. Figure 2-1 illustrates that Host-1 and Host-2 store information of their local EC2 instances into the Mapping cache. Then they register these instances into Mapping Service. You can consider the registration process as a routing update. We need to inform the Mapping Service about the EC2 instance's a) MAC/IP addresses bind to ENI, b) Virtual Network Identifier (=VPC), c) the physical host IP, d) and the encapsulation mode (VPC tunnel header). If you are familiar with Locator/Id Separation Protocol LISP, you may notice that its Control-Plane process follows the same principles. The main difference is that switches in LISP-enabled networks have state information related to virtual/bare-metal servers running in a virtual network.

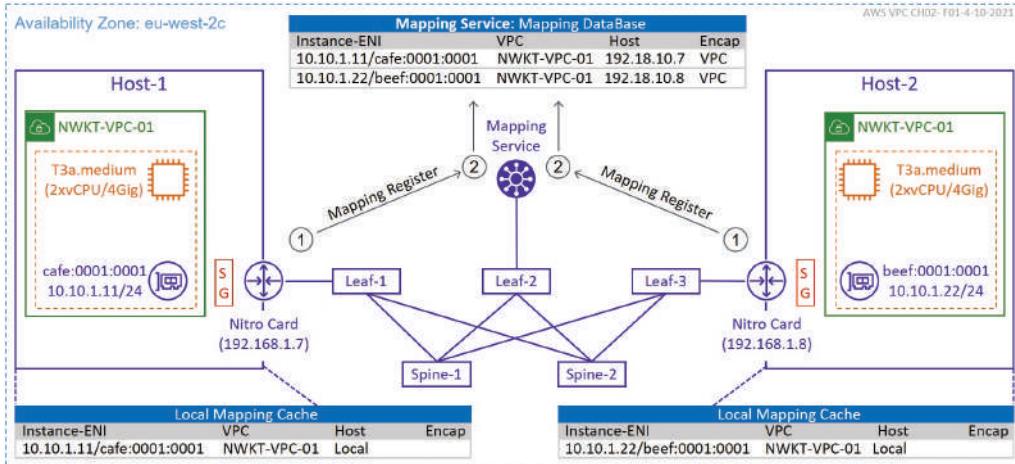


Figure 2-1: VPC Control-Plane Operation: Mapping Register.

Mapping Request - Reply

When the EC2 instance Cafe sends data to EC2 instance Beef for the first time, it sends an ARP Request to resolve the Mac address of IP 10.10.1.22 (1). This event starts the Control-Plane process. Host-1 first checks its local Mapping Cache. The result is miss, so it requests MAC/IP address binding and location information from the Mapping Service (2). Mapping Service processes the request (3) and replies (4) to Host-1. After receiving the Mapping Reply, Host-1 sends an ARP Reply message to the EC2 instance Beef. It also stores the information into the local Mapping Cache. This way, further data flows related to EC2 instance Beef can be processed without Mapping Request.

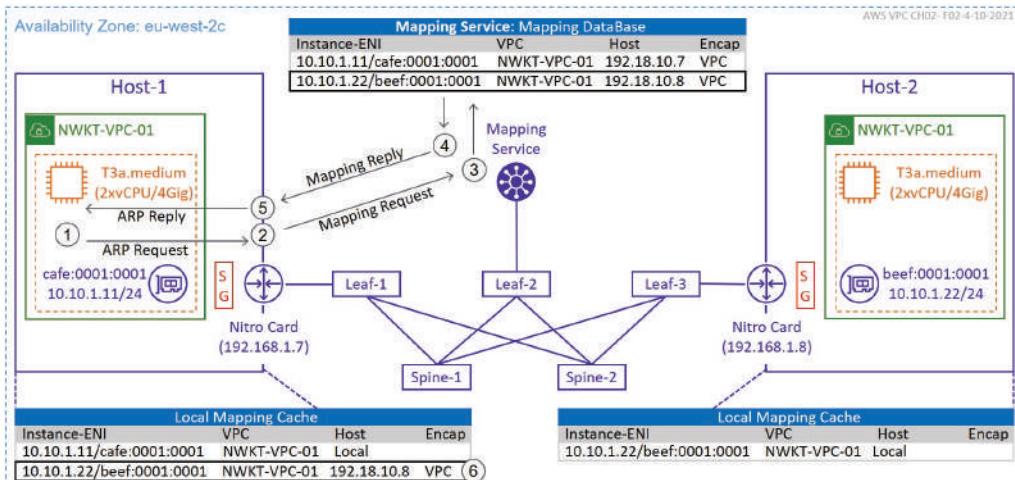


Figure 2-2: VPC Control-Plane Operation: Mapping Request.

Data-Plane Operation

After receiving the ARP-Reply message, EC2 Cafe starts sending data to EC2 Beef. It wraps the data within Ethernet and IP headers, then it forwards the packet out of the ENI through the Security Group (allow all) to Nitro Card. Based on the local Mapping cache, the Nitro card encapsulates the packet with a VPC header, Outer IP (dst IP 192.168.10.8), and Outer Ethernet header (dst MAC Leaf-1). Then it forwards the packet to the physical network. Leaf and Spine switches routes packet towards Host-2 without looking beyond the outer IP header.

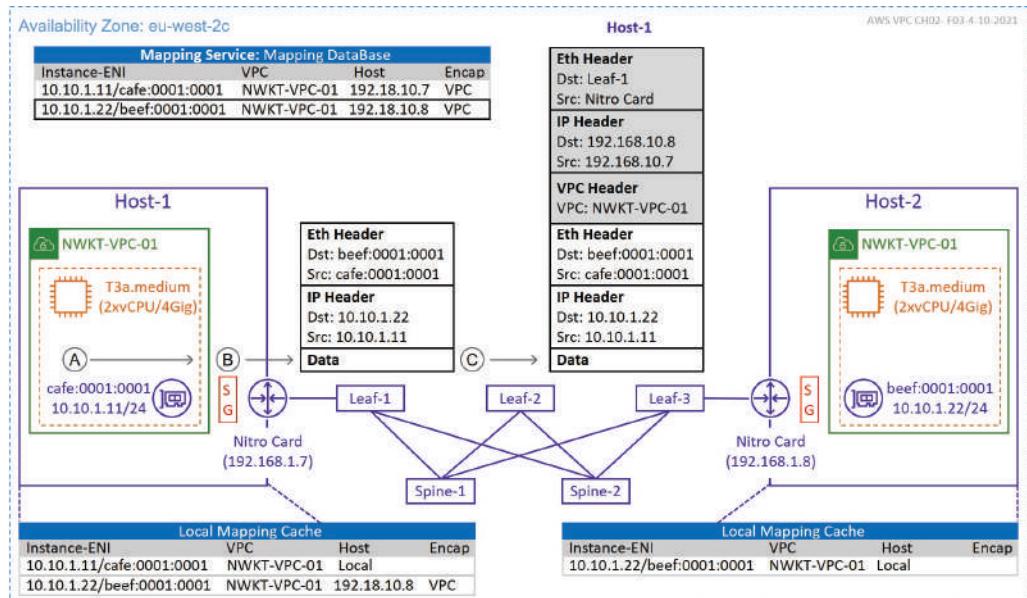


Figure 2-3: VPC Data-Plane Operation: Packet Forwarding.

When the Nitro Card for VPC in Host-2 receives the packet, it consults the Mapping Service for validating it. It checks that the source and the destination EC2 instances are in the same VPC. After a positive reply, it decapsulates the packet and forwards the original packet through the Security Group (SG) attached to EC2 instance Beef. SG is a stateful filter, and in addition to the permit/deny policy rule, it validates the TCP Sequence Numbers and Ack Numbers.

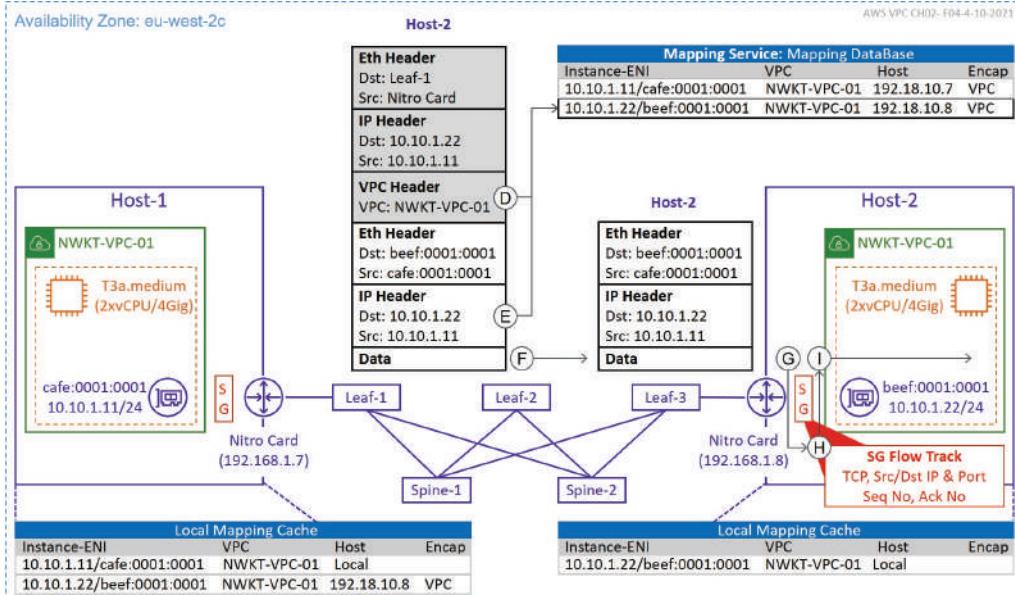


Figure 2-4: Default NACL: Subnet Association.

The data forwarding from Beef to Cafe follows the same process.

References

- [NC-VPC] James Hamilton, AWS Nitro System Blog post, <https://perspectives.mvdirona.com/2019/02/aws-nitro-system/>
- Ravi Murty, AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2). <https://www.youtube.com/watch?v=rUY-00yFlE4>
- Colm MacCárthaigh, AWS re:Invent 2017: Another Day, Another Billion Flows (NET405) <https://www.youtube.com/watch?v=8gc2DgBqo9U&t=1088s>

Chapter 3: VPC Internet Gateway

Introduction

This chapter explains what components/services and configurations we need to allow Internet traffic to and from an EC2 instance. VPCs themselves are closed entities. If we need an Internet connection, we need to use an AWS Internet Gateway (IGW) service. The IGW is running on a Blackfoot Edge Device in the AWS domain. It performs Data-Plane VPC encapsulation and decapsulation, as well as IP address translation. We also need public, Internet routable IP addresses. In our example, we allocate an AWS Elastic-IP (EIP) address. Then we associate it with EC2 Instance. By doing it, we don't add the EIP to the EC2 instance itself. Instead, we create a static one-to-one NAT entry into the VPC associated IGW. The subnet Route Table includes only a VPC's CIDR range local route. That is why we need to add a routing entry to the Subnet RT, default or more specific, towards IGW. Note that a subnet within an AWS VPC is not a Broadcast domain (VPC doesn't even support Broadcasts). Rather, we can think of it as a logical place for EC2 instances having uniform connection requirements, like reachability from the Internet. As a next step, we define the security policy. Each Subnet has a Network Access Control List (NACL), which is a stateless Data-Plane filter. The Stateless definition means that to allow bi-directional traffic flow, we have to permit flow-specific Request/Reply data separately. For simplicity, we are going to use the Subnet Default NACL. The Security Group (SG), in turn, is a stateful EC2 instance-specific Data-Plane filter. The Stateful means that filter permits flow-based ingress and egress traffic. Our example security policy is based on the SG. We will allow an SSH connection from the external host 91.152.204.245 to EC2 instance NWKT-EC-Front-End. In addition, we allow all ICMP traffic from the EC2 instance to the same external host. As the last part, this chapter introduces the Reachability Analyzer service, which we can use for troubleshooting connections. Figure 3-1 illustrates what we are going to build in this chapter.

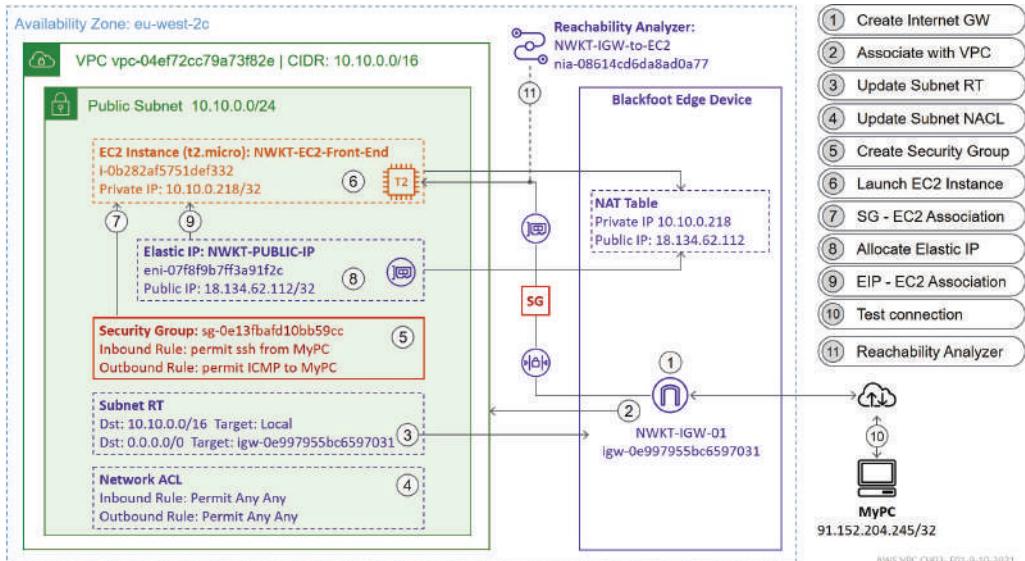


Figure 3-1: Setting Up an Internet Connection for Public Subnet of AWS VPC.

Allow Internet Access from Subnet

The first part of the AWS VPC Internet Access includes four steps. First, we create an Internet Gateway. Then we associate it with our VPC. Third, we add a route towards IGW into the Subnet Route Table. As the last step, we update the Subnet NACL if needed. We are using the Default NACL that permits all traffic. Figure 3-2 illustrates these four steps.

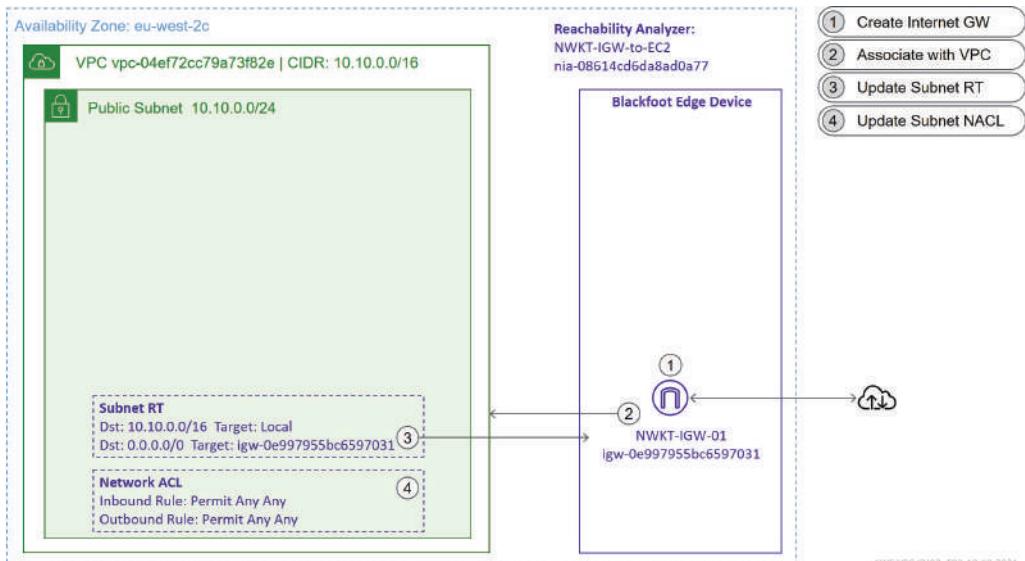


Figure 3-2: Basic Internet Access Building Blocks.

Create Internet Gateway

Navigate to the VPC Dashboard. Then select the *Internet Gateways* either from the Virtual Private Cloud drop-down menu or from the Resources by Region section.

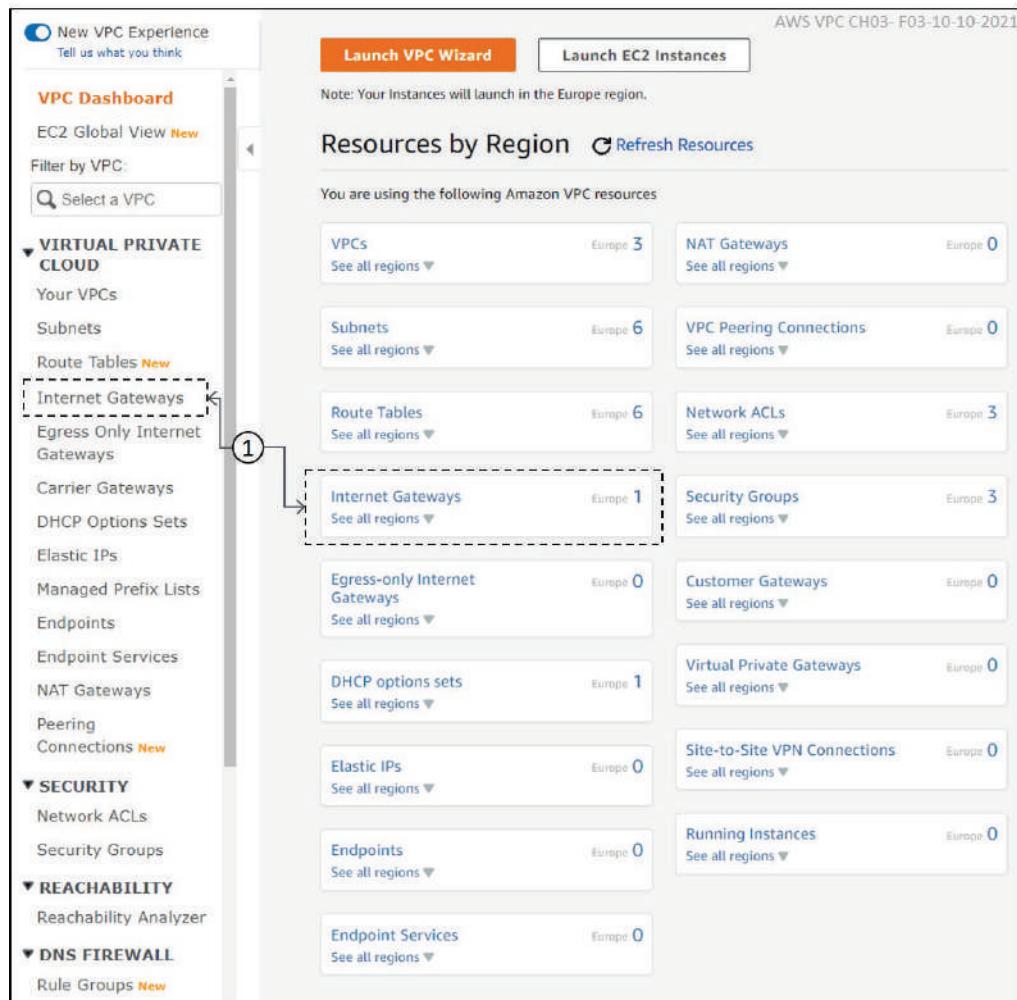


Figure 3-3: Create Internet Gateway, Step-1.

Figure 3-4 shows that we have a default Internet Gateway. Instead of using it, we create a new IGW. Click the *Create internet gateway* button.

Internet gateways (1/1) Info					Actions ▾	Create internet gateway
<input type="text"/> Filter internet gateways					< 1 >	
<input checked="" type="checkbox"/>	Name	Internet gateway ID	State	VPC ID	▼	
<input checked="" type="checkbox"/>	DFLT-INTG	igw-fb2ac893	Attached	vpc-cfbac1a7 DFLT-VPC		AWS VPC CH03- F04-10-10-2021

Figure 3-4: Create Internet Gateway, Step-2.

Fill the Name tag in the *Internet gateway settings* section. The name that you give will be auto-filled as a Key/Value pair in *Tag – optional* section. To proceed, click the *Create internet gateway* button.

VPC > Internet gateways > Create internet gateway AWS VPC CH03- F05-10-10-2021

Create internet gateway [Info](#)

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

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

Tags - optional

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.

Key	Value - optional	Remove
<input type="text" value="Name"/>	<input type="text" value="NWKT-IGW-01"/>	Remove
Add new tag		

You can add 49 more tags.

[Cancel](#) [Create internet gateway](#)

Figure 3-5: Create Internet Gateway, Step-3.

Attach the IGW with your VPC by selecting *Attach to a VPC* from the *Actions* drop-down menu or by clicking the *Attach to a VPC* button in the notification field.

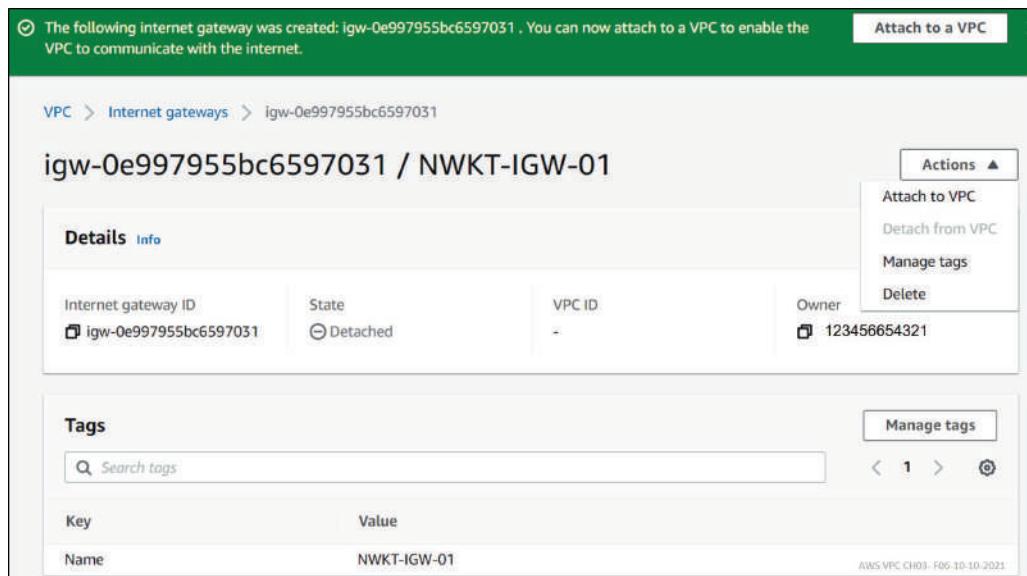


Figure 3-6: Attach an Internet GW with a VPC, Step-1.

Select our VPC NVKT-VPC-01 from the drop-down menu of the *Available VPCs* section.

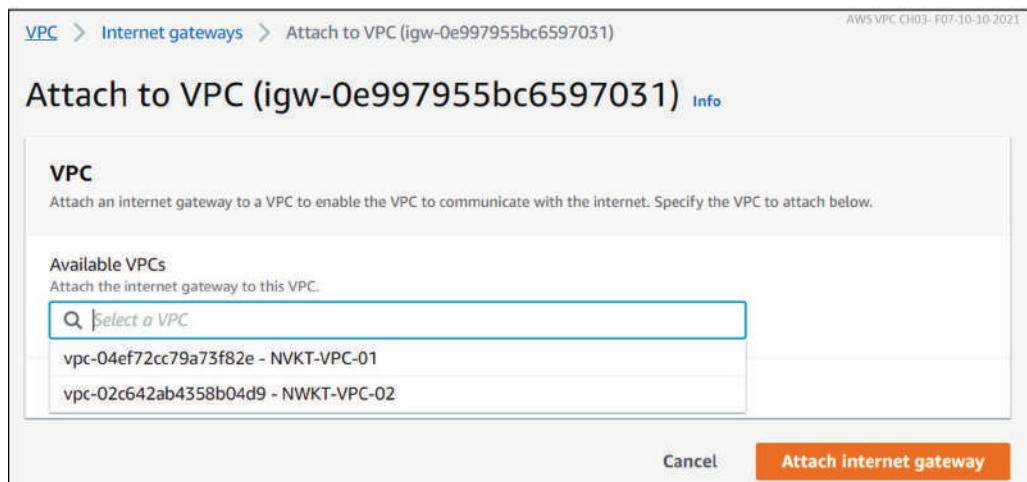


Figure 3-7: Attach an Internet GW with a VPC, Step-2.

After selecting the VPC, the AWS Command Line Interface command section appears. You can choose either Linux or Windows CLI commands to create an IGW-VPC association. Save the configuration by clicking the *Attach internet gateway* button.

The screenshot shows the 'Attach to VPC' configuration page for an Internet Gateway (igw-0e997955bc6597031). The top navigation bar includes 'VPC > Internet gateways > Attach to VPC (igw-0e997955bc6597031)'. The date 'AWS VPC CH03- F08-10-10-2021' is also visible. The main section is titled 'VPC' with a sub-instruction: 'Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.' A search bar contains the VPC ID 'vpc-04ef72cc79a73f82e'. Below this is a section for 'Available VPCs' with a note to 'Attach the internet gateway to this VPC.' A dropdown menu is set to 'Windows command prompt'. A 'CLI command' section displays the AWS CLI command: 'aws ec2 attach-internet-gateway --vpc-id \"vpc-04ef72cc79a73f82e\" --internet-gateway-id \"igw-0e997955bc6597031\" --region eu-west-2'. A 'Copy' button is available for this command. At the bottom are 'Cancel' and 'Attach internet gateway' buttons, with the latter being orange.

Figure 3-8: Attach an Internet GW with a VPC, Step-3.

Figure 3-8 shows that you have successfully created an Internet Gateway NWKT-IGW-01 (igw-0e997955bc6597031) and attached into the VPC NVKT-VPC-01 (vpc-04ef72cc79a73f82e).

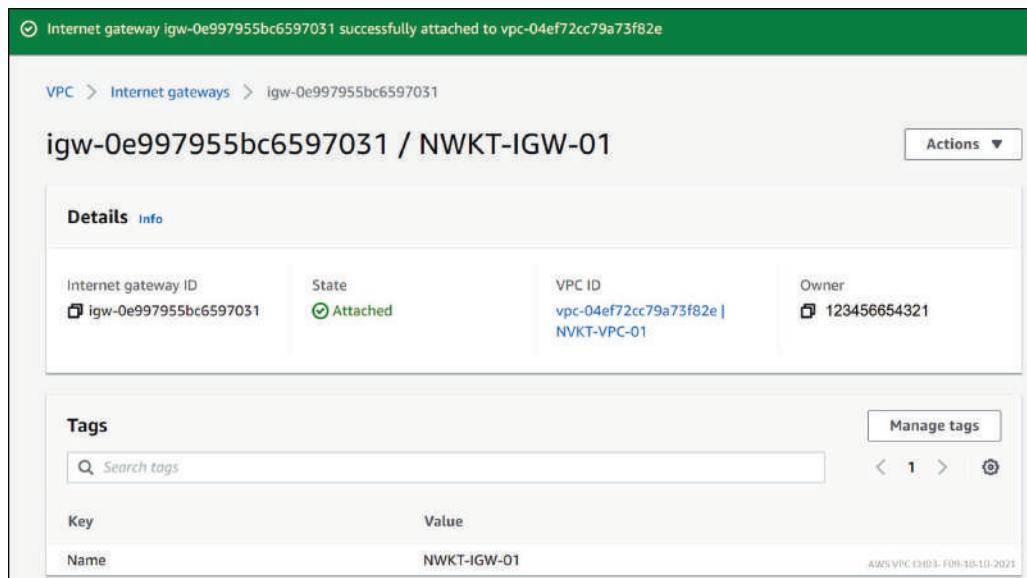


Figure 3-9: Attach an Internet GW with a VPC, Step-4.

Now the new IGW is listed in the Internet gateway window.

Internet gateways (1/2) Info					Actions	Create internet gateway		
<input type="text"/> Filter internet gateways					«	1	»	⚙️
Name	Internet gateway ID	State	VPC ID					
<input checked="" type="checkbox"/> NWKT-IGW-01	igw-0e997955bc6597031	Attached	vpc-04ef72cc79a73f82e NVKT-VPC-01					
<input type="checkbox"/> DFLT-INTG	igw-fb2ac893	Attached	vpc-cfbac1a7 DFLT-VPC					

Figure 3-10: Internet Gateway Configuration Verification.

Example 3-1 shows how we can verify the Internet Gateway configuration from the CLI in JSON format.

```
aws ec2 describe-internet-gateways --filters Name=tag:Name,Values=NWKT-IGW-01
{
    "InternetGateways": [
        {
            "Attachments": [
                {
                    "State": "available",
                    "VpcId": "vpc-04ef72cc79a73f82e"
                }
            ],
            "InternetGatewayId": "igw-0e997955bc6597031",
            "OwnerId": "123456654321",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "NWKT-IGW-01"
                }
            ]
        }
    ]
}
```

Example 3-1: Internet Gateway Configuration Verification – JSON Format.

Update Subnet Route Table

The next thing to do is to add route towards the Internet Gateway into the Subnet Route Table. Navigate back to the VPC dashboard and select Route Tables. We are going to update the Route Table NWKT-PUB-RT that is associated with the subnet 10.10.0.0/24 (NWKT-Pub-euw2c | subnet-04af160d1d0aee071).

Name	Route table ID	Explicit subnet associations	Main	VPC
NWKT-PRI-RT	rtb-0e7261b40b0d523...	-	No	vpc-04ef72cc79a73f82e NVKT-VPC-01
DFLT-RTBL	rtb-8edeeae6	-	Yes	vpc-cfbac1a7 DFLT-VPC
NWKT-PUB2-RT	rtb-01552a97d0e6878...	subnet-08eee681493045f37 / NWKT-PUB2-e...	No	vpc-02c642ab4358b04d9 NWKT-VP...
-	rtb-0d1a7e36c4412df0f	-	Yes	vpc-02c642ab4358b04d9 NWKT-VP...
NWKT-MAIN-RT	rtb-069ac98ac692271fe	-	Yes	vpc-04ef72cc79a73f82e NVKT-VPC-01
NWKT-PUB-RT	rtb-0fd4639034844c3ea	subnet-04af160d1d0aee071 / NWKT-Pub-eu...	No	vpc-04ef72cc79a73f82e NVKT-VPC-01

Figure 3-11: Subnet Route Tables.

Figure 3-12 shows that there is only a VPC CIDR range local route in the Route Table.

rtb-0fd4639034844c3ea / NWKT-PUB-RT				Actions ▾									
					AWS VPC CH03- F12-10-10-2021								
Details		Info											
Route table ID	rtb-0fd4639034844c3ea	Main	Explicit subnet associations	Edge associations									
VPC	vpc-04ef72cc79a73f82e NVKT-VPC-01	No	subnet-04af160d1d0aee071 / NWKT-Pub-euw2c	-									
Owner ID	123456654321												
<hr/>													
Routes Subnet associations Edge associations Route propagation Tags													
Routes (1) <div style="display: flex; justify-content: space-between;"> Edit routes Filter routes Both 1 < > ⚙️ </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Destination</th> <th>Target</th> <th>Status</th> <th>Propagated</th> </tr> </thead> <tbody> <tr> <td>10.10.0.0/16</td> <td>local</td> <td>Active</td> <td>No</td> </tr> </tbody> </table>						Destination	Target	Status	Propagated	10.10.0.0/16	local	Active	No
Destination	Target	Status	Propagated										
10.10.0.0/16	local	Active	No										

Figure 3-12: Subnet Route Tables – New Route Towards IGW.

You can see the Subnet-specific information by clicking the Explicit subnet associations hyperlink in the Details window shown in the figure 3-12 above. Figure 3-13 shows that the Ipv4 CIRD block 10.10.0.0/24. You can navigate back to Route Table view by clicking the Route Table hyperlink. Click the Edit routes button to add a new route (figure 3-12).

subnet-04af160d1d0aee071 / NWKT-Pub-euw2c				Actions ▾
Details				
Subnet ID subnet-04af160d1d0aee071	Subnet ARN arn:aws:ec2:eu-west-2:017857243309:subnet/subnet-04af160d1d0aee071	State Available	IPv4 CIDR 10.10.0.0/24	
Available IPv4 addresses 251	IPv6 CIDR -	Availability Zone eu-west-2c	Availability Zone ID euw2-az1	
Network border group eu-west-2	VPC vpc-04ef72cc79a73f82e NVKT-VPC-01	Route table rtb-0fd4639034844c3ea NWKT-PUB-RT	Network ACL acl-0dfc4c4ef28ae6491 NWKT-NACL	
Default subnet: No	Auto-assign IPv6 address: No	Auto-assign IPv6 address: No	Auto-assign customer-owned IPv4 address: No	
Customer-owned IPv4 pool: -	Auto-assign public IPv4 address: No	IPv4 CIDR reservations: -	IPv6 CIDR reservations: -	
Owner 123456654321	Outpost ID -			

AWS VPC CH03 - F13-10-10-2021

Figure 3-13: Subnet Route Tables – New Route Towards IGW, Subnet Verification.

Add a default route 0.0.0.0/0 to the Destination field. Then select our Internet Gateway from the Target drop-down menu and click the Save changes button. You can also preview your changes by clicking the Preview button

Edit routes					AWS VPC CH03 - F14-10-10-2021
Destination	Target	Status	Propagated		
10.10.0.0/16	local	Active	No		
0.0.0.0/0	igw-0e997955bc6597031	-	No	Remove	
<input type="button" value="Add route"/>					
				<input type="button" value="Cancel"/>	<input type="button" value="Preview"/> <input type="button" value="Save changes"/>

Figure 3-14: Subnet Route Tables – New Route Towards IGW.

Edit routes							AWS VPC CH03 - F15-10-10-2021
Edit routes - preview (1)							
Action	Destination	Target	Status	Propagated			
Will create	0.0.0.0/0	igw-0e997955bc6597031	-	No			
					<input type="button" value="Cancel"/>	<input type="button" value="Back to edit"/>	<input type="button" value="Save changes"/>

Figure 3-15: Subnet Route Tables – New Route Towards IGW - Preview.

After saving the changes the new route is shown in Route Table.

The screenshot shows the AWS Route Table configuration page. At the top, there are tabs for 'Details' and 'Info'. Below this, the Route table ID is rtb-0fd4639034844c3ea, and it is associated with VPC vpc-04ef72cc79a73f82e | NVKT-VPC-01. The table has one entry: a Main route with no explicit subnet associations, pointing to the igw-0e997955bc6597031 internet gateway. The status is 'Active' and propagation is 'No'.

Destination	Target	Status	Propagated
10.10.0.0/16	local	Active	No
0.0.0.0/0	igw-0e997955bc6597031	Active	No

Figure 3-16: Subnet Route Tables – New Route Towards IGW - Completed.

Network Access Control List

We are using our default Network ACL NWKT-NACL (acl-0dfc4c4ef28ae6491) in this example. It permits all Inbound (figure 3-17) and Outbound (figure 3-18) traffic by default and that's fine for us because we will later use Security Groups for defining the access policy.

The screenshot shows the AWS Network Access Control List (NACL) configuration page. The NACL is named acl-0dfc4c4ef28ae6491 / NWKT-NACL. It has two inbound rules: rule 100 allows all traffic from all sources (0.0.0.0/0) to all destinations (All) using all protocols and ports. Rule * denies all traffic from all sources (0.0.0.0/0) to all destinations (All) using all protocols and ports.

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Figure 3-17: Network Access Control List – Inbound Rules.

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	<input checked="" type="checkbox"/> Allow
*	All traffic	All	All	0.0.0.0/0	<input checked="" type="checkbox"/> Deny

Figure 3-18: Network Access Control List – Outbound Rules.

The Subnet association tab shows that this NACL is attached to both subnets 10.10.0.0/24 and 10.10.1.0/24.

Name	Subnet ID	Associated with	Availability Zone	IPv4 CIDR
NWKT-Pri-euw2a	subnet-045deac93e40bf218	acl-0dfc4c4ef28ae6491 / NWKT-NACL	eu-west-2a	10.10.1.0/24
NWKT-Pub-euw2c	subnet-04af160d1d0aee071	acl-0dfc4c4ef28ae6491 / NWKT-NACL	eu-west-2c	10.10.0.0/24

Figure 3-19: Network Access Control List – Subnet Association.

The example below shows how you can check the NACL Inbound and Outbound rules by using the AWS CLI.

```
aws ec2 describe-network-acls --query "NetworkAcls[2].Entries" --output table
-----
|                               DescribeNetworkAcls                               |
+-----+-----+-----+-----+
| CidrBlock | Egress | Protocol | RuleAction | RuleNumber |
+-----+-----+-----+-----+
| 0.0.0.0/0 | True  | -1     | allow     | 100      |
| 0.0.0.0/0 | True  | -1     | deny     | 32767    |
| 0.0.0.0/0 | False | -1     | allow     | 100      |
+-----+-----+-----+-----+
```

Example 3-2: Network Access Control List – Subnet Association.

Associate SG and Elastic-IP with EC2

In the previous section, we create an Internet Gateway for our VPC. We also add a static route towards IGW into the Route Table of Subnet 10.10.0.0/24. In this section, we first create a Security Group (SG). The SG allows SSH connection to the EC2 instance and ICMP from the EC2. Then we launch an EC2 and attach the previously configured SG to it. As the last step, we allocate an Elastic IP address (EIP) from the AWS IPv4 address pool and associate it with the EC instance. When we are done with all the previous steps, we will test the connection. First, we take ssh connection from MyPC to EC2. Then, we ping MyPC from the EC2. We also use AWS Reachability Analyzer to validate the path from IGE to EC2 instance. The last section introduces AWS billing related to this chapter.

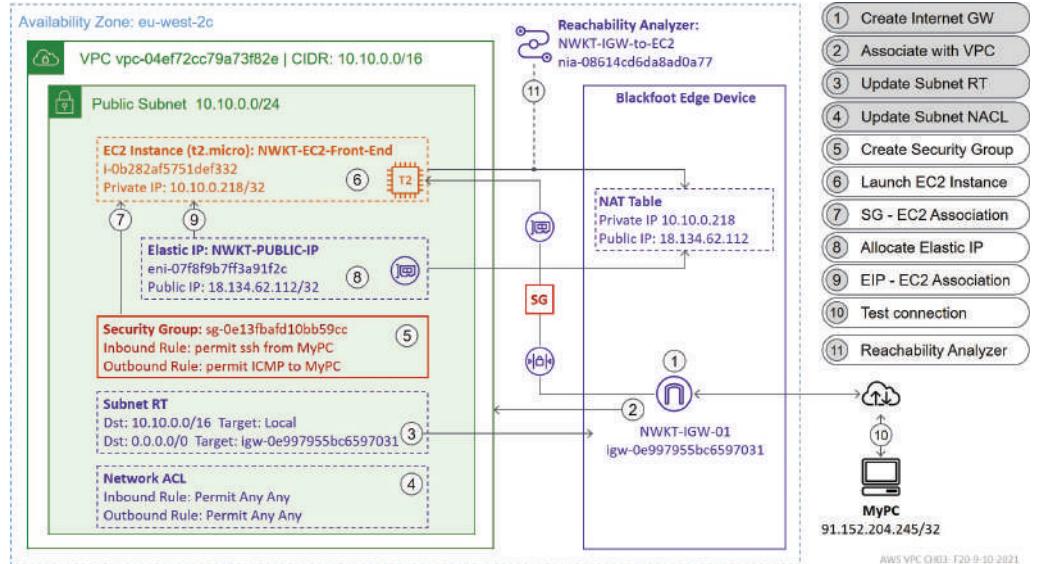


Figure 3-20: EC2 Instance, Elastic IP, and Security Group.

Create Security Group

Navigate to the VPC Dashboard. Then select Security Group either from the Security drop-down menu or Resource by Region section.

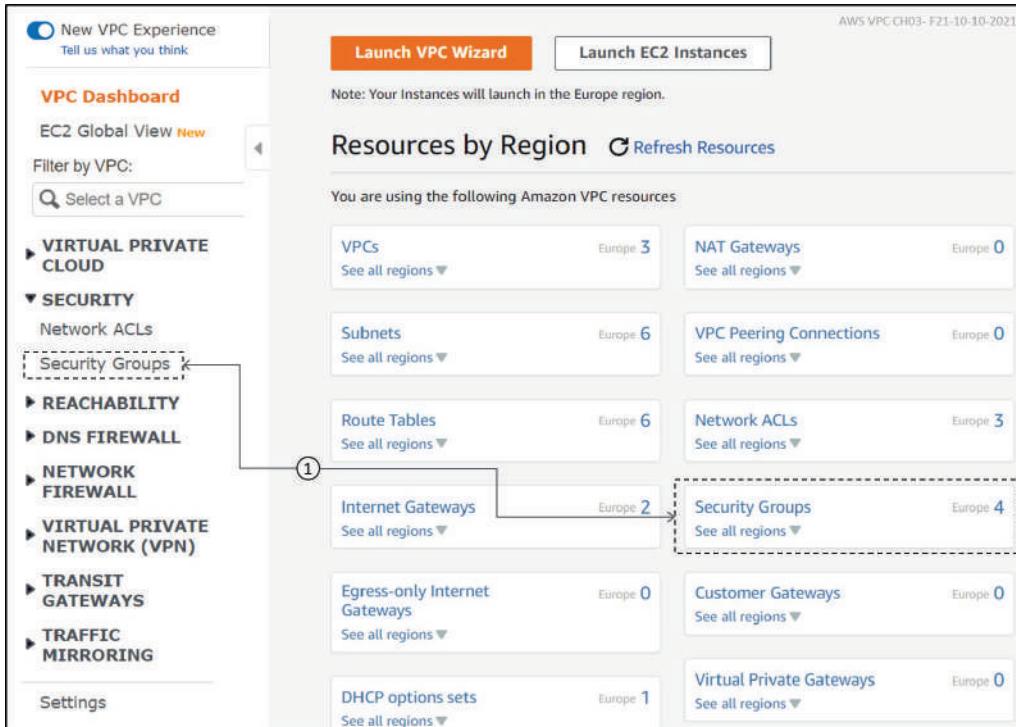


Figure 3-21: Create Security Group: Phase-1.

In the Security Group window, click the Create Security Group button. Note that the SG DFLT-SECG is the default SG for the default VPC. The two other SGs, in turn, are default SG for our VPCs.

AWS VPC CH03-F21-10-10-J021					
Security Groups (3) Info					
Actions Create security group					
<input type="checkbox"/> Filter security groups					
<input type="checkbox"/> Name ▼ Security group ID ▼ Security group name ▼ VPC ID ▼ Description ▼					
<input type="checkbox"/> - sg-07d9c05d97b2abbafe default vpc-02c642ab4358b04d9 default VPC security gr...					
<input type="checkbox"/> - sg-08ca8f649e6498109 default vpc-04ef72cc79a73f82e default VPC security gr...					
<input type="checkbox"/> DFLT-SECG sg-345cac57 default vpc-cfbac1a7 default VPC security gr...					

Figure 3-22: Create Security Group: Phase-2.

Fill in the Security Group Name, Description in the Basics details section in Create security group window. Then select our VPC from the VPC drop-down menu.

VPC > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name Info
NWKT-SGT-No-01
Name cannot be edited after creation.

Description Info
Permit SSH & Permit ICMP

VPC Info
vpc-04ef72cc79a73f82e

AWS VPC CH03-F23-10-10-2021

Figure 3-23: Create Security Group: Phase-3.

Create an Inbound rule that permits SSH from the external host. Then Create an Outbound rule that allows all ICMP traffic towards the external host.

Inbound rules Info

Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
SSH	TCP	22	Custom <input type="button" value="Q"/> 91.152.204.245/32 <input type="button" value="X"/>	SSH from Home PC

Add rule

Outbound rules Info

Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Destination <small>Info</small>	Description - optional <small>Info</small>
All ICMP - IPv4	ICMP	All	Custom <input type="button" value="Q"/> 91.152.204.245/32 <input type="button" value="X"/>	ICMP to Home PC

Add rule

Tags - optional
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.

No tags associated with the resource.

Add new tag
You can add up to 50 more tags.

Cancel

AWS VPC CH03-F23-10-10-2021

Figure 3-24: Create Security Group: Phase-4.

Figure 3-25 verifies that we have successfully created an SG sg-0e13fbaf10bb59cc.

Security group (sg-0e13fbaf10bb59cc | NWKT-SGT-No-01) was created successfully

Details

VPC > Security Groups > sg-0e13fbaf10bb59cc - NWKT-SGT-No-01

sg-0e13fbaf10bb59cc - NWKT-SGT-No-01

Details

Security group name NWKT-SGT-No-01	Security group ID sg-0e13fbaf10bb59cc	Description Permit SSH & Permit ICMP	VPC ID vpc-04ef72cc79a73f82e
Owner 123456654321	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules Outbound rules Tags

AWS VPC CH03-F25-10-10-2021

Inbound rules (1/1)

Name	Security group rule...	IP versi...	Type	Protocol	Port ra...	Source	Description
-	sgr-0ffb705a9f3cea065	IPv4	SSH	TCP	22	91.152.204.245/32	SSH from Hom...

Figure 3-25: Create Security Group: Phase-5.

When we create an SG, we named it NWKT-SGT-No-01. However, it is visible in the Security group name field while the name column is still empty.

Security Groups (1/4) Info

Filter security groups

Name	Security group ID	Security group name	VPC ID	Description	Inbound rules cou...
-	sg-07d9c05d97b2abba	default	vpc-02c642ab4358b04d9	default VPC security gr...	1 Permission entry
-	sg-08ca8f649e6498109	default	vpc-04ef72cc79a73f82e	default VPC security gr...	1 Permission entry
✓	-	NWKT-SGT-No-01	vpc-04ef72cc79a73f82e	Permit SSH & Permit I...	1 Permission entry
-	DFLT-SECG	default	vpc-cfbac1a7	default VPC security gr...	1 Permission entry

sg-0e13fbaf10bb59cc - NWKT-SGT-No-01

Details Inbound rules Outbound rules Tags

Tags

No tags associated with this resource

Manage tags

AWS VPC CH03-F26-10-10-2021

Figure 3-26: Create Security Group: Phase-6.

You can give a name by editing the Name field. You can also select the Tags tab and add a new tag with Key/Value pair Name/NWKT-SSH/ICMP-SGT. Click the Save changes button when done.

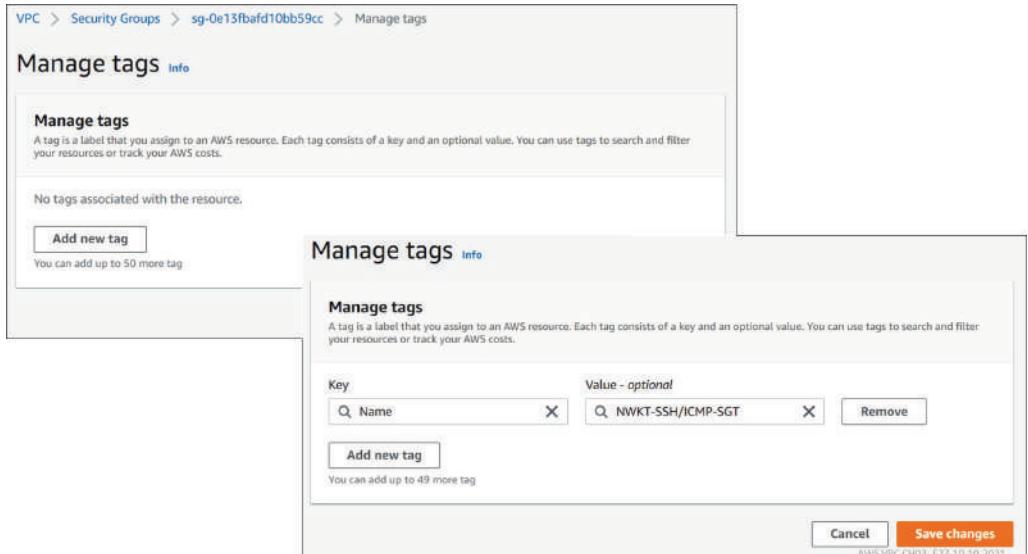


Figure 3-27: Create Security Group: Phase-7.

Figure 3-28 verifies our changes.

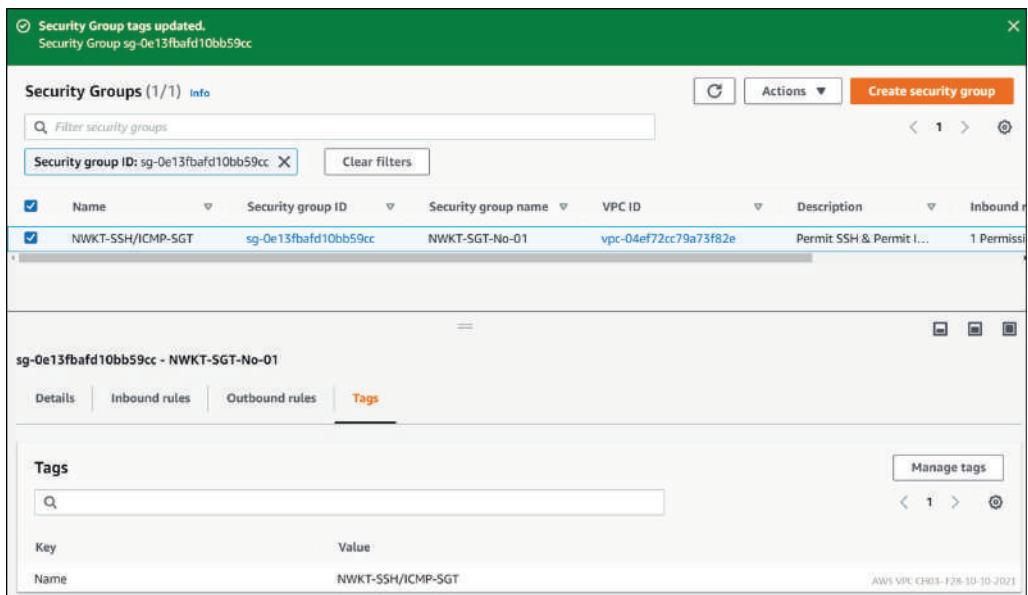


Figure 3-28: Create Security Group: Phase-8.

Example 3-3 shows the SG in table format on AWS CLI. Note that we are using Tags Key/Pair values. As a recap, if we don't use “--output table” option, the output format is JSON.

```
aws ec2 describe-security-groups --filters Name=tag:Name,Values=NWKT-SSH/ICMP-SGT --output table
```

DescribeSecurityGroups	
SecurityGroups	
Description	Permit SSH & Permit ICMP
GroupId	sg-0e13fbafdf10bb59cc
GroupName	NWKT-SGT-No-01
OwnerId	123456654321
VpcId	vpc-04ef72cc79a73f82e
IpPermissions	
FromPort	22
IpProtocol	tcp
ToPort	22
IpRanges	
CidrIp	Description
91.152.204.245/32	SSH from Home PC
141.192.166.178/32	SSH from Home PC-new
IpPermissionsEgress	
FromPort	-1
IpProtocol	icmp
ToPort	-1
IpRanges	
CidrIp	91.152.204.245/32
Description	ICMP to Home PC
Tags	
Key	Name
Value	NWKT-SSH/ICMP-SGT

Example 3-3: Security Groups – Table Output from AWS CLI.

Launch an EC2 Instance

Navigate to the EC2 Dashboard. Then select the Launch instance button.

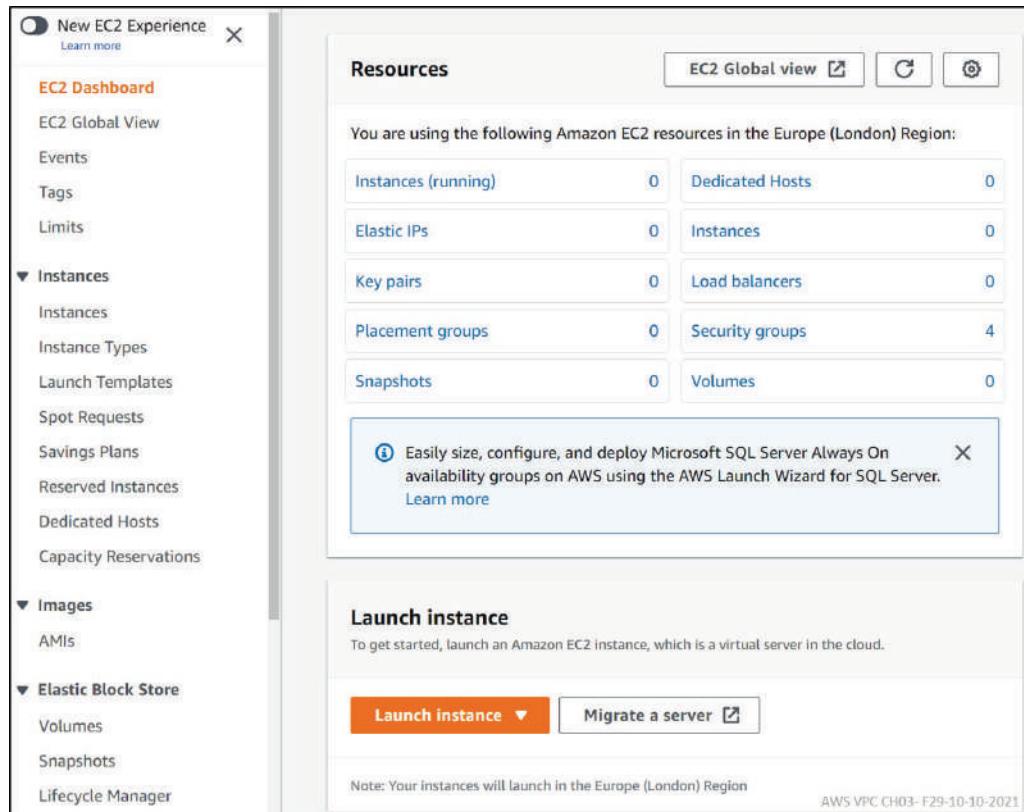


Figure 3-29: Launch EC2 Instance.

I'm using an Amazon Linux 2 (figure 3-30), which is one of the free tier Amazon Machine Images (AMI). Note that free tier doesn't always mean free. For example, the t2.micro instance type is free for one year only with a new AWS account. My account is more than 12 years old, so the t2.micro is not free.

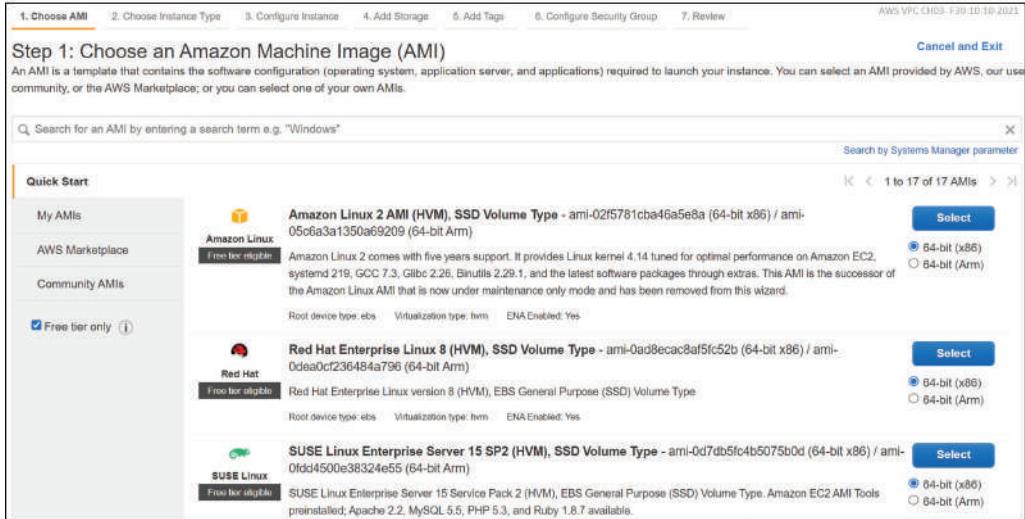


Figure 3-30: Launch EC2 Instance: Phase-1.

Select the t2.micro. Click the Next: Configure Instance Details button.

The screenshot shows the 'Step 2: Choose an Instance Type' page. At the top, there are tabs for '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '5. Add Tags', '6. Configure Security Group', and '7. Review'. A note at the top right says 'AWS VPC CH03-F31-10-10-2021'. Below the tabs, there are filters: 'Filter by: All instance families' (dropdown), 'Current generation' (dropdown), and 'Show/Hide Columns'. A note says 'Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only.)'. The main content area is a table showing various instance types:

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
<input type="checkbox"/>	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
<input type="checkbox"/>	t3	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes

At the bottom are buttons: 'Cancel', 'Previous', 'Review and Launch' (highlighted in blue), and 'Next: Configure Instance Details'.

Figure 3-31: Launch EC2 Instance: Phase-2.

Select the VPC and the Subnet where you want to launch the instance. Leave all other fields with their default options (figures 3-32, 3-33, and 3-34).

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances	<input type="text" value="1"/>	Launch into Auto Scaling Group <input type="checkbox"/>
Purchasing option	<input type="checkbox"/> Request Spot Instances	
Network	vpc-04ef72cc79a73f82e NVKT-VPC-01	<input type="button" value="Create new VPC"/>
Subnet	subnet-04af160d1d0aee071 NWKT-Pub-euw2c eu	<input type="button" value="Create new subnet"/> 251 IP Addresses available
Auto-assign Public IP	<input type="button" value="Use subnet setting (Disable)"/>	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	<input type="button" value="Open"/>	
Domain join directory	No directory	<input type="button" value="Create new directory"/>
IAM role	<input type="button" value="None"/>	

AWS VPC CH03-F32-10-10-2021

Figure 3-32: Launch EC2 Instance: Phase-3.

Step 3: Configure Instance Details

Shutdown behavior	<input type="button" value="Stop"/>
Stop - Hibernate behavior	<input type="checkbox"/> Enable hibernation as an additional stop behavior
Enable termination protection	<input type="checkbox"/> Protect against accidental termination
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring <small>Additional charges apply.</small>
Tenancy	<input type="button" value="Shared - Run a shared hardware instance"/> <small>Additional charges will apply for dedicated tenancy.</small>
Credit specification	<input type="checkbox"/> Unlimited <small>Additional charges may apply</small>
File systems	<input type="button" value="Add file system"/> <input type="button" value="Create new file system"/>

AWS VPC CH03-F33-10-10-2021

Figure 3-33: Launch EC2 Instance: Phase-3 continues.

We don't modify the Storage setting or add Tags. Click *Next* buttons until you end up on to *Configure Security Group* page.

The screenshot shows the 'Step 3: Configure Instance Details' page of the EC2 instance launch wizard. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance (highlighted in orange), 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review.

Network interfaces:

Device	Network Interface	Subnet	Primary IP	Secondary	IPv6 IPs
eth0	New network interface	subnet-04af160d	Auto-assign	Add IP	The selected subnet does not support IPv6 because it does not have an IPv6 CIDR.

Add Device button.

Advanced Details:

- Enclave: Enable
- Metadata accessible: Enabled
- Metadata version: V1 and V2 (token optional)
- Metadata token response hop limit: 1
- User data: As text As file Input is already base64 encoded
(Optional)

Buttons at the bottom: Cancel, Previous, Review and Launch (highlighted in blue), Next: Add Storage.

Figure 3-34: Launch EC2 Instance: Phase-3 continues.

Choose the *Select an existing security group* radio button. Then select the SG NWKT-SGT-No-01.

The screenshot shows the 'Step 6: Configure Security Group' page of the EC2 instance launch wizard. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group (highlighted in orange), and 7. Review.

Assign a security group:

- Create a new security group
- Select an existing security group

Security Groups:

Security Group ID	Name	Description	Actions
sg-08ca8f649e6498109	default	default VPC security group	Copy to new
sg-0e13fbaf10bb59cc	NWKT-SGT-No-01	Permit SSH & Permit ICMP	Copy to new

Inbound rules for sg-0e13fbaf10bb59cc (Selected security groups: sg-0e13fbaf10bb59cc):

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	91.152.204.245/32	SSH from Home PC

AWS VPC CH03-F34-10-10-2021

Figure 3-35: Launch EC2 Instance: Phase-4 Security Group Association.

Preview your instance setting before launching it. Define the Key pair name and download it before launching (figure 3-37). You use this keypair name when connecting to an instance.

Step 7: Review Instance Launch
Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-02f5781cba46a5e8a
Free tier eligible

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups

Security Group ID	Name	Description
sg-0e13fbaf10bb59cc	NWKT-SGT-No-01	Permit SSH & Permit ICMP

All selected security groups inbound rules:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	91.152.204.245/32	SSH from Home PC

Figure 3-36: Launch EC2 Instance: Phase-5 Review.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair type
 RSA ED25519

Key pair name

Download Key Pair

You have to download the **private key file (*.pem file)** before you can continue. **Store it in a secure and accessible location**. You will not be able to download the file again after it's created.

Cancel Launch Instances

Figure 3-37: Launch EC2 Instance: Phase-6 Key Pair.

The Launch status window shows the instance id (i-0b28af5751def332). It also shows the instructions on how you can connect to an instance.

Launch Status AWS VPC CH03- F38-10-10-2023

Your instances are now launching
The following instance launches have been initiated: [i-0b28af5751def332](#) [View launch log](#)

Get notified of estimated charges
[Create billing alerts](#) to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click [View Instances](#) to monitor your instances' status. Once your instances are in the **running** state, you can [connect](#) to them from the Instances screen. [Find out](#) how to connect to your instances.

▼ Here are some helpful resources to get you started

• How to connect to your Linux Instance	• Amazon EC2: User Guide
• Learn about AWS Free Usage Tier	• Amazon EC2: Discussion Forum

While your instances are launching you can also

- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)
- [Manage security groups](#)

[View Instances](#)

Figure 3-38: *Launch EC2 Instance: Phase-7 Launch Status.*

Figure 3-39 shows the information related to our EC2 instance.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 P
i-0b282af5751def332	t2.micro	eu-west-2c	running	Initializing	None			
Instance: i-0b282af5751def332 Private IP: 10.10.0.218								
Description	Status Checks	Monitoring	Tags					
Instance ID	i-0b282af5751def332			Public DNS (IPv4)	-			
Instance state	running	(i)		IPv4 Public IP	-			
Instance type	t2.micro			IPv6 IPs	-			
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more			Elastic IPs				
Private DNS	ip-10-10-0-218.eu-west-2.compute.internal			Availability zone	eu-west-2c			
Private IPs	10.10.0.218			Security groups	NWKT-SGT-No-01, view inbound rules, view outbound rules			
Secondary private IPs				Scheduled events	No scheduled events			
VPC ID	vpc-04ef72cc79a73f82e (NVKT-VPC-01)			AMI ID	amzn2.ami-hvm-2.0.20211001.1-x86_64-gp2 (ami-0215781cba46a5e8a)			
Platform	Amazon Linux			Subnet ID	subnet-04af160d1d00ee071 (NWKT-Pub-euw2c)			
Platform details	Linux/UNIX			Network interfaces	eth0			
Usage operation	RunInstances			!AM role	-			
Source/dest. check	True			Key pair name	NWKT-KEY-EC2:			
T2/T3 Unlimited	Disabled (i)			Owner	017857243309			
EBS-optimized	False			Launch time	October 6, 2021 at 6:37:19 PM UTC+3 (less than one hour) (i)			
Root device type	ebs			Termination protection	False			
Root device	/dev/xvda			Lifecycle	normal			
Block devices	/dev/xvda			Monitoring	basic			
Capacity Reservation	-			Alarm status	None			
Capacity Reservation Settings	Open			Kernel ID	-			
				RAM disk ID	-			
				Nitro Enclaves	Disabled	AWS VPC CH03-F39-10-10-2021		

Figure 3-39: Launch EC2 Instance –Running Instance Description.

Allocate Elastic IP address from Amazon Ipv4 Pool

Navigate to the EC2 dashboard and select an Elastic IP option from the Security drop-down menu. Select the eu-west-2 Network Border Group. Click the *Amazon's pool of Ipv4 address* option. Then click the *Allocate* button.

Figure 3-41 shows that we have successfully allocated a public Ipv4 address 18.134.62.112. The allocation id is eipalloc-01c7d2a4877a061a7. Next, select the *Associate Elastic IP address* from the Action drop-down menu.

EC2 > Elastic IP addresses > Allocate Elastic IP address AWS VPC CH03- F40-10-10-2021

Allocate Elastic IP address Info

Elastic IP address settings Info

Network Border Group Info

eu-west-2 X

Public IPv4 address pool

- Amazon's pool of IPv4 addresses
- Public IPv4 address that you bring to your AWS account (option disabled because no pools found) [Learn more](#)
- Customer owned pool of IPv4 addresses (option disabled because no customer owned pools found) [Learn more](#)

Global static IP addresses

AWS Global Accelerator can provide global static IP addresses that are announced worldwide using anycast from AWS edge locations. This can help improve the availability and latency for your user traffic by using the Amazon global network. [Learn more](#)

[Create accelerator](#)

Tags - optional

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.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tag

[Cancel](#) Allocate

Figure 3-40: Elastic IP Allocation.

⌚ Elastic IP address allocated successfully.
Elastic IP address 18.134.62.112 Associate this Elastic IP address >

Elastic IP addresses (1/1)						Actions ▾	Allocate Elastic IP address
						View details	Private IP address
						Release Elastic IP addresses	-
<input checked="" type="checkbox"/> Filter Elastic IP addresses Public IPv4 address: 18.134.62.112 X Clear filters						Associate Elastic IP address	Disassociate Elastic IP address
<input checked="" type="checkbox"/> Name Allocated IPv4 add... Type Allocation ID						Update reverse DNS	-
<input checked="" type="checkbox"/> NWKT-PUBLIC-IP 18.134.62.112 Public IP eipalloc-01c7d2a4877a061a7							

AWS VPC CH03- F41-10-10-2021

Figure 3-41: Elastic IP Allocation.

Select the Instance as a resource type. Select our EC2 instance from the Instance drop-down menu. The Private IP 10.10.10.218 is automatically allocated to the instance by AWS when we created it. This IP is auto-filled on the Private IP address field. You can also allow EIP reassociation in this window. As the last step, click the Associate button.

The screenshot shows the 'Associate Elastic IP address' dialog box. At the top, there's a breadcrumb navigation: EC2 > Elastic IP addresses > Associate Elastic IP address. On the right, a timestamp is displayed: AWS VPC CH03- F42-10-10-2021. The main title is 'Associate Elastic IP address'. Below it, a sub-instruction says 'Choose the instance or network interface to associate to this Elastic IP address (18.134.62.112)'. A bolded section 'Elastic IP address: 18.134.62.112' is followed by a note: 'Resource type: Choose the type of resource with which to associate the Elastic IP address.' Two radio buttons are shown: 'Instance' (selected) and 'Network interface'. A warning message in a box states: '⚠ If you associate an Elastic IP address to an instance that already has an Elastic IP address associated, this previously associated Elastic IP address will be disassociated but still allocated to your account. [Learn more](#)'.

Resource type
Choose the type of resource with which to associate the Elastic IP address.
 Instance
 Network interface

Elastic IP address: 18.134.62.112

⚠ If you associate an Elastic IP address to an instance that already has an Elastic IP address associated, this previously associated Elastic IP address will be disassociated but still allocated to your account. [Learn more](#)

Instance

Private IP address
The private IP address with which to associate the Elastic IP address.

Reassociation
Specify whether the Elastic IP address can be reassigned to a different resource if it's already associated with a resource.
 Allow this Elastic IP address to be reassigned

Figure 3-42: Elastic IP Allocation – EIP to Instance Association.

After association, you will be notified about the successful association (figure 3-43).

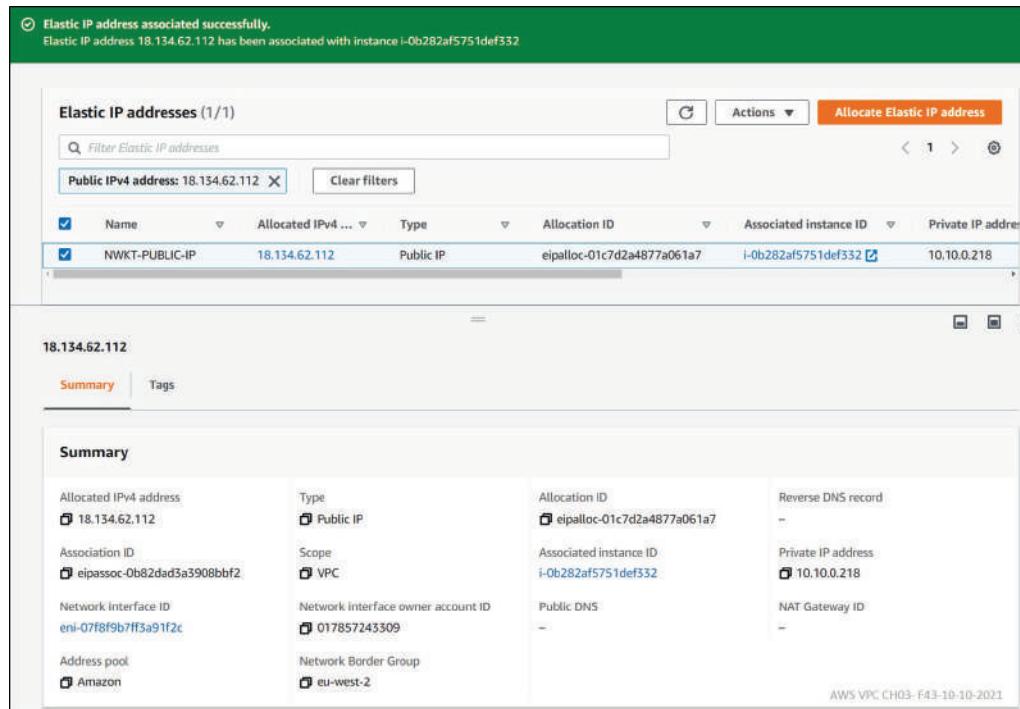


Figure 3-43: Elastic IP Allocation – EIP to Instance Association.

Note that the Public DNS (Ipv4) field is empty. That is because when we created the VPC, we didn't enable DNS hostname resolution.

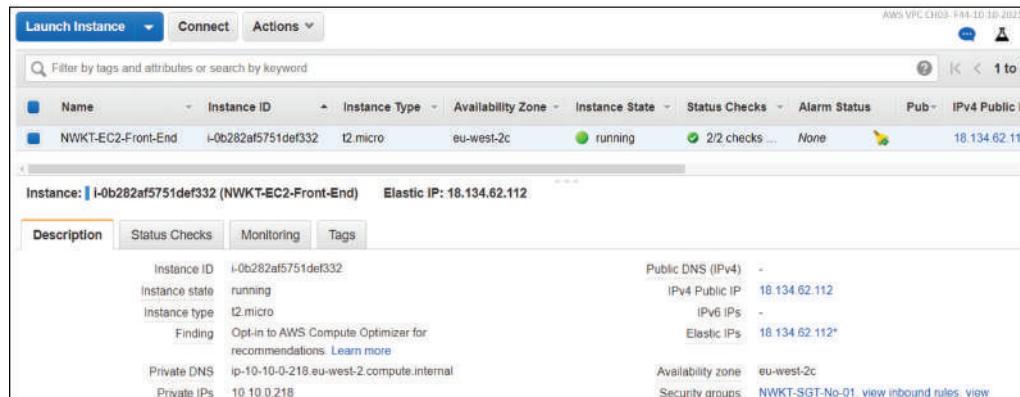


Figure 3-44: Elastic IP Allocation – Public DNS.

In order to assign DNS name to instance's Public IP address, we need to change the VPC DNS setting. Go to the VPC Dashboard and select the VPC. Select the *Edit DNS hostnames* and click the *Save changes* button. Note that the VPC window in figure 3-40 is taken after the DNS Resolution change. That is why it shows that that DNS Hostname resolution is enabled.

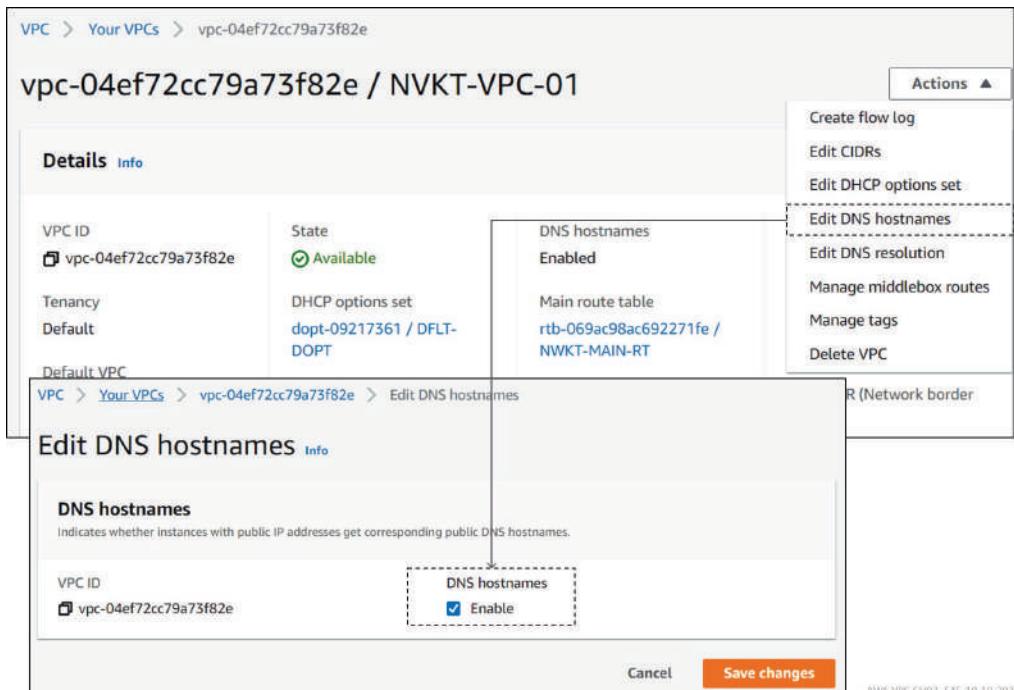


Figure 3-45: Elastic IP Allocation – EIP to Instance Association.

Figure 4-46 verifies that now we also have a DNS name for the Public IP address. To get instructions on how to connect to the selected instance, click the *Connect* button. Note that you have to restrict the access permissions of NWKT-EC2-KEY.pem.

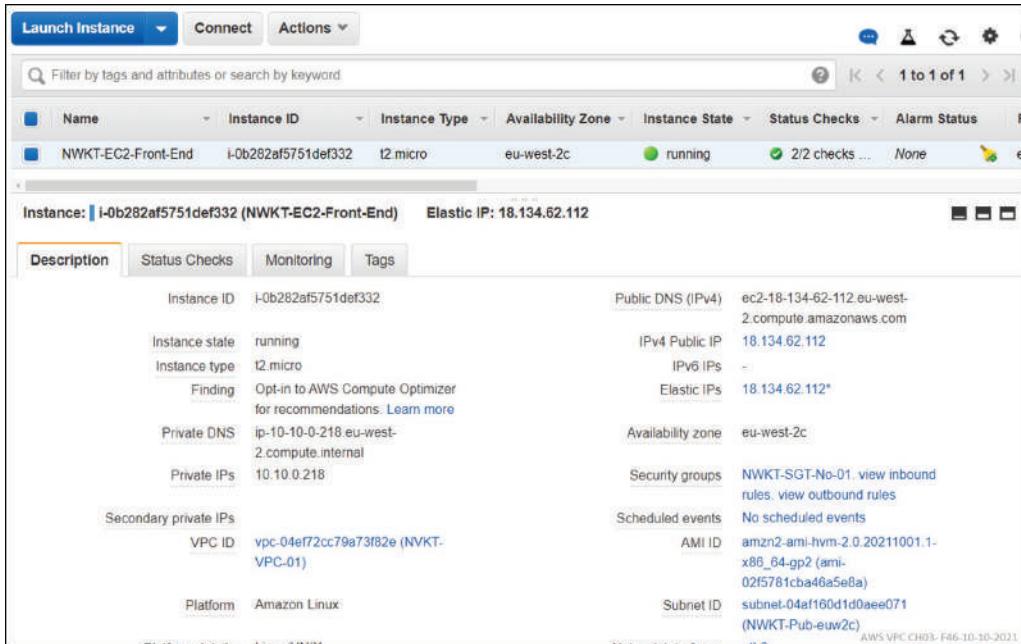


Figure 3-46: Elastic IP Allocation – EIP to Instance Association.

The guide in figure 3-47 shows how to do that on Linux. However, it does not tell how the process is done in Windows.

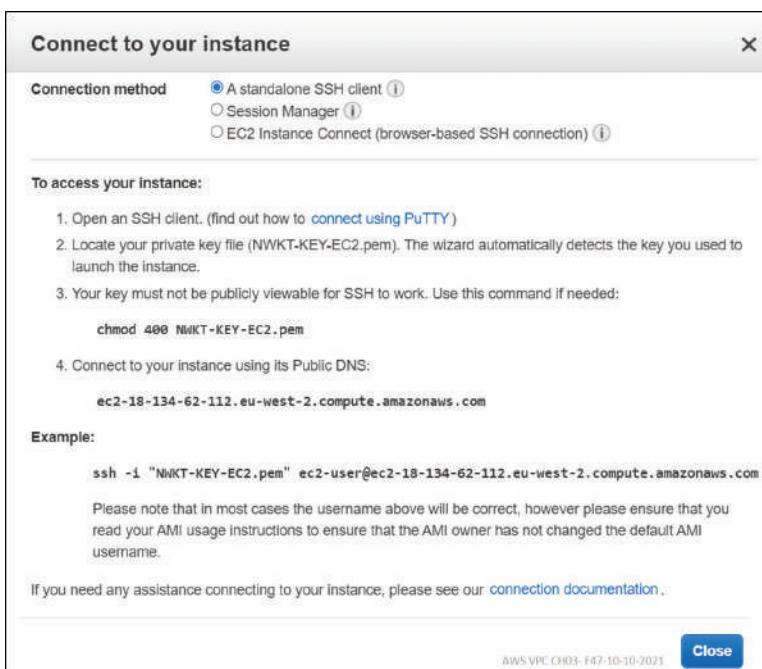


Figure 3-47: Elastic IP Allocation – EIP to Instance Association.

When you try to connect to EC2 instance when your pem file user access rights are left to their default settings your connection won't work. You may receive an error notification about Unprotected Private Key File (example 3-4).

```
C:\folder-1\folder-2\AWS\Chapter-3>ssh -i "NWKT-KEY-EC2.pem" ec2-user@ec2-18-134-62-112.eu-west-2.compute.amazonaws.com
@@@@@@@WARNING: UNPROTECTED PRIVATE KEY FILE! @
Permissions for 'NWKT-KEY-EC2.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "NWKT-KEY-EC2.pem": bad permissions
ec2-user@ec2-18-134-62-112.eu-west-2.compute.amazonaws.com: Permission denied
(publickey,gssapi-keyex,gssapi-with-mic).
```

Example 3-4: SSH Connection Failure Due to too Open File Permissions.

In order to change access rights, go to the properties of an NWKT-KEY-EC2.pem and select the Security tab. Figure 3-48 shows that four groups/users have full access to file by default. Click the Advanced button. Next, click the Disable inheritance button.

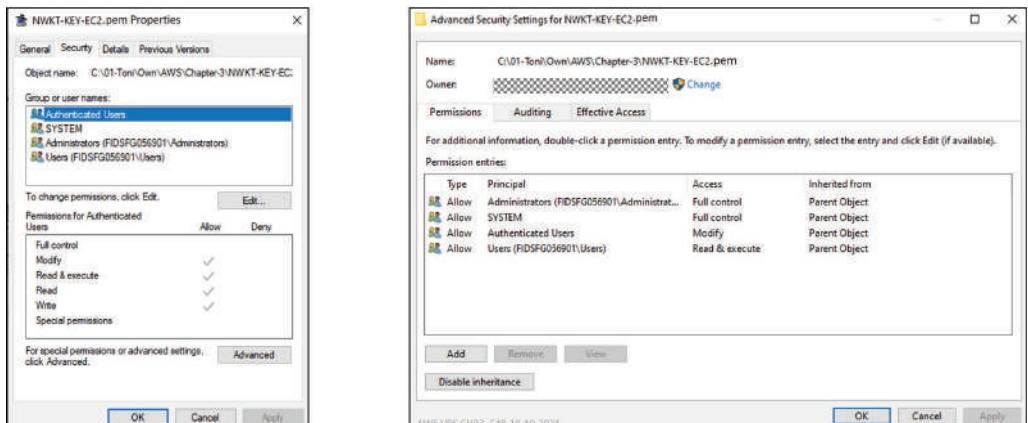


Figure 3-48: Changing pem File User Access Policy in Windows – Step-1.

Remove all default users from the permission entry. Then add your account to it with full file permission.

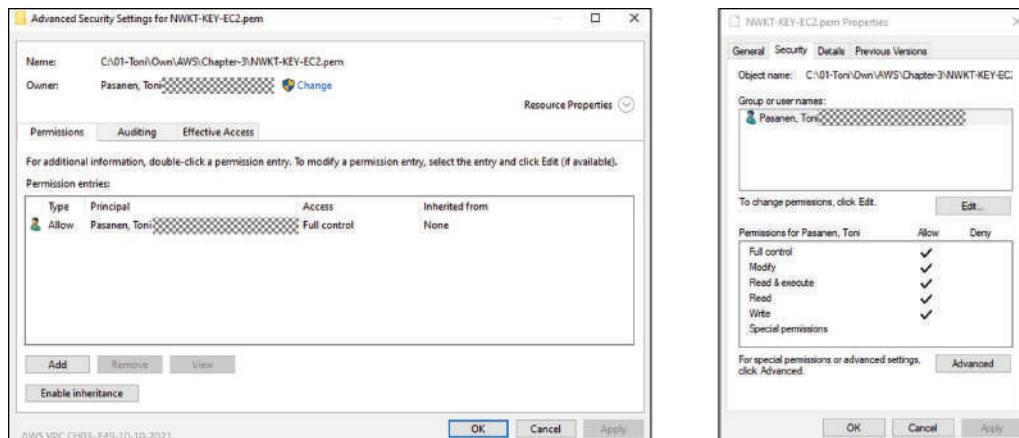


Figure 3-49: Changing pem File User Access Policy in Windows – Step-2.

After the change, you can connect to your instance.

```
C:\folder-1\folder-2\AWS\Chapter-3>ssh -i "NWKT-KEY-EC2.pem" ec2-user@ec2-18-134-62-112.eu-west-2.compute.amazonaws.com
[ec2-user@ip-10-10-0-218 ~]$
```

Example 3-5: Successful SSH Connection.

The example below verifies that we also have a connection from the EC2 Instance to the external host 91.152.204.245.

```
[ec2-user@ip-10-10-0-218 ~]$ ping 91.152.204.245
PING 91.152.204.245 (91.152.204.245) 56(84) bytes of data.
64 bytes from 91.152.204.245: icmp_seq=1 ttl=102 time=0.885 ms
64 bytes from 91.152.204.245: icmp_seq=2 ttl=102 time=0.844 ms
64 bytes from 91.152.204.245: icmp_seq=3 ttl=102 time=0.913 ms

64 bytes from 91.152.204.245: icmp_seq=5 ttl=102 time=0.902 ms
^C
--- 91.152.204.245 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4053ms
rtt min/avg/max/mdev = 0.844/0.904/0.976/0.042 ms
[ec2-user@ip-10-10-0-218 ~]$
```

Example 3-6: Ping from the EC2 Instance to External Host.

Example 3-7 below shows the complete Instance information.

```
C:\folder-1\folder-2\AWS\Chapter-3>aws ec2 describe-instances --output table
```

DescribeInstances	
Reservations	
OwnerId	123456654321
ReservationId	r-0edc1be860412bb63
Instances	
AmiLaunchIndex	0
Architecture	x86_64
ClientToken	
EbsOptimized	False
EnaSupport	True
Hypervisor	xen
ImageId	ami-02f5781cba46a5e8a
InstanceId	i-0b282af5751def332
InstanceType	t2.micro
KeyName	NWKT-KEY-EC2
LaunchTime	2021-10-08T15:37:19.000Z
PrivateDnsName	ip-10-10-0-218.eu-west-2.compute.internal
PrivateIpAddress	10.10.0.218
PublicDnsName	ec2-18-134-62-112.eu-west-2.compute.amazonaws.com
PublicIpAddress	18.134.62.112
RootDeviceName	/dev/xvda
RootDeviceType	ebs
SourceDestCheck	True
StateTransitionReason	
SubnetId	subnet-04af160d1d0aee071
VirtualizationType	hvm
VpcId	vpc-04ef72cc79a73f82e
BlockDeviceMappings	
DeviceName	/dev/xvda
Ebs	
AttachTime	2021-10-08T15:37:20.000Z
DeleteOnTermination	True
Status	attached
VolumeId	vol-0a6d0b7dccab72820
CapacityReservationSpecification	
CapacityReservationPreference	open
CpuOptions	
CoreCount	1
ThreadsPerCore	1
HibernationOptions	
Configured	False
Monitoring	

State	disabled
NetworkInterfaces	
Description	Primary network interface
InterfaceType	interface
MacAddress	02:cb:a0:af:b7:a0
NetworkInterfaceId	eni-07f8f9b7ff3a91f2c
OwnerId	017857243309
PrivateDnsName	ip-10-10-0-218.eu-west-2.compute.internal
PrivateIpAddress	10.10.0.218
SourceDestCheck	True
Status	in-use
SubnetId	subnet-04af160d1d0aee071
VpcId	vpc-04ef72cc79a73f82e
Association	
IpOwnerId	123456654321
PublicDnsName	ec2-18-134-62-112.eu-west-2.compute.amazonaws.com
PublicIp	18.134.62.112
Attachment	
AttachTime	2021-10-08T15:37:19.000Z
AttachmentId	eni-attach-06e4d5fc3698f08f6
DeleteOnTermination	True
DeviceIndex	0
Status	attached
Groups	
GroupId	sg-0e13fbafdf10bb59cc
GroupName	NWKT-SGT-No-01
PrivateIpAddresses	
Primary	True
PrivateDnsName	ip-10-10-0-218.eu-west-2.compute.internal
PrivateIpAddress	10.10.0.218
Association	
IpOwnerId	123456654321
PublicDnsName	ec2-18-134-62-112.eu-west-2.compute.amazonaws.com
PublicIp	18.134.62.112
Placement	
AvailabilityZone	eu-west-2c
GroupName	
Tenancy	default
SecurityGroups	
GroupId	sg-0e13fbafdf10bb59cc
GroupName	NWKT-SGT-No-01
State	
Code	16

Name	running
Tags	
Key	Name
Value	NWKT-EC2-Front-End

Example 3-7: Complete EC2 Instance Information.

Reachability Analyzer

AWS *Reachability Analyzer* is a tool for troubleshooting reachability problems. We can use it for monitoring the path status between two objects in our VPC. In this section, we verify the path status from the Internet Gateway Endpoint to EC2 Instance NWKT-EC2-Front-End. You can find the Reachability Analyzer from the VPC Dashboard's Reachability drop-down menu. Click the *Create and analyze path* button.

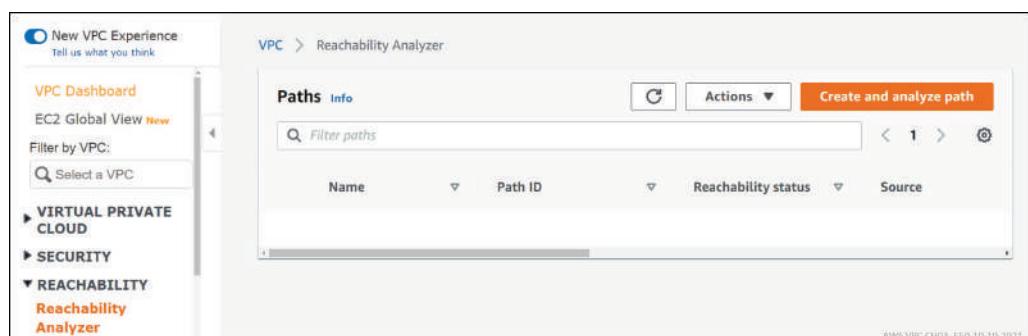


Figure 3-50: Reachability Analyzer.

Fill in the name Tag field. Then select the Internet Gateways as Source type. Select our IGW from the Source drop-down menu. The Destination type is Instance. Choose our EC2 instance from the Destination drop-down menu. The inbound security group associated with the EC2 instance allows only SSH connection. SSH use the destination port is 22, and the transport protocol is TCP. Click Create and analyze path button after filling all fields.

The screenshot shows the 'Create and analyze path' page in the AWS VPC Reachability Analyzer. At the top, there's a breadcrumb navigation: VPC > Reachability Analyzer > Create and analyze path. Below the navigation, the title 'Create and analyze path' has an 'Info' link. A descriptive text explains what a path is and how it's analyzed, mentioning charges for each analysis. A 'Learn more' link is also present.

Path configuration

Name tag - optional: Creates a tag with a key of 'Name' and a value that you specify. The input field contains 'NWKT-IGW-to-EC2'.

Source type	Source	Source IP address - optional
Internet Gateways	igw-0e997955bc6597031	192.0.2.1

Destination type	Destination	Destination IP address - optional
Instances	i-0b282af5751def332	192.0.2.1

Destination port - optional: The input field contains '22'. A note below says 'Number must be between 0 and 65535'.

Protocol: Use the appropriate protocol. The dropdown menu shows 'TCP'.

Add tags - optional: A note says '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.' There is one tag listed: Key 'Name' with Value 'NWKT-IGW-to-EC2'. Buttons for 'Add new tag' and 'Remove' are shown.

You can add up to 49 more tags.

At the bottom, there are 'Cancel' and 'Create and analyze path' buttons. The 'Create and analyze path' button is highlighted in orange.

Figure 3-51: Reachability Analyzer – Create and Analyze Path.

Figure 3-52 shows the progress of Reachability status. The Reachability status field turns to Reachable, and the State field shows Succeeded.

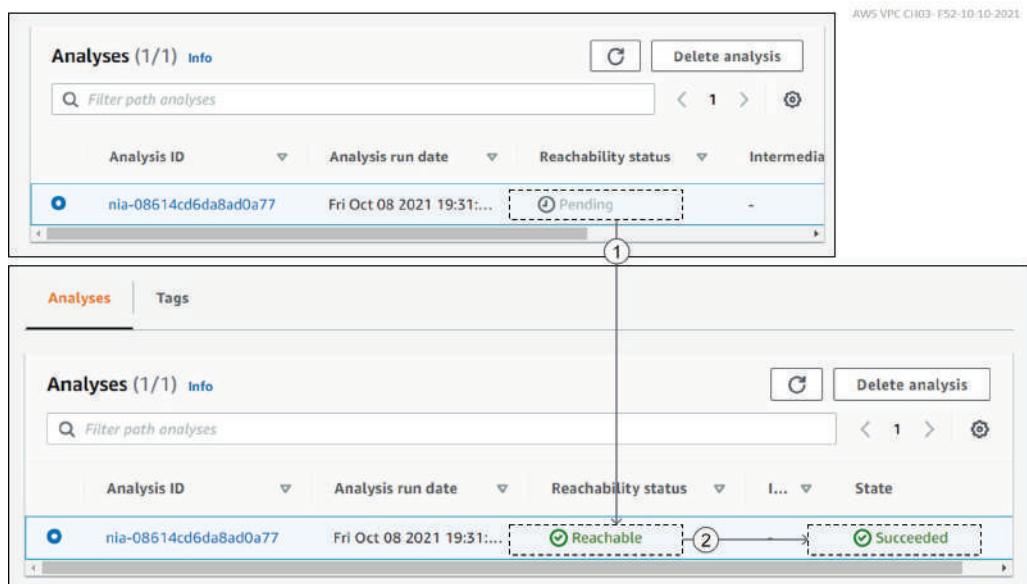


Figure 3-52: Reachability Analyzer – Create and Analyze Path.

Click the Analysis ID nia-08614cd6da8ad0a77 in order to see the visual path representation. Figure 3-53 shows that the path from the Internet GW to EC2 instance first goes through the NACL and then through the SGT.

VPC > Reachability Analyzer > nip-0a9f87a68ea39b676 > nia-08614cd6da8ad0a77

nia-08614cd6da8ad0a77

Summary		Manage tags		Delete analysis
Source igw-0e997955bc6597031	Destination i-0b282af5751def332	Reachability status Reachable	Analysis run date Fri Oct 08 2021 19:31:52 GMT+0300 (Eastern European Summer Time)	
Intermediate component filter -	Path ID nip-0a9f87a68ea39b676			

Analysis nia-08614cd6da8ad0a77 | Tags

Analysis explorer [Info](#)

View reverse path

```
graph TD; Source((igw-0e997955bc6597031)) --- act1((act-0dfc4c4ef28ae6491)); act1 --- sg1[sg-0e13fbaf10bb59cc]; sg1 --- eni1[eni-07f8f9b7ff3a91f2c]; eni1 --- Destination((i-0b282af5751def332))
```

AWS VPC CH03- F53-10-10-2021

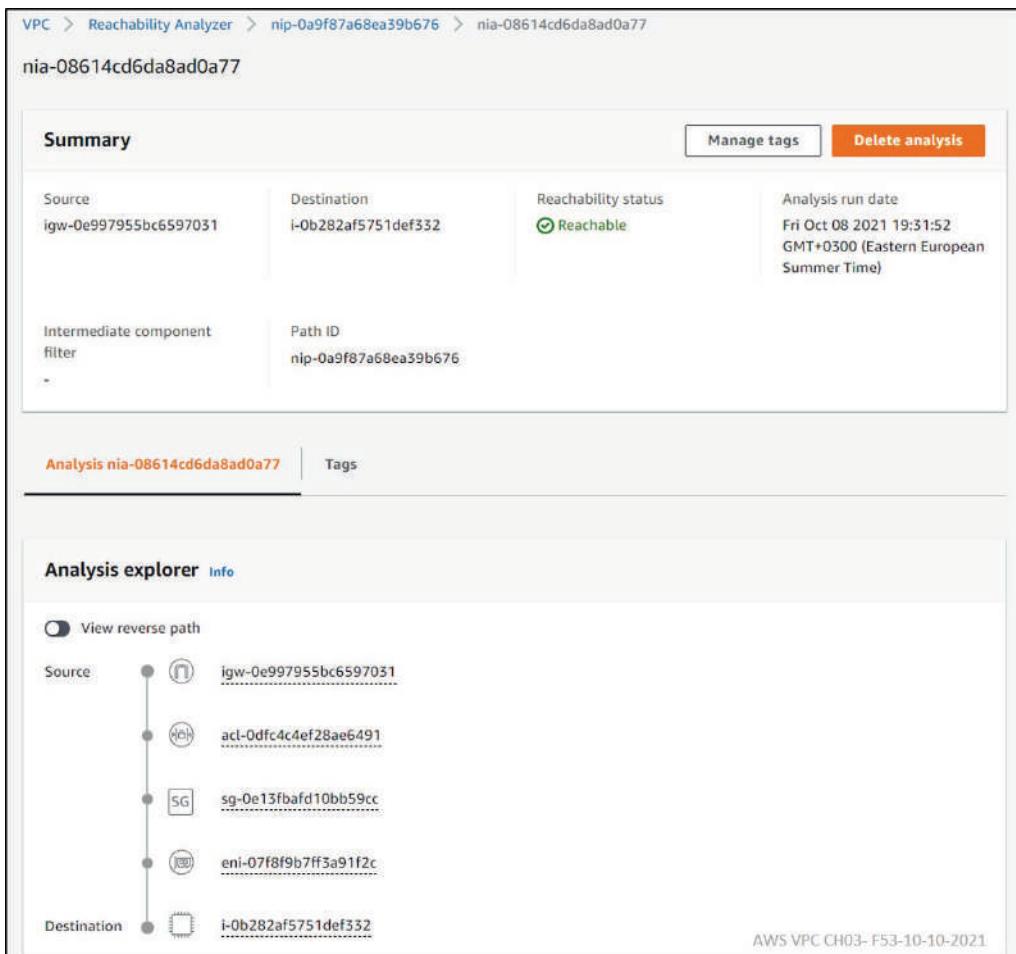


Figure 3-53: Reachability Analyzer – Visual View.

Billing

The last but not least subject is billing. Figure 3-54 summarizes the cost components for our example environment. Note that AWS charges \$0.1 per Path Analysis. Running t2.micro is not expensive. It has been running for 61 hours which costs only 0.81\$. It means approximately 10 €/month.

AWS Service Charges		\$1.32
▼ Data Transfer		\$0.00
▼ EU (London)		\$0.00
Bandwidth		\$0.00
\$0.000 per GB - data transfer in per month	0.000074 GB	\$0.00
\$0.000 per GB - first 1 GB of data transferred out per month	0.000084 GB	\$0.00
▼ Elastic Compute Cloud		\$0.87
▼ EU (London)		\$0.87
Amazon Elastic Compute Cloud running Linux/UNIX		\$0.81
\$0.0132 per On Demand Linux t2.micro Instance Hour	61.000 Hrs	\$0.81
EBS		\$0.07
\$0.116 per GB-month of General Purpose SSD (gp2) provisioned storage - EU (London)	0.581 GB-Mo	\$0.07
▼ Virtual Private Cloud		\$0.20
▼ EU (London)		\$0.20
No Instance Type		\$0.20
\$0.1 per analysis processed by VPC Reachability Analyzer	2.000 ReachabilityAnalyses	\$0.20
Taxes		
VAT to be collected		\$0.25

Usage and recurring charges for this statement period will be charged on your next billing date. Estimated charges shown on this page, or shown on any notifications that we send to you, may differ from your actual charges for this statement period. This is because estimated charges presented on this page do not include usage charges accrued during this statement period after the date you view this page. Similarly, information about estimated charges sent to you in a notification do not include usage charges accrued during this statement period after the date we send you the notification. One-time fees and subscription charges are assessed separately from usage and recurring charges, on the date that they occur.

AWS VPC CH03 - F54-10-10-2021

Figure 3-54: Billing.

Chapter 4: VPC NAT Gateway

Introduction

Back-End EC2 instances like Application and Database servers are most often launched on a Private subnet. As a recap, a Private subnet is a subnet that doesn't have a route to the Internet Gateway in its Route table. Besides, EC2 instances in the Private subnet don't have Elastic-IP address association. These two facts mean that EC2 instances on the Private subnet don't have Internet access. However, these EC2 instances might still need occasional Internet access to get firmware upgrades from the external source. We can use a NAT Gateway (NGW) for allowing IPv4 Internet traffic from Private subnets to the Internet. When we launch an NGW, we also need to allocate an Elastic-IP address (EIP) and associate it with the NGW. This association works the same way as the EIP-to-EC2 association. It creates a static NAT entry to IGW that translates NGW's local subnet address to its associated EIP. The NGW, in turn, is responsible for translating the source IP address from the ingress traffic originated from the Private subnet to its local subnet IP address. As an example, EC2 instance NWKT-EC2-Back-End sends packets towards the Internet to NGW. When the NGW receives these packets, it rewrites the source IP address 10.10.1.172 with its Public subnet IP address 10.10.0.195 and forwards packets to the Internet gateway. IGW translates the source IP address 10.10.0.195 to EIP 18.132.96.95 (EIP associated with NGW). That means that the source IP of data is rewritten twice, first by NGW and then by IGW.

Figure 4-1 illustrates our example NAT GW design and its configuration steps. As a pre-task, we launch an EC2 instance on the Private subnet 10.10.1.0/24 (1). We also modify the existing Security Group (SG) to allow an Inbound/Outbound ICMP traffic within VPC CIDR 10.10.0.0/16 (2). We also allow an SSH session initiation from the 10.10.0.218/24. I'm using the same SG for both EC2 instances to keep things simple. Besides, both EC2 uses the same Key Pair. Chapter 3 shows how to launch an EC2 instance and how we modify the SGs, and that is why we go ahead straight to the NGW configuration.

When we have done pre-tasks, we launch an NGW on the Public subnet (3). Then we allocate an EIP and associate it with NGW (4). Next, we add a default route towards NGW on the Private subnet Route Table (5).

The last three steps are related to connectivity testing. First, verify Intra-VPC IP connectivity using ICMP (6). Then we test the Internet connectivity (7). As the last

step, we can confirm that no route exists back to NWKT-EC-Back-End from the IGW. We are using an AWS Path Analyzer for that (8).

Note! Our example doesn't follow good design principles. AWS Availability Zones (AZ) are restricted failure domains, which means that failure in one AZ doesn't affect the operation of another AZ. Now, if our NGW on AZ eu-west-2c fails, Internet traffic from the Private subnet on eu-west2a fails. The proper design is to launch NGW on the AZ where unidirectional egress Internet access is needed.

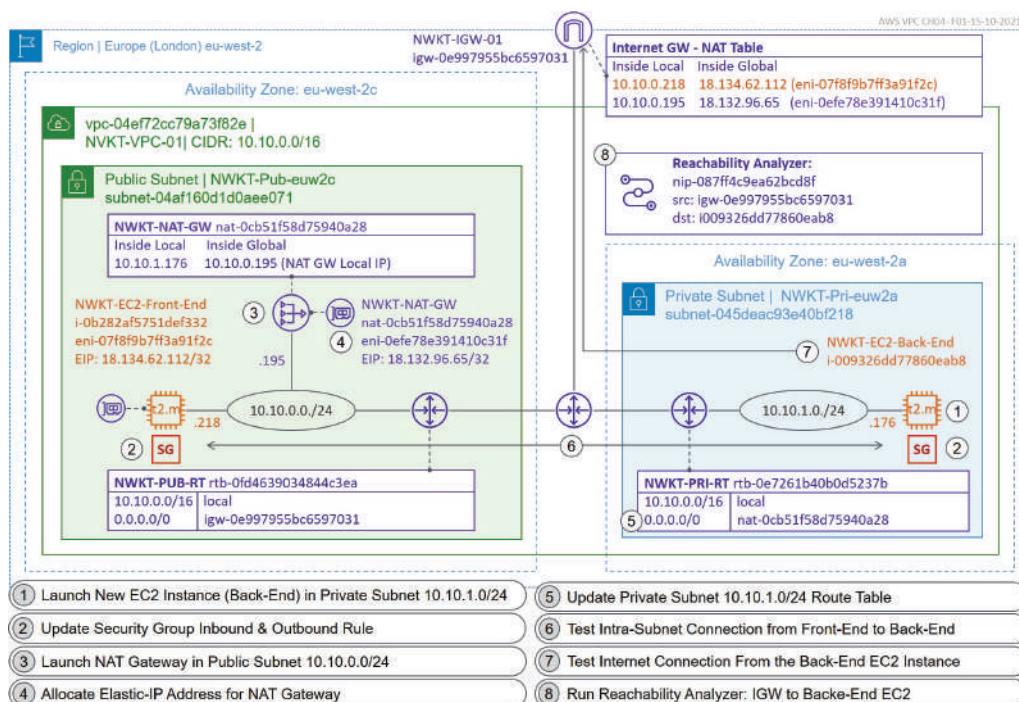


Figure 4-1: Example Topology.

Create NAT Gateway and Allocate Elastic IP

Navigate to the AWS VPC dashboard and select NAT Gateways.

The screenshot shows the AWS VPC dashboard with the 'New VPC Experience' header. On the left, there's a sidebar with various VPC-related options like 'Your VPCs', 'Subnets', 'Route Tables', etc. A dashed box highlights the 'NAT Gateways' option under 'Endpoint Services'. The main area is titled 'Resources by Region' and shows the following data:

Resource Type	Region	Count	Action
VPCs	Europe	3	See all regions
NAT Gateways	Europe	0	See all regions
Subnets	Europe	6	See all regions
VPC Peering Connections	Europe	0	See all regions
Route Tables	Europe	6	See all regions
Network ACLs	Europe	3	See all regions
Internet Gateways	Europe	2	See all regions
Security Groups	Europe	4	See all regions
Egress-only Internet Gateways	Europe	0	See all regions
Customer Gateways	Europe	0	See all regions

Figure 4-2: Create NAT Gateway – Step 1.

We don't have any NGW at this phase. Click the Create NAT gateway button.

The screenshot shows the 'Create NAT gateway' wizard step 1. At the top, there's a header with 'NAT gateways' and a 'Create NAT gateway' button. Below is a search bar with the placeholder 'Filter NAT gateways'. The main area is a table with the following columns:

Name	NAT gateway ID	Connectivity	State	State message
<hr/>				

Figure 4-3: Create NAT Gateway – Step 2.

Fill in the optional Name field. It creates a Key/Value Pair where the key is Name and the Value NWKT-NAT-GW. Select the Private subnet NWKT-Pub-euw2c (subnet-04af160d1d10aee071). We have two options for the Connectivity option. Using the default option Public, we allow EC2 instances from Private subnets gets uni-directional, stateful connection to the Internet. That means that they are not accessible from the Internet. The public option requires an Elastic IP address. The Private option allows connection from the Private subnet to on-prem resources, either via Transit Gateway or Virtual Private Gateway. You can't associate EIP with Private NGW. Select the Public option. Then click the Allocate Elastic IP address button. Figure 4-5 shows that AWS has allocated the EIP allocation Id eipalloc-0cda321ba6a5c5e06. Click the Create NAT gateway button to launch the NGW.

VPC > [NAT gateways](#) > Create NAT gateway

Create NAT gateway [Info](#)

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

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

The name can be up to 256 characters long.

Subnet
Select a subnet in which to create the NAT gateway.

▼

Connectivity type
Select a connectivity type for the NAT gateway.

Public
 Private

Elastic IP allocation ID [Info](#)
Assign an Elastic IP address to the NAT gateway.

Select an Elastic IP

AWS VPC CH04- F04-15-10-2021

Figure 4-4: Create NAT Gateway – Step 3.1.

Create NAT gateway Info

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

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

The name can be up to 256 characters long.

Subnet
Select a subnet in which to create the NAT gateway.

Connectivity type
Select a connectivity type for the NAT gateway.

Public
 Private

Elastic IP allocation ID Info
Assign an Elastic IP address to the NAT gateway.

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.

Key	Value - optional	Remove
<input type="text" value="Name"/> <input type="button" value="X"/>	<input type="text" value="NWKT-NAT-GW"/> <input type="button" value="X"/>	<input type="button" value="Remove"/>

You can add 49 more tags.

Figure 4-5: Create NAT Gateway – Step 3.2.

Figure 4-6 shows that the State of the process is Pending. Our NGW Id is nat-0cb51f58d75940a28. At this phase, the EIP, Private IP address, and Network Interface Id are not visible.



Figure 4-6: Create NAT Gateway – Step 4.1.

After the state is changed to Available, we can see the allocated EIP (18.132.96.65), the Private IP address (10.10.0.195), and the Elastic Network Interface Id (eni-0efe78e391410c31f).

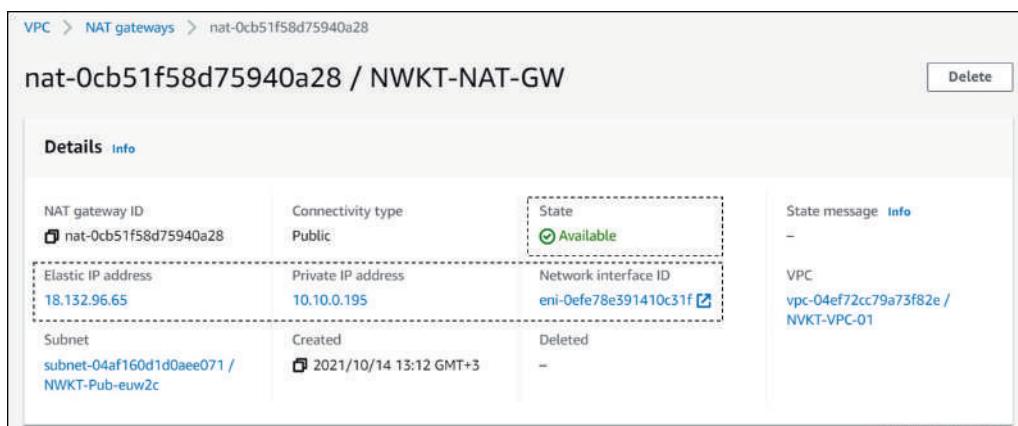


Figure 4-7: Create NAT Gateway – Step 4.2.

Example 4-1 shows the same information in JSON format.

```
aws ec2 describe-nat-gateways
{
    "NatGateways": [
        {
            "CreateTime": "2021-10-14T10:12:46.000Z",
            "NatGatewayAddresses": [
                {
                    "AllocationId": "eipalloc-0cda321ba6a5c5e06",
                    "NetworkInterfaceId": "eni-0efe78e391410c31f",
                    "PrivateIp": "10.10.0.195",
                    "PublicIp": "18.132.96.65"
                }
            ],
            "NatGatewayId": "nat-0cb51f58d75940a28",
            "State": "available",
            "SubnetId": "subnet-04af160d1d0aee071",
            "VpcId": "vpc-04ef72cc79a73f82e",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "NWKT-NAT-GW"
                }
            ]
        }
    ]
}
```

Example 4-1: Confirm the Nat Gateway from the AWS CLI.

Example 4-2 shows our two EIP allocations.

```
aws ec2 describe-addresses --output table
+-----+-----+
|             DescribeAddresses           |
+-----+-----+
||             Addresses               ||
+-----+-----+
|| AllocationId | eipalloc-0cda321ba6a5c5e06   ||
|| AssociationId | eipassoc-003e422c757246773  ||
|| Domain       | vpc                         ||
|| InstanceId   |                            ||
|| NetworkInterfaceId | eni-0efe78e391410c31f ||
|| NetworkInterfaceOwnerId | 123456654321  ||
|| PrivateIpAddress | 10.10.0.195   ||
|| PublicIp      | 18.132.96.65  ||
|| PublicIpv4Pool | amazon          ||
+-----+-----+
||             Tags                  ||
+-----+-----+
|| Key     | Name   ||
|| Value   | NWKT-NAT-GW ||
+-----+-----+
||             Addresses             ||
+-----+-----+
|| AllocationId | eipalloc-01c7d2a4877a061a7   ||
|| AssociationId | eipassoc-0b82dad3a3908bbf2  ||
|| Domain       | vpc                         ||
|| InstanceId   | i-0b282af5751def332  ||
+-----+-----+
```

NetworkInterfaceId	eni-07f8f9b7ff3a91f2c
NetworkInterfaceOwnerId	123456654321
PrivateIpAddress	10.10.0.218
PublicIp	18.134.62.112
PublicIpv4Pool	amazon
Tags	
Key	Name
Value	NWKT-PUBLIC-IP

Example 4-2: Elastic IP Address Allocation.

Add Route to NGW on Private Subnet Route Table

We have already created Subnet 10.10.1.0/24 specific NWKT-PRI-RT in chapter one. Though, we didn't associate a subnet with it. Navigate to the Route Table view and select the Route Table NWKT-PRI-RT. Next, associate subnet NWKT-Pri-euw2a (subnet-045deac93 e40bf218) with it.

The screenshot shows the AWS Route Tables page with the following details:

- Route tables (1/6) Info**: Shows 1 route table (rtb-0e7261b40b0d5237b / NWKT-PRI-RT).
- Filter route tables**: A search bar.
- Actions**: A dropdown menu.
- Create route table**: An orange button.
- Table Headers**: Name, Route table ID, Explicit subnet associations, Main.
- Data Rows**:
 - NWKT-PRI-RT**: Selected (checked), Route table ID rtb-0e7261b40b0d5237b, Explicit subnet associations: subnet-045deac93e40bf218 / NWKT-Pri-euw2a, Main: No.
 - DFTL-RTBL**: Unselected, Route table ID rtb-8edeeae6, Explicit subnet associations: -, Main: Yes.
 - NWKT-PUB2-RT**: Unselected, Route table ID rtb-01552a97d0e6878..., Explicit subnet associations: subnet-08eee681493045f37 / NWKT-PUB2-e..., Main: No.
- rtb-0e7261b40b0d5237b / NWKT-PRI-RT** (Details):
 - Subnet associations** (selected tab): Shows 1 explicit subnet association.
 - Details**, **Routes**, **Edge associations**, **Route propagation**, **Tags** tabs.
- Explicit subnet associations (1)**:
 - Edit subnet associations** button.
 - Find subnet association** search bar.
 - Subnet ID**: subnet-045deac93e40bf218 / NWKT-Pri-euw2a.
 - IPv6 CIDR**: -.
 - AWS VPC CH04- F08-15-10-2021

Figure 4-8: Associate Private Subnet to Route Table.

Next, we add the default route to the Route Table. Click the Edit routes button.

The screenshot shows the AWS VPC Route Tables interface. At the top, the navigation path is VPC > Route tables > rtb-0e7261b40b0d5237b. The main title is rtb-0e7261b40b0d5237b / NWKT-PRI-RT. On the right, there is an 'Actions' dropdown menu. Below the title, there is a 'Details' section with tabs for 'Info' (selected) and 'Logs'. The 'Info' tab displays the following details:

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0e7261b40b0d5237b	No	subnet-045deac93e40bf218 / NWKT-Pri-euw2a	-
VPC	Owner ID	017857243309	
vpc-04ef72cc79a73f82e		NVKT-VPC-01	

Below the details, there are tabs for 'Routes' (selected), 'Subnet associations', 'Edge associations', 'Route propagation', and 'Tags'. The 'Routes' tab shows a table with one route entry:

Routes (1)			
Destination	Target	Status	Propagated
10.10.0.0/16	local	Active	No AWS VPC CH04- F09-15-10-2021

On the right side of the routes table, there is an 'Edit routes' button, a search bar with a 'Filter routes' placeholder, and pagination controls (1 of 1).

Figure 4-9: Add Default Route to Route Table.

Fill in the Destination field with 0.0.0.0/0. Then select NAT Gateway from the Target drop-down menu.

The screenshot shows the 'Edit routes' section of the AWS VPC console. On the left, there are two 'Edit routes' sections. The top one has a 'Destination' of 10.10.0.0/16 and is 'Propagated' No. The bottom one has a 'Destination' of 0.0.0.0/0 and is 'Propagated' No. Both sections have a 'Remove' button and an 'Add route' button. On the right, a large list of targets is shown in a table format. A dashed box highlights the 'NAT Gateway' row, which is selected. This row has a 'Status' column showing 'Active' with a green checkmark. Other targets listed include Carrier Gateway, Egress Only Internet Gateway, Gateway Load Balancer Endpoint, Instance, Internet Gateway, local, Network Interface, Outpost Local Gateway, Peering Connection, Transit Gateway, and Virtual Private Gateway. A search bar is at the bottom of the target list. The bottom right corner of the interface shows the timestamp 'AWS VPC CH04- F10-15-10-2023'.

Figure 4-10: Route 0.0.0.0/0 to NAT Gateway.

Select our NAT Gateway from the list and click the Save changes button.

The screenshot shows a confirmation dialog for saving route table changes. It has a 'Target' field containing 'nat-0cb51f58d75940a28' with a search icon and an 'X' button. To the right is a 'Status' field showing '-'. At the bottom are three buttons: 'Cancel', 'Preview', and a prominent orange 'Save changes' button. The bottom right corner shows the timestamp 'AWS VPC CH04- F11-15-10-2023'.

Figure 4-11: Save Route Table Changes.

```
aws ec2 describe-route-tables --filters Name=tag:Name,Values=NWKT-PRI-RT --output table
```

DescribeRouteTables				
RouteTables				
OwnerId		017857243309		
RouteTableId		rtb-0e7261b40b0d5237b		
VpcId		vpc-04ef72cc79a73f82e		
Associations				
Main		False		
RouteTableAssociationId		rtbassoc-0c17e355e3b156dda		
RouteTableId		rtb-0e7261b40b0d5237b		
SubnetId		subnet-045deac93e40bf218		
Routes				
DestinationCidrBlock	GatewayId	NatGatewayId	Origin	State
10.10.0.0/16	local	nat-0cb51f58d75940a28	CreateRouteTable	active
0.0.0.0/0			CreateRoute	active
Tags				
Key		Name		
Value		NWKT-PRI-RT		

Example 4-3: Route Table of Subnet NWKT-PRI-RT.

Test Connections

We are going to use NWKT-EC2-Fron-End as a Jump Server to access NWKT-EC-Back-End. To do that, I copy the shared NWKT-KEY-EC2.pem file to the instance NWKT-EC-Front-End. Note that this is not a recommended solution. In reality, you should never store a pem file in the EC2 instance which uses it for authentication.

```
scp -i NWKT-KEY-EC2.pem NWKT-KEY-EC2.pem ec2-user@ec2-18-134-62-112.eu-west-2.compute.amazonaws.com:/home/ec2-user
```

Example 4-4: Route Table of Subnet NWKT-PRI-RT.

Then we open an SSH connection to instance NWKT-EC2-Front-End. As a first test, we ping the Private IP address of the instance NWKT-EC-Back-END (10.10.1.176). Example 4-5 shows that it works fine.

```
C:\folder-1\folder-2\AWS> ssh -i "NWKT-KEY-EC2.pem" ec2-user@ec2-18-134-62-112.eu-west-2.compute.amazonaws.com

Last login: Mon Oct 11 10:39:06 2021 from 91-153-26-170.elisa-laajakaista.fi

      _\|_ )   Amazon Linux 2 AMI
     __| \__|_|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-10-0-218 ~]$ ping 10.10.1.176
PING 10.10.1.176 (10.10.1.176) 56(84) bytes of data.
64 bytes from 10.10.1.176: icmp_seq=1 ttl=255 time=3.27 ms
64 bytes from 10.10.1.176: icmp_seq=2 ttl=255 time=0.983 ms
^C
--- 10.10.1.176 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.983/2.129/3.275/1.146 ms
```

Example 4-5: Route Table of Subnet NWKT-PRI-RT.

As the last test, we open an SSH connection to instance NWKT-EC2-Back-End and ping the Google DNS Anycast address. We can see that the connection works fine.

```
[ec2-user@ip-10-10-0-218 ~]$ ssh -i "NWKT-KEY-EC2.pem" ec2-user@ip-10-10-1-176.eu-west-2.compute.internal

Last login: Thu Oct 14 10:07:15 2021 from ip-10-10-0-218.eu-west-2.compute.internal
      _\|_ )   Amazon Linux 2 AMI
     __| \__|_|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-10-1-176 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=100 time=3.19 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=100 time=2.83 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=100 time=2.72 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=100 time=2.78 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=100 time=2.74 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 2.720/2.854/3.192/0.176 ms
[ec2-user@ip-10-10-1-176 ~]$
```

Example 4-6: Route Table of Subnet NWKT-PRI-RT.

As the last step, we use the Reachability Analyzer tool to confirm that the instance NWKT-EC2-Back-End is not reachable from the Internet. We are using IGW as a source and the instance NWKT-EC2-Back-End as the destination. The destination port is 22 and protocol TCP.

Create and analyze path Info

A path consists of a source, destination, protocol, and optional destination port. When you analyze a path, we determine whether it is reachable and identify the intermediate components. If the path is not reachable, we identify the blocking components. You are charged each time you analyze a path.

Path configuration

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

Source type: Internet Gateways **Source**: igw-0e997955bc6597031 **Source IP address - optional**: 192.0.2.1

Destination type: Instances **Destination**: i-009326dd77860eab8 **Destination IP address - optional**: 192.0.2.1

Destination port - optional: 22
Number must be between 0 and 65535

Protocol
Use the appropriate protocol

TCP

AWS VPC © 2021 F12-15-10-2021

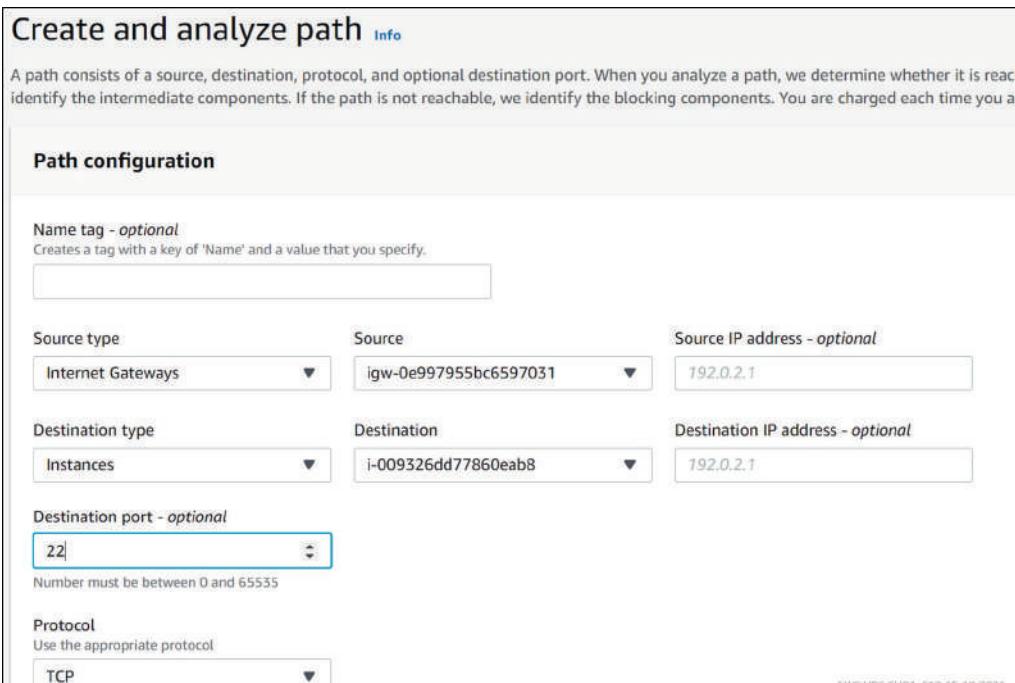


Figure 4-12: Reachability Analyzer – Define Path Information.

VPC > Reachability Analyzer > nip-087ff4c9ea62bcd8f AWS VPC CH04- F13-15-10-2023

nip-087ff4c9ea62bcd8f

Summary		Actions	
Path ID nip-087ff4c9ea62bcd8f	Last analysis date Thu Oct 14 2021 13:34:37 GMT+0300 (Eastern European Summer Time)	Reachability status ✖ Not reachable	Last analysis status ✔ Succeeded
Source igw-0e997955bc6597031	Destination i-009326dd77860eab8	Destination port 22	Protocol TCP

Analyses Analyses (1/1) Info

C Delete analysis

Filter path analyses

Analysis ID	Analysis run date	Reachability status	I...	State
nia-069492c678135d60f	Thu Oct 14 2021 13:3...	✖ Not reachable	-	✔ Succeeded

Figure 4-13: Reachabilit Analyzer Test – Result: Nor Reachable.

Billing

Figure 4-14 shows a detailed report of our costs so far. It is self-explanatory, and I won't go through it in detail.

AWS Service Charges		\$3.47
▼ Data Transfer		\$0.00
▼ EU (London)		\$0.00
AWS Data Transfer EUW2-EU-AWS-In-Bytes		\$0.00
\$0.00 per GB - EU (London) data transfer from EU (Ireland)	0.000000040 GB	\$0.00
Bandwidth		\$0.00
\$0.00 per GB - data transfer in per month	0.000331 GB	\$0.00
\$0.00 per GB - first 1 GB of data transferred out per month	0.000347 GB	\$0.00
\$0.010 per GB - regional data transfer - in/out/between EC2 AZs or using elastic IPs or ELB	0.000175 GB	\$0.00
▼ Elastic Compute Cloud		\$2.42
▼ EU (London)		\$2.42
Amazon Elastic Compute Cloud NatGateway		\$0.10
\$0.05 per GB Data Processed by NAT Gateways	0.000001 GB	\$0.00
\$0.05 per NAT Gateway Hour	2.000 Hrs	\$0.10
Amazon Elastic Compute Cloud running Linux/UNIX		\$2.13
\$0.0132 per On Demand Linux t2.micro Instance Hour	161.179 Hrs	\$2.13
EBS		\$0.19
\$0.116 per GB-month of General Purpose SSD (gp2) provisioned storage - EU (London)	1.637 GB-Mo	\$0.19
▼ Key Management Service		\$0.00
▼ EU (London)		\$0.00
AWS Key Management Service eu-west-2-KMS-Requests		\$0.00
\$0.00 per request - Monthly Global Free Tier for KMS requests	1.000 Requests	\$0.00
▼ Virtual Private Cloud		\$0.40
▼ EU (London)		\$0.40
No Instance Type		\$0.40
\$0.1 per analysis processed by VPC Reachability Analyzer	4.000 ReachabilityAnalyses	\$0.40
Taxes		
VAT to be collected		AWS VPC CH04- F14-15-10-2021 \$0.65

Figure 4-14: Billing.

Chapter 5: Virtual Private Gateway - VGW

Introduction

A Hybrid Architecture refers to an application design model where part of the application resources are running locally on the corporate's on-prem Datacenter while some resources are running in the Cloud. For example, we can run Web application front-end Web servers locally in on-prem Datacenter, while Back-End Application and Database servers are running in the Cloud. Another example is the solution where both front-end and back-end workloads are in the on-prem Datacenter, and their storage resources are in the Cloud. At the time of writing, AWS has three connectivity solutions for Hybrid Architecture: 1) VPC specific Site-to-Site VPN with Virtual Private Gateway, 2) VPN for Multi-VPC connection with Transit Gateway, 3) and AWS Direct Connect. This chapter introduces the Single-VPC site-to-site VPN with Virtual Private Gateway (VGW). Note that VGW is AWS managed entity, and you don't have access to its configuration. That means that to build a VPC egress policy, we need to use the BGP MED attribute or AS-PATH prepending.

Figure 5-1 illustrates our example environment where we have a Corporate on-prem Datacenter and virtual Datacenter NVKT-VPC-01 in the AWS region eu-west-2 (London). Our intent is to build a VPN connection between these two sites to establish a secure IP connection between EC2 instances within VPC and subnet 172.16.10.0/24 in on-prem DC. First, we need to create Customer Gateway (CGW) and Virtual Private Gateway (VGW). Then we attach VGW into the VPC and enable route propagation from subnet NWKT-PUB-RT Route Table to BGP process. After these steps, we can set up a VPN connection between VGW and CGW. As the last step, we download an AWS-generated CGW configuration file. After filling in the local Interface/IP address information, we can upload the configuration into the CGW.

When we implement an AWS VGW, we launch two high availability endpoints located in different Availability Zone with Region. This way, AWS can do device life cycle management without causing downtime for customers. You can also use two CGWs in an on-prem location for high availability. Another thing to keep in mind is that you can attach a VGW to a single VPC and use the VPN connection only for resources within VPC. That means that resources outside VPC, like S3 (Simple Storage Resource), are not available via VPN.

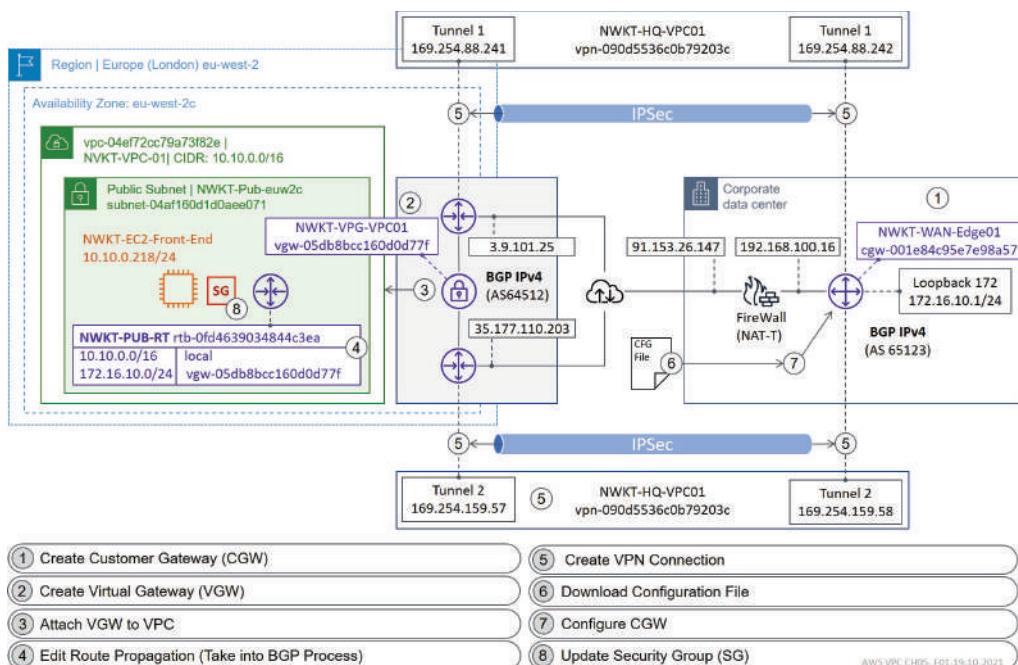


Figure 5-1: Site-to-Site VPN Example Environment.

Customer Gateway (CGW)

Customer Gateway (CGW) is a VPN termination point in the on-prem Datacenter. When creating a CGW, we define its name, public IP address, and BGP AS number. Note that our CGW is behind Firewall, which translates the CGW's local IP address to a public IP address. These settings are then used for VGW configuration.

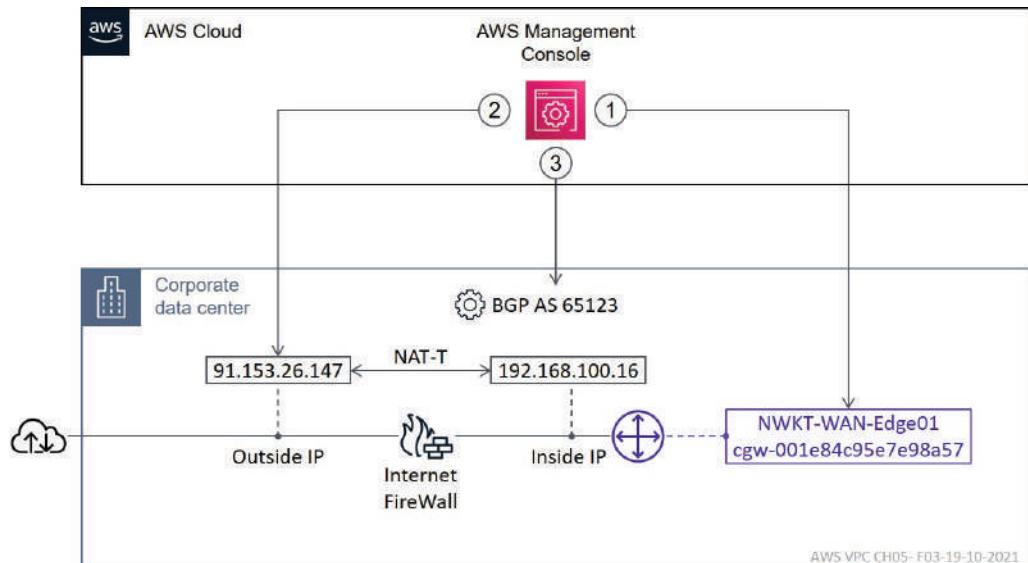


Figure 5-2: Customer Gateway (CGW).

Create CGW

Navigate to the VPC dashboard and select *Customer Gateway*.

The screenshot shows the AWS VPC dashboard with the following interface elements:

- New VPC Experience**: A feedback link.
- VIRTUAL PRIVATE CLOUD**: A main category.
- SECURITY**, **REACHABILITY**, **DNS FIREWALL**, **NETWORK FIREWALL**: Sub-categories under VPC.
- VIRTUAL PRIVATE NETWORK (VPN)**: A main category.
- Customer Gateways**: A sub-category under VPN, highlighted with a dashed box.
- Virtual Private Gateways**, **Site-to-Site VPN Connections**, **Client VPN Endpoints**: Options under Customer Gateways.
- TRANSIT GATEWAYS**: A main category.
- Transit Gateways New**, **Transit Gateway Attachments New**, **Transit Gateway Route Tables New**, **Transit Gateway Multicast New**, **Network Manager New**: Options under Transit Gateways.
- TRAFFIC MIRRORING**: A main category.
- Launch VPC Wizard** and **Launch EC2 Instances**: Buttons at the top right.
- Note: Your Instances will launch in the Europe region.**
- Resources by Region**: A section title.
- Refresh Resources**: A button in the Resources by Region section.
- You are using the following Amazon VPC resources**: A heading for the resource grid.
- VPCs**: Europe 3, See all regions ▾
- NAT Gateways**: Europe 0, See all regions ▾
- Subnets**: Europe 6, See all regions ▾
- VPC Peering Connections**: Europe 0, See all regions ▾
- Route Tables**: Europe 6, See all regions ▾
- Network ACLs**: Europe 3, See all regions ▾
- Internet Gateways**: Europe 2, See all regions ▾
- Security Groups**: Europe 4, See all regions ▾
- Egress-only Internet Gateways**: Europe 0, See all regions ▾
- Customer Gateways**: Europe 0, See all regions ▾ (highlighted with a dashed box)
- DHCP options sets**: Europe 1, See all regions ▾
- Virtual Private Gateways**: Europe 0, See all regions ▾
- Elastic IPs**: Europe 1, See all regions ▾
- Site-to-Site VPN Connections**: Europe 0, See all regions ▾
- Endpoints**: Europe 0

Figure 5-3: Create Customer Gateway, Step-1.

Click the *Create Customer Gateway* button.

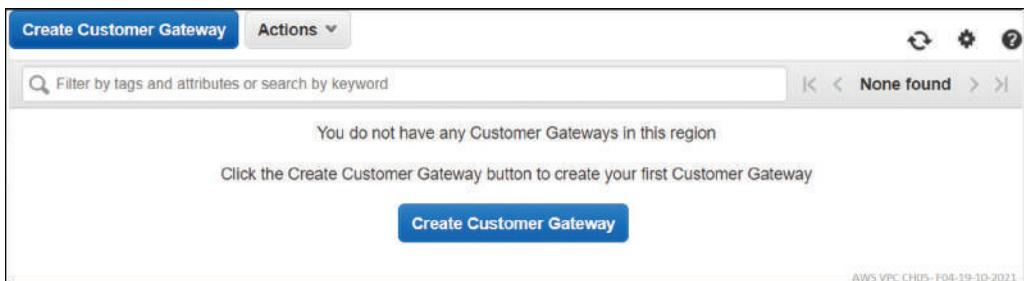


Figure 5-4: Create Customer Gateway, Step-2.

Fill in the name field and select the routing model. We are using dynamic routing with BGP. Define the AS number attached to our on-prem Datacenter, and then fill in the IP address. Click the *Create Customer Gateway* button to proceed.

Name	NWKT-WAN-Edge01	<small>i</small>
Routing	<input checked="" type="radio"/> Dynamic <input type="radio"/> Static	
BGP ASN*	65123	<small>i</small>
IP Address	91.153.26.147	<small>i</small>
Certificate ARN	Select Certificate ARN	<small>C i</small>
Device	Optional	<small>i</small>

* Required

AWS VPC CH05 - F04-19-10-2021

Figure 5-5: Create Customer Gateway, Step-3.

You will get a notification about the successful CGW creation process and the auto-generated CGW Id. Click the *Close* button.

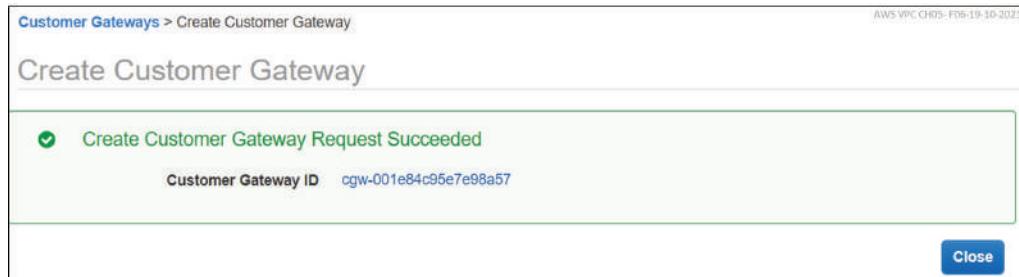


Figure 5-6: Create Customer Gateway, Step-4.

You will be forwarded to CGW view After clicking the Close button. Details tab shows the CGW Id cgw-001e84c95e7e98a57, Type IPSec, BGP ASN 65123, State Available and the IP address 91.153.26.147.

The screenshot shows the 'Customer Gateways' page in the AWS VPC console. A single entry is listed:

Name	ID	State	Type	IP Address	BGP ASN
NWKT-WAN-Edge01	cgw-001e84c95e7e98a57	available	ipsec.1	91.153.26.147	65123

Below the table, a detailed view for the gateway 'cgw-001e84c95e7e98a57' is shown. The 'Details' tab is selected, displaying the following information:

ID	cgw-001e84c95e7e98a57	State	available
Type	ipsec.1	IP Address	91.153.26.147
BGP ASN	65123	Certificate ARN	
Device	-	AWS VPC CH05 - F07-19-10-2021	

Figure 5-7: Create Customer Gateway, Step-5.

Virtual Gateway (VGW)

Virtual Gateway (VGW) is a VPC-specific VPN termination point on the AWS side. The VGW creation process is simple. First, we give it a name and assign BGP ASN to it. Then we attach the CGW to our selected VPC.

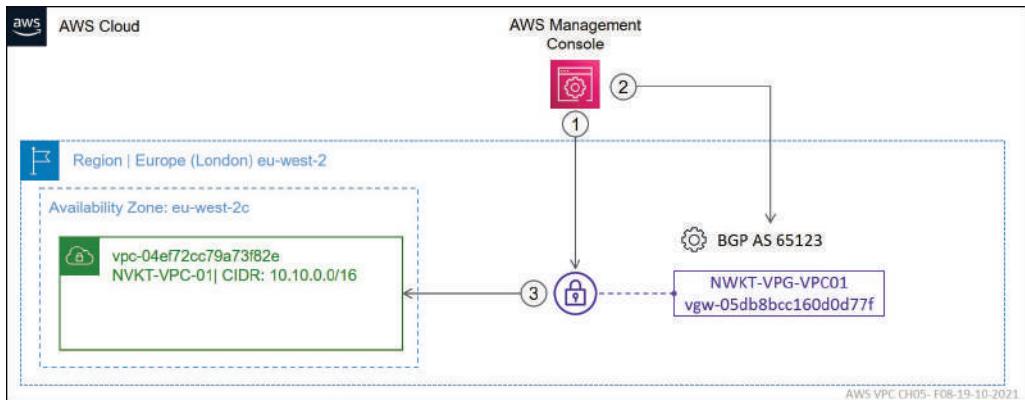


Figure 5-8: Virtual Private Gateway.

Create CGW

Navigate to the VPC dashboard and select the Virtual Private Gateways option (figure 5-3 in previous section).

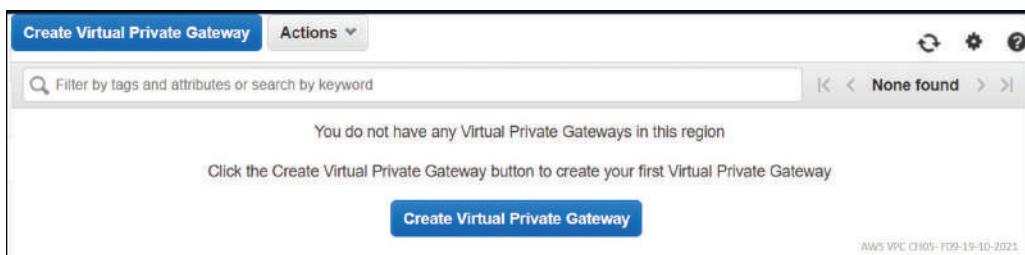


Figure 5-9: Create Virtual Private Gateway, Step-1.

Fill in the name field and select the Amazon default ASN radio button. You can also define your preferred BGP ASN. After clicking the Create Virtual Private Gateway button, the AWS console notifies you about the successful process. Click the Close button.

The screenshot shows two sequential steps in the AWS Create Virtual Private Gateway wizard:

Step 2: Create Virtual Private Gateway

- URL: Virtual Private Gateways > Create Virtual Private Gateway
- Description: A virtual private gateway is the router on the Amazon side of the VPN tunnel.
- Name tag: NWKT-VPG-VPC01
- ASN: Amazon default ASN (selected)
- Custom ASN: (radio button)
- Required: * Required
- Buttons: Cancel, Create Virtual Private Gateway

Step 3: Confirmation

- URL: Virtual Private Gateways > Create Virtual Private Gateway
- Description: Create Virtual Private Gateway succeeded
- Virtual Private Gateway ID: vgw-05db8bcc160d0d77f
- Buttons: Close

Figure 5-10: Create Virtual Private Gateway, Steps 2 and 3.

Attach CGW to VPC

The Details tab in Figure 5-11 shows the CGW that VGW the State is detached and the VPC field is empty. Select the Attach to VPC option from the Actions drop-down menu.

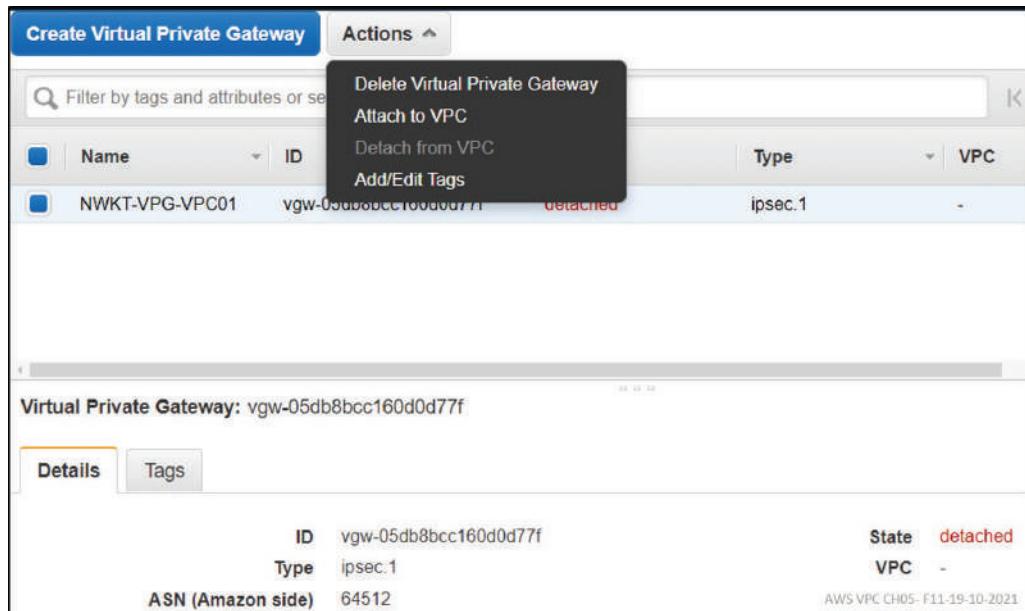


Figure 5-11: Attach VGW to VPC, Step-1.

Select our VPC `vpc-04ef72cc79a73f82e` from the VPC drop-down menu. When you open the drop-down menu, each VPC is listed with its name, so you don't have to remember the VPC Id. Next, click the Yes, Attach button.



Figure 5-12: Attach VGW to VPC, Step-2.

Figures 5-13 and 5-14 show the process. The state is first attaching, and then after a while, it turns to attached. Also, the VPC field shows the attached VPC with its name tag.

Name	ID	State	Type	VPC
NWKT-VPG-VPC01	vgw-05db8bcc160d0d77f	attaching	ipsec.1	vpc-04ef72cc79a73f82e NVKT-VPC-01

Virtual Private Gateway: vgw-05db8bcc160d0d77f

Details **Tags**

ID	vgw-05db8bcc160d0d77f	State	attaching
Type	ipsec.1	VPC	vpc-04ef72cc79a73f82e NVKT-VPC-01
ASN (Amazon side)	64512	AWS VPC C105-F13-19-10-2021	

Figure 5-13: Attaching VBGW to VPC – State attaching.

Name	ID	State	Type	VPC
NWKT-VPG-VPC01	vgw-05db8bcc160d0d77f	attached	ipsec.1	vpc-04ef72cc79a73f82e NVKT-VPC-01

Virtual Private Gateway: vgw-05db8bcc160d0d77f

Details **Tags**

ID	vgw-05db8bcc160d0d77f	State	attached
Type	ipsec.1	VPC	vpc-04ef72cc79a73f82e NVKT-VPC-01
ASN (Amazon side)	64512	AWS VPC C105-F14-19-10-2021	

Figure 5-14: Attaching VBGW to VPC – State attached.

Route Table Propagation

We are using a BGP for route advertisement over a VPN connection. Route Table Propagation defines the Route Table which routes we want VGW to advertise to on-prem CGW.

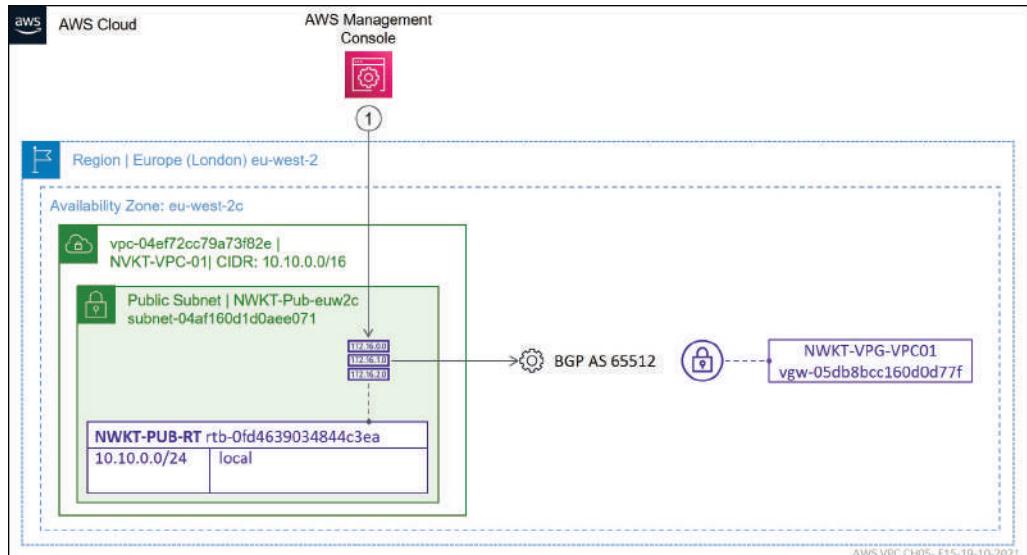


Figure 5-15: Route Table Route Propagation.

Edit Route Table Route Propagation

Navigate to the Route Table window (shown in Chapter 1, figure 1-26) and select Route table. Choose the Edit route propagation option from the Actions drop-down menu.

Route table	ID	Subnets
NWKT-PUB2-RT	rtb-01552a97d0e687854	subnet-08eeee
-	rtb-0d1a7e36c4412df0f	-
NWKT-MAIN-RT	rtb-069ac98ac692271fe	-
NWKT-PUB-RT	rtb-0fd4639034844c3ea	subnet-04af16

Figure 5-16: Route Table Route Propagation.

Enable route propagation and click the Save button.

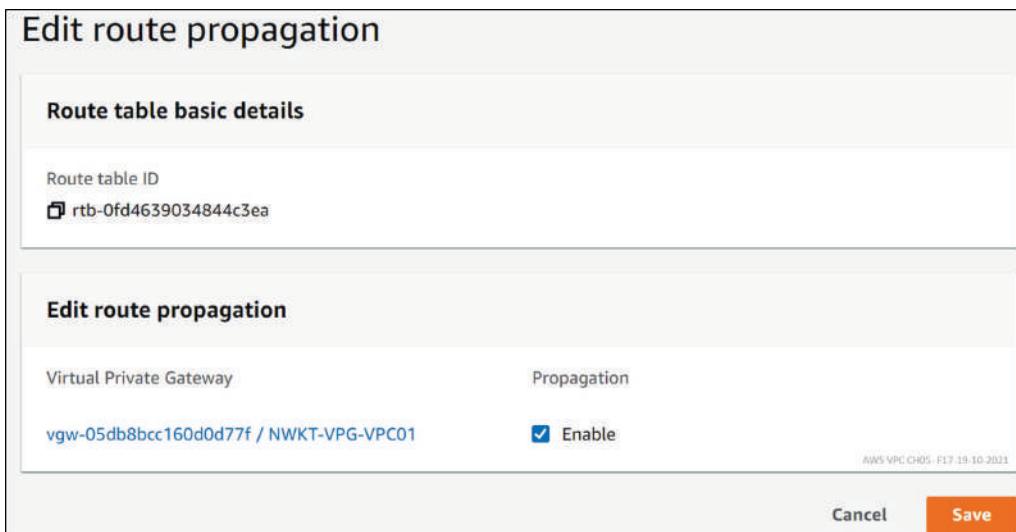


Figure 5-17: Route Table Route Propagation.

Figure 5-18 shows that we have successfully updated route table route propagation.

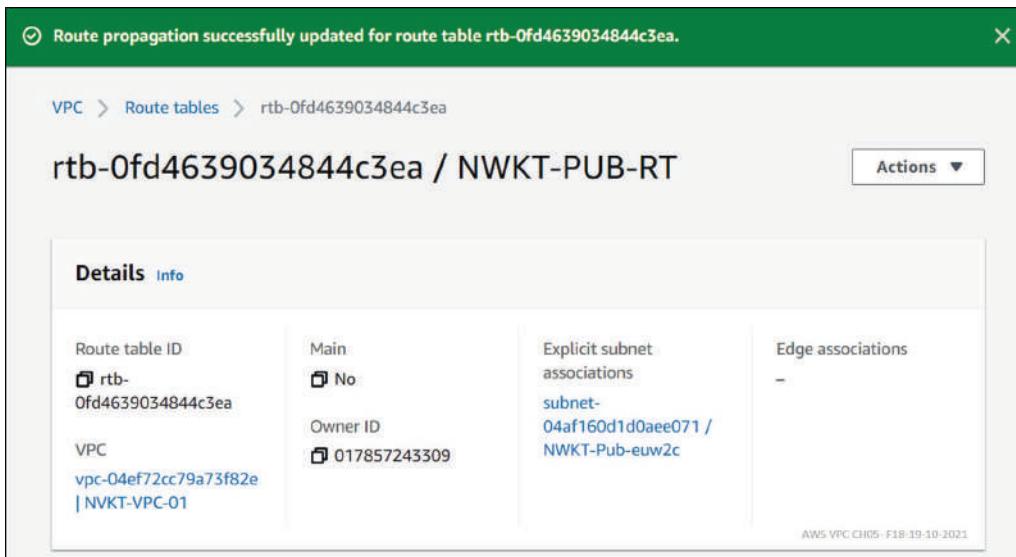


Figure 5-18: Route Table Route Propagation.

VPN Connection

We have already created a Customer Gateway (CGW) and Virtual Private Gateway (VGW). Now we create a VPN connection between them. First, we give a name to our VPN connection (1) and define the VPN termination endpoints (CGW 2 and VGW 3). Next, we enable dynamic routing (BGP is the only option) between endpoints (4 and 5). I left the Local and Remote IPv4 Network to their default value. However, you can define a more specific network range. We use Amazon-generated Inside CIDR and Pre-Shared Keys for both tunnels. We also use Amazon default Advanced option for both tunnels.

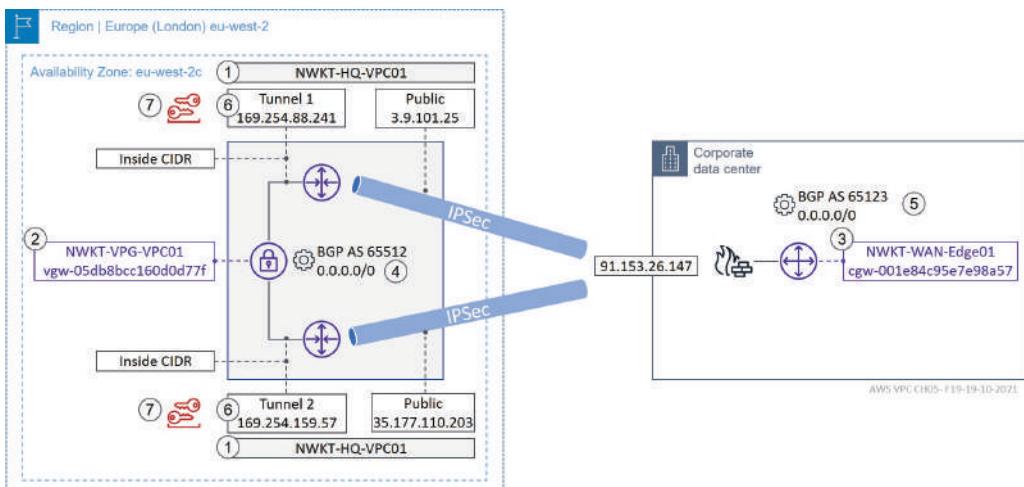


Figure 5-19: VPN Connection.

Edit Route Table Route Propagation

Navigate to the VPC dashboard (figure 5-3) and select Site-to-Site VPN Connections. Next, click the Create VPN Connection button.

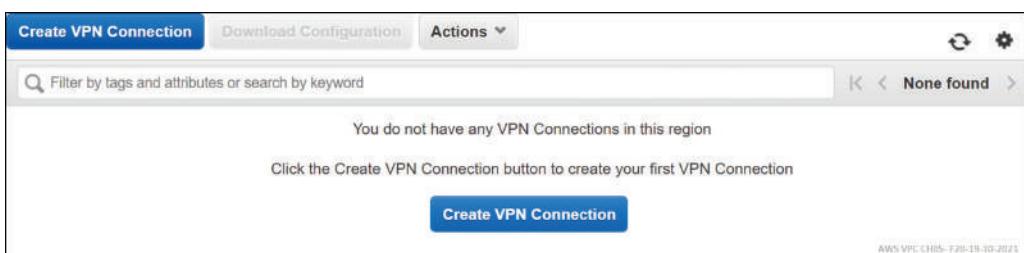


Figure 5-20: Create VPN Connection, Step-1.

Fill in the Name tag field. Select VGW vgw-05db8bcc160d0d77f from the *Virtual Private Gateway* drop-down menu, and cgw-001e84c95e7e98a57 from the *Customer Gateway ID* drop-down menu. Note that Name tags for both VGW and CGW are shown in drop-down menus, so you don't have to remember their IDs. Select the *Dynamic (requires BGP)* option as Routing Options and the IPv4 as Tunnel Inside Ip version. You can define subnets for both Local and Remote IPv4 Network CIDR fields. We use default CIDR 0.0.0.0/0. By doing this, the VGW will advertise all routes that we propagate from the Subnet Route Table (10.10.0.0/24). From the CGW point of view, this generates a BGP configuration where advertises a default route 0.0.0.0/0. We will change this later when we edit the configuration file.

Create VPN Connection

Select the target gateway and customer gateway that you would like to connect via a VPN connection. You must have entered the target gateway information already.

Name tag	NWKT-HQ-VPC01	<small>i</small>
Target Gateway Type	<input checked="" type="radio"/> Virtual Private Gateway <input type="radio"/> Transit Gateway	
Virtual Private Gateway*	vgw-05db8bcc160d0d77f	<small>C</small>
Customer Gateway	<input checked="" type="radio"/> Existing <input type="radio"/> New	
Customer Gateway ID*	cgw-001e84c95e7e98a57	<small>C</small>
Routing Options	<input checked="" type="radio"/> Dynamic (requires BGP) <input type="radio"/> Static	
Tunnel Inside Ip Version	<input checked="" type="radio"/> IPv4 <input type="radio"/> IPv6	
Local IPv4 Network Cidr	0.0.0.0/0	<small>i</small>
Remote IPv4 Network Cidr	0.0.0.0/0	<small>i</small>

AWS VPC CH05 - F21-10-2021

Figure 5-21: Create VPN Connection, Step-2.1.

Inside IPv4 CIDR for both tunnels is generated by Amazon. The same applies to Pre-Shared keys. To proceed, click the Create VPN Connection.

Tunnel Options

Customize tunnel Inside CIDR and pre-shared keys for your VPN tunnels. Unspecified tunnel options will be randomly generated by Amazon.

Inside IPv4 CIDR for Tunnel 1	Generated by Amazon	?
Pre-Shared Key for Tunnel 1	Generated by Amazon	?
Inside IPv4 CIDR for Tunnel 2	Generated by Amazon	?
Pre-shared key for Tunnel 2	Generated by Amazon	?

Advanced Options for Tunnel 1 Use Default Options Edit Tunnel 1 Options

Advanced Options for Tunnel 2 Use Default Options Edit Tunnel 2 Options

VPN connection charges apply once this step is complete. [View Rates](#)

* Required [Cancel](#) [Create VPN Connection](#)

Figure 5-22: Create VPN Connection, Step-2.2.

The AWS Console shows the notification about the successful VPN creation process. Click the Close button.

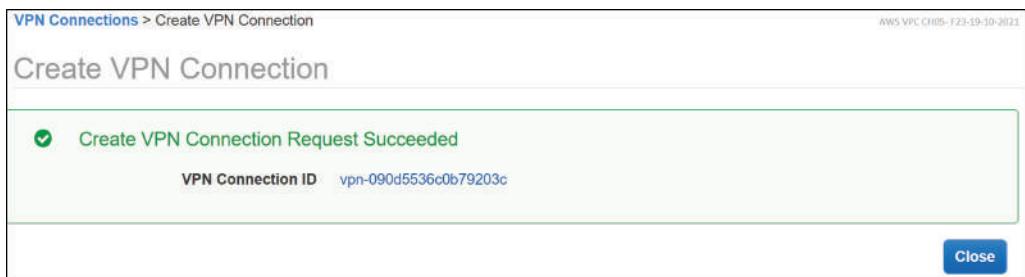


Figure 5-23: VPN Connection Verification.

The *Detail* tab in figure 5-24 shows the Id of VPN connection and its configuration parameters. It might take several minutes before the state is changed from pending to available (figure 5-25).

The screenshot shows the AWS VPC console interface for managing VPN connections. At the top, there are three buttons: "Create VPN Connection", "Download Configuration", and "Actions". Below these is a search bar with placeholder text "Filter by tags and attributes or search by keyword". A navigation bar indicates "1 to 1 of 1". The main table has columns for Name, VPN ID, State, Virtual Private Gateway, and Transit Gateway. One row is visible, representing a connection named "NWKT-HQ-VPC01" with a VPN ID of "vpn-090d5536c0b79203c" in a "pending" state. The "Virtual Private Gateway" field contains "vgw-05db8bcc160d0d77f | NWKT-VPG-VPC01". The "Transit Gateway" field is empty. The "VPN Connection" details panel below the table shows the same information: "VPN Connection: vpn-090d5536c0b79203c". It includes tabs for "Details", "Tunnel Details", and "Tags", with the "Details" tab selected. The "Details" section displays various configuration parameters such as Customer Gateway Address (91.153.26.147), Category (VPN), Routing (Dynamic), and Authentication Type (Pre Shared Key). The bottom right corner of the screenshot shows the text "AWS VPC CH05 - F24-19-10-2021".

Figure 5-24: VPN Connection Verification.

This screenshot is identical to Figure 5-24, showing the AWS VPC console for VPN connections. The main difference is the status of the connection. In this version, the connection "NWKT-HQ-VPC01" is listed as "available" instead of "pending". All other details, including the VPN ID, Customer Gateway, and tunnel configuration, remain the same. The "VPN Connection" details panel also reflects the "available" status. The bottom right corner of the screenshot shows the text "AWS VPC CH05 - F25-19-10-2021".

Figure 5-25: VPN Connection Verification.

CGW Configuration

We can download the CGW configuration after configuring the VPN connection parameter by clicking the Download Configuration button in the VPN Connection window (Figure 5-25 on the previous page). The file includes all configurations with detailed explanations related to the VPN connection. We only need to specify our CGW's local Interface (or IP address) information related to Crypto Key Ring, ISAKMP Profile, and tunnel source address. Among these changes, we will configure BGP to advertise the subnet 172.16.10.0/24. After modifying the CGW configuration file, we can copy the configuration to CGW.

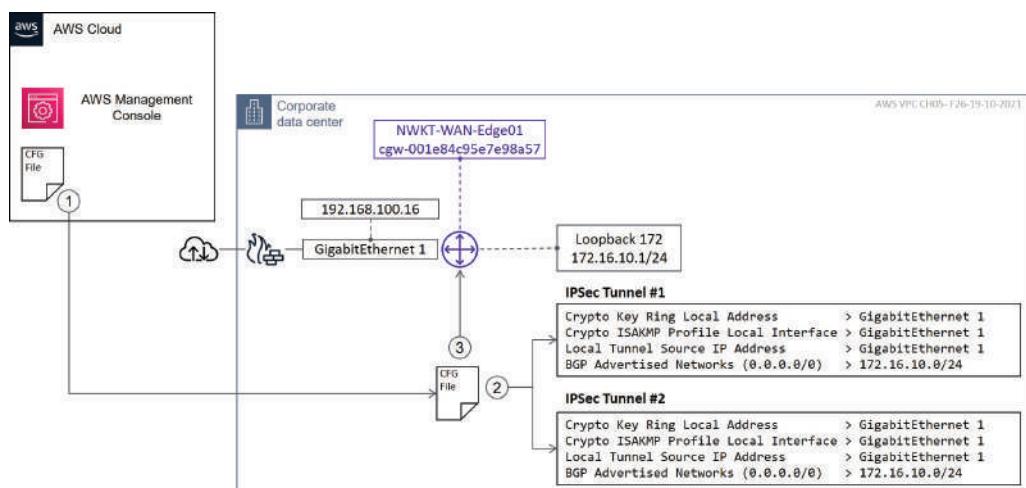


Figure 5-26: CGW Configuration.

Download CFG File

Select the Download Configuration button on a top bar on the VPN window (figure 5-25). Our Customer Gateway (CGW) is a Cisco CSR1000v (IOS-XE 16.6). Select Vendor, Platform, Software, and IKE from the respective drop-down menus. Then, click the Download button.



Figure 5-27: CGW Configuration.

Example 5-1 shows the Amazon auto-generated CGW configuration for CSR1000v based on what we specified during the VGW, CGW, and VPN connection. I have changed the font type of instructions and highlighted the configuration part.

```
! Amazon Web Services
! Virtual Private Cloud

! AWS utilizes unique identifiers to manipulate the configuration of
! a VPN Connection. Each VPN Connection is assigned an identifier and is
! associated with two other identifiers, namely the
! Customer Gateway Identifier and Virtual Private Gateway Identifier.
!
! Your VPN Connection ID      : vpn-090d5536c0b79203c
! Your Virtual Private Gateway ID : vgw-05db8bcc160d0d77f
! Your Customer Gateway ID    : cgw-001e84c95e7e98a57
!
!
! This configuration consists of two tunnels. Both tunnels must be
! configured on your Customer Gateway.
!
! You may need to populate these values throughout the config based on your setup:
! <interface_name/private_IP_on_outside_interface> - External interface of the CSR
!
! -----
! #1: Internet Key Exchange (IKE) Configuration
!
! A policy is established for the supported ISAKMP encryption,
! authentication, Diffie-Hellman, lifetime, and key parameters.
! Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and DH Group 2.
! Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group
14.
```

! You will need to modify these sample configuration files to take advantage of AES256, SHA256, or other DH groups like 2, 14-18, 22, 23, and 24.

! NOTE: If you customized tunnel options when creating or modifying your VPN connection, you may need to modify these sample configurations to match the custom settings for your tunnels.

!

! Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".

! The address of the external interface for your customer gateway must be a static address.

! Your customer gateway may reside behind a device performing network address translation (NAT).

! To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall !rules to unblock UDP port 4500.

| If not behind NAT, and you are not using an Accelerated VPN, we recommend disabling NAT-T. If you are using an Accelerated VPN, make sure that NAT-T is enabled.

!

! Note that there are a global list of ISAKMP policies, each identified by sequence number. This policy is defined as #200, which may conflict with an existing policy using the same number. If so, we recommend changing the sequence number to avoid conflicts.

!

```
crypto isakmp policy 200
    encryption aes 128
    authentication pre-share
    group 2
    lifetime 28800
    hash sha
exit
```

! The ISAKMP keyring stores the Pre Shared Key used to authenticate the tunnel endpoints.

!

```
crypto keyring keyring-vpn-090d5536c0b79203c-0
    local-address <interface_name/private_IP_on_outside_interface>
    pre-shared-key address 3.9.101.25 key I.hav3_Chang3d_This.Key4Security
exit
```

! An ISAKMP profile is used to associate the keyring with the particular endpoint.

!

```
crypto isakmp profile isakmp-vpn-090d5536c0b79203c-0
    local-address <interface_name/private_IP_on_outside_interface>
    match identity address 3.9.101.25
    keyring keyring-vpn-090d5536c0b79203c-0
exit
```

! #2: IPSec Configuration

!

! The IPSec transform set defines the encryption, authentication, and IPSec mode parameters.

! Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group 14.

! Please note, you may use these additionally supported IPSec parameters for encryption like AES256 and other DH groups like 2, 5, 14-18, 22, 23, and 24.

! Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".

!

```
crypto ipsec transform-set ipsec-prop-vpn-090d5536c0b79203c-0 esp-aes 128 esp-sha-hmac
    mode tunnel
exit
```

! The IPSec profile references the IPSec transform set and further defines the Diffie-Hellman group and security association lifetime.

!

```
crypto ipsec profile ipsec-vpn-090d5536c0b79203c-0
```

```

set pfs group2
set security-association lifetime seconds 3600
set transform-set ipsec-prop-vpn-090d5536c0b79203c-0
exit

! Additional parameters of the IPSec configuration are set here. Note that
! these parameters are global and therefore impact other IPSec
! associations.
! This option instructs the router to clear the "Don't Fragment"
! bit from packets that carry this bit and yet must be fragmented, enabling
! them to be fragmented.
!
crypto ipsec df-bit clear

! This option enables IPSec Dead Peer Detection, which causes periodic
! messages to be sent to ensure a Security Association remains operational.
! If you are using Accelerated AWS VPN, please configure periodic Dead Peer Detection.
! isakmp keepalive threshold 10 retry 10 periodic
!
crypto isakmp keepalive 10 10

! This configures the gateway's window for accepting out of order
! IPSec packets. A larger window can be helpful if too many packets
! are dropped due to reordering while in transit between gateways.
!
crypto ipsec security-association replay window-size 128

! This option instructs the router to fragment the unencrypted packets
! (prior to encryption).
!
crypto ipsec fragmentation before-encryption

! -----
#3: Tunnel Interface Configuration
!
! A tunnel interface is configured to be the logical interface associated
! with the tunnel. All traffic routed to the tunnel interface will be
! encrypted and transmitted to the VPC. Similarly, traffic from the VPC
! will be logically received on this interface.
!
! Association with the IPSec security association is done through the
!"tunnel protection" command.
!
! The address of the interface is configured with the setup for your
! Customer Gateway. If the address changes, the Customer Gateway and VPN
! Connection must be recreated with Amazon VPC.
!
interface Tunnel1
  ip address 169.254.88.242 255.255.255.252
  ip virtual-reassembly
  tunnel source <interface_name/private_IP_on_outside_interface>
  tunnel destination 3.9.101.25
  tunnel mode ipsec ipv4
  tunnel protection ipsec profile ipsec-vpn-090d5536c0b79203c-0
  ! This option causes the router to reduce the Maximum Segment Size of
  ! TCP packets to prevent packet fragmentation.
  ip tcp adjust-mss 1379
  no shutdown
exit

```

```

! -----
! #4: Border Gateway Protocol (BGP) Configuration
!
! BGP is used within the tunnel to exchange prefixes between the
! Virtual Private Gateway and your Customer Gateway. The Virtual Private Gateway
! will announce the prefix corresponding to your VPC.
!
! Your Customer Gateway may announce a default route (0.0.0.0/0),
! which can be done with the 'network' and 'default-originate' statements.
!
! The BGP timers are adjusted to provide more rapid detection of outages.
!
! The local BGP Autonomous System Number (ASN) (65123) is configured
! as part of your Customer Gateway. If the ASN must be changed, the
! Customer Gateway and VPN Connection will need to be recreated with AWS.
!
router bgp 65123
neighbor 169.254.88.241 remote-as 64512
neighbor 169.254.88.241 activate
neighbor 169.254.88.241 timers 10 30 30
address-family ipv4 unicast
neighbor 169.254.88.241 remote-as 64512
neighbor 169.254.88.241 timers 10 30 30
neighbor 169.254.88.241 default-originate
neighbor 169.254.88.241 activate
neighbor 169.254.88.241 soft-reconfiguration inbound
! To advertise additional prefixes to Amazon VPC, copy the 'network' statement
! and identify the prefix you wish to advertise. Make sure the prefix is present
! in the routing table of the device with a valid next-hop.
    network 0.0.0.0
    exit
exit
!
!
! IPSec Tunnel #2
! -----
! #1: Internet Key Exchange (IKE) Configuration
!
! A policy is established for the supported ISAKMP encryption,
! authentication, Diffie-Hellman, lifetime, and key parameters.
! Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and DH Group 2.
! Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group 14.
! You will need to modify these sample configuration files to take advantage of AES256, SHA256, or other DH groups
! like 2, 14-18, 22, 23, and 24.
! NOTE: If you customized tunnel options when creating or modifying your VPN connection, you may need to modify
these sample configurations to match the custom settings for your tunnels.
!
! Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".
! The address of the external interface for your customer gateway must be a static address.
! Your customer gateway may reside behind a device performing network address translation (NAT).
! To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall rules to unlock UDP port 4500.
! If not behind NAT, and you are not using an Accelerated VPN, we recommend disabling NAT-T. If you are using an
Accelerated VPN, make sure that NAT-T is enabled.
!
! Note that there are a global list of ISAKMP policies, each identified by
! sequence number. This policy is defined as #201, which may conflict with
! an existing policy using the same number. If so, we recommend changing
! the sequence number to avoid conflicts.

```

```
!
crypto isakmp policy 201
  encryption aes 128
  authentication pre-share
  group 2
  lifetime 28800
  hash sha
exit

! The ISAKMP keyring stores the Pre Shared Key used to authenticate the
! tunnel endpoints.
!
crypto keyring keyring-vpn-090d5536c0b79203c-1
  local-address <interface_name/private_IP_on_outside_interface>
  pre-shared-key address 35.177.110.203 key 1.have_Changed_This.K3y45ecurity
exit
```

```
! An ISAKMP profile is used to associate the keyring with the particular
! endpoint.
!
crypto isakmp profile isakmp-vpn-090d5536c0b79203c-1
  local-address <interface_name/private_IP_on_outside_interface>
  match identity address 35.177.110.203
  keyring keyring-vpn-090d5536c0b79203c-1
exit
```

! #2: IPsec Configuration

```
!
! The IPsec transform set defines the encryption, authentication, and IPsec
! mode parameters.
! Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group
14.
! Please note, you may use these additionally supported IPsec parameters for encryption like AES256 and other DH
groups like 2, 5, 14-18, 22, 23, and 24.
! Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".
!
```

```
crypto ipsec transform-set ipsec-prop-vpn-090d5536c0b79203c-1 esp-aes 128 esp-sha-hmac
  mode tunnel
exit
```

```
! The IPsec profile references the IPsec transform set and further defines
! the Diffie-Hellman group and security association lifetime.
!
```

```
crypto ipsec profile ipsec-vpn-090d5536c0b79203c-1
  set pfs group2
  set security-association lifetime seconds 3600
  set transform-set ipsec-prop-vpn-090d5536c0b79203c-1
exit
```

```
! Additional parameters of the IPsec configuration are set here. Note that
! these parameters are global and therefore impact other IPsec
! associations.
```

```
! This option instructs the router to clear the "Don't Fragment"
! bit from packets that carry this bit and yet must be fragmented, enabling
! them to be fragmented.
!
```

```
crypto ipsec df-bit clear
```

```
! This option enables IPsec Dead Peer Detection, which causes periodic
! messages to be sent to ensure a Security Association remains operational.
! If you are using Accelerated AWS VPN, please configure periodic Dead Peer Detection.
```

```

! isakmp keepalive threshold 10 retry 10 periodic
!
crypto isakmp keepalive 10 10

! This configures the gateway's window for accepting out of order
! IPSec packets. A larger window can be helpful if too many packets
! are dropped due to reordering while in transit between gateways.
!
crypto ipsec security-association replay window-size 128

! This option instructs the router to fragment the unencrypted packets
! (prior to encryption).
!
crypto ipsec fragmentation before-encryption

! -----
#3: Tunnel Interface Configuration
!

! A tunnel interface is configured to be the logical interface associated
! with the tunnel. All traffic routed to the tunnel interface will be
! encrypted and transmitted to the VPC. Similarly, traffic from the VPC
! will be logically received on this interface.
!
! Association with the IPSec security association is done through the
! "tunnel protection" command.
!
! The address of the interface is configured with the setup for your
! Customer Gateway. If the address changes, the Customer Gateway and VPN
! Connection must be recreated with Amazon VPC.
!
interface Tunnel1
    ip address 169.254.159.58 255.255.255.252
    ip virtual-reassembly
    tunnel source <interface_name/private_IP_on_outside_interface>
    tunnel destination 35.177.110.203
    tunnel mode ipsec ipv4
    tunnel protection ipsec profile ipsec-vpn-090d5536c0b79203c-1
        ! This option causes the router to reduce the Maximum Segment Size of
        ! TCP packets to prevent packet fragmentation.
        ip tcp adjust-mss 1379
    no shutdown
exit

! -----
#4: Border Gateway Protocol (BGP) Configuration
!

! BGP is used within the tunnel to exchange prefixes between the
! Virtual Private Gateway and your Customer Gateway. The Virtual Private Gateway
! will announce the prefix corresponding to your VPC.
!
! Your Customer Gateway may announce a default route (0.0.0.0/0),
! which can be done with the 'network' and 'default-originate' statements.
!
! The BGP timers are adjusted to provide more rapid detection of outages.
!
! The local BGP Autonomous System Number (ASN) (65123) is configured
! as part of your Customer Gateway. If the ASN must be changed, the
! Customer Gateway and VPN Connection will need to be recreated with AWS.

```

```

!
router bgp 65123
  neighbor 169.254.159.57 remote-as 64512
  neighbor 169.254.159.57 activate
  neighbor 169.254.159.57 timers 10 30 30
  address-family ipv4 unicast
    neighbor 169.254.159.57 remote-as 64512
    neighbor 169.254.159.57 timers 10 30 30
    neighbor 169.254.159.57 default-originate
    neighbor 169.254.159.57 activate
    neighbor 169.254.159.57 soft-reconfiguration inbound
! To advertise additional prefixes to Amazon VPC, copy the 'network' statement
! and identify the prefix you wish to advertise. Make sure the prefix is present
! in the routing table of the device with a valid next-hop.
  network 0.0.0.0
exit
exit
!

! Additional Notes and Questions
! - Amazon Virtual Private Cloud Getting Started Guide:
!   http://docs.amazonwebservices.com/AmazonVPC/latest/GettingStartedGuide
! - Amazon Virtual Private Cloud Network Administrator Guide:
!   http://docs.amazonwebservices.com/AmazonVPC/latest/NetworkAdminGuide

```

Example 5-1: CGW Configuration File.

Configure CGW Device

Example 5-2 shows the CSR1000v configuration.

```

NWKT-WAN-Edge01#sh run
Building configuration...

<snipped>
!
hostname NWKT-WAN-Edge01
!
<snipped>
!
crypto pki trustpoint TP-self-signed-1949931239
enrollment selfsigned
subject-name cn=IOS-Self-Signed-Certificate-1949931239
revocation-check none
rsakeypair TP-self-signed-1949931239
!
!
crypto pki certificate chain TP-self-signed-1949931239
!
<snipped>
!
crypto keyring keyring-vpn-090d5536c0b79203c-1
  local-address GigabitEthernet1
  pre-shared-key address 35.177.110.203 key I.hav3_Chang3d_This.Key4Security
crypto keyring keyring-vpn-090d5536c0b79203c-0
  local-address GigabitEthernet1
  pre-shared-key address 3.9.101.25 key I.hav3_Chang3d_This.Key4Security
!
crypto isakmp policy 200

```

```
encr aes
authentication pre-share
group 2
lifetime 28800
!
crypto isakmp policy 201
encr aes
authentication pre-share
group 2
lifetime 28800
crypto isakmp keepalive 10 10
crypto isakmp profile isakmp-vpn-090d5536c0b79203c-0
    keyring keyring-vpn-090d5536c0b79203c-0
    match identity address 3.9.101.25 255.255.255.255
    local-address GigabitEthernet1
crypto isakmp profile isakmp-vpn-090d5536c0b79203c-1
    keyring keyring-vpn-090d5536c0b79203c-1
    match identity address 35.177.110.203 255.255.255.255
    local-address GigabitEthernet1
!
crypto ipsec security-association replay window-size 128
!
crypto ipsec transform-set ipsec-prop-vpn-090d5536c0b79203c-0 esp-aes esp-sha-hmac
mode tunnel
crypto ipsec transform-set ipsec-prop-vpn-090d5536c0b79203c-1 esp-aes esp-sha-hmac
mode tunnel
crypto ipsec df-bit clear
!
!
crypto ipsec profile ipsec-vpn-090d5536c0b79203c-0
set transform-set ipsec-prop-vpn-090d5536c0b79203c-0
set pfs group2
!
crypto ipsec profile ipsec-vpn-090d5536c0b79203c-1
set transform-set ipsec-prop-vpn-090d5536c0b79203c-1
set pfs group2
!
interface Loopback172
ip address 172.16.10.1 255.255.255.0
!
interface Tunnel1
ip address 169.254.88.242 255.255.255.252
ip tcp adjust-mss 1379
tunnel source GigabitEthernet1
tunnel mode ipsec ipv4
tunnel destination 3.9.101.25
tunnel protection ipsec profile ipsec-vpn-090d5536c0b79203c-0
ip virtual-reassembly
!
interface Tunnel2
ip address 169.254.159.58 255.255.255.252
ip tcp adjust-mss 1379
tunnel source GigabitEthernet1
tunnel mode ipsec ipv4
tunnel destination 35.177.110.203
tunnel protection ipsec profile ipsec-vpn-090d5536c0b79203c-1
ip virtual-reassembly
!
!
router bgp 65123
bgp log-neighbor-changes
neighbor 169.254.88.241 remote-as 64512
```

```

neighbor 169.254.88.241 timers 10 30 30
neighbor 169.254.159.57 remote-as 64512
neighbor 169.254.159.57 timers 10 30 30
!
address-family ipv4
  network 172.16.10.0 mask 255.255.255.0
  neighbor 169.254.88.241 activate
  neighbor 169.254.88.241 default-originate
  neighbor 169.254.88.241 soft-reconfiguration inbound
  neighbor 169.254.159.57 activate
  neighbor 169.254.159.57 default-originate
  neighbor 169.254.159.57 soft-reconfiguration inbound
exit-address-family
!
!
NWKT-WAN-Edge01#

```

Example 5-2: CGW NWKT-WAN-Edge01 Configuration.

Tunnel Verification

Example 5-3 shows that both tunnels are up.

Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet1	192.168.100.18	YES	DHCP	up	up
GigabitEthernet2	unassigned	YES	NVRAM	administratively down	down
GigabitEthernet3	10.1.3.1	YES	NVRAM	up	up
GigabitEthernet4	unassigned	YES	NVRAM	administratively down	down
Loopback0	192.168.10.1	YES	NVRAM	up	up
Loopback172	172.16.10.1	YES	NVRAM	up	up
Tunnel1	169.254.88.242	YES	manual	up	up
Tunnel2	169.254.159.58	YES	manual	up	up

Example 5-3: Show IP interface Brief.

Example 5-4 shows detailed information about Tunnel 1. We can see that the destination IP address is 3.9.101.25, and we have bound the IPSec profile ipsec-vpn-090d5536c0b79203c-0, which defines the encryption and authentication algorithms used with this tunnel interface.

```

NWKT-WAN-Edge01#show interfaces tunnel 1
Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 169.254.88.242/30
  MTU 9922 bytes, BW 100 Kbit/sec, DLY 50000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel linestate evaluation up
  Tunnel source 192.168.100.18 (GigabitEthernet1), destination 3.9.101.25
  Tunnel Subblocks:
    src-track:
      Tunnel1 source tracking subblock associated with GigabitEthernet1

```

```

        Set of tunnels with source GigabitEthernet1, 1 member (includes
iterators), on interface <OK>
Tunnel protocol/transport IPSEC/IP
Tunnel TTL 255
Tunnel transport MTU 1422 bytes
Tunnel transmit bandwidth 8000 (kbps)
Tunnel receive bandwidth 8000 (kbps)
Tunnel protection via IPSec (profile "ipsec-vpn-090d5536c0b79203c-0")
Last input never, output never, output hang never
Last clearing of "show interface" counters 00:13:41
Input queue: 0/375/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/0 (size/max)
<interface counters snipped for brevity>
NWKT-WAN-Edge01#

```

Example 5-4: Show Interface Tunnel 1.

Example 5-5 shows detailed information about Tunnel 2. We can see that the destination IP address is 35.177.110.203, and we have bound the IPSec profile ipsec-vpn-090d5536c0b79203c-1, which defines the encryption and authentication algorithms used with this tunnel interface.

```

NWKT-WAN-Edge01#sh int tunnel 2
Tunnel2 is up, line protocol is up
Hardware is Tunnel
Internet address is 169.254.159.58/30
MTU 9922 bytes, BW 100 Kbit/sec, DLY 50000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
Encapsulation TUNNEL, loopback not set
Keepalive not set
Tunnel linestate evaluation up
Tunnel source 192.168.100.18 (GigabitEthernet1), destination 35.177.110.203
Tunnel Subblocks:
    src-track:
        Tunnel2 source tracking subblock associated with GigabitEthernet1
        Set of tunnels with source GigabitEthernet1, 2 members (includes
iterators), on interface <OK>
        Tunnel protocol/transport IPSEC/IP
        Tunnel TTL 255
        Tunnel transport MTU 1422 bytes
        Tunnel transmit bandwidth 8000 (kbps)
        Tunnel receive bandwidth 8000 (kbps)
        Tunnel protection via IPSec (profile "ipsec-vpn-090d5536c0b79203c-1")
        Last input never, output never, output hang never
        Last clearing of "show interface" counters 00:09:27
        Input queue: 0/375/0/0 (size/max/drops/flushes); Total output drops: 0
        Queueing strategy: fifo
        Output queue: 0/0 (size/max)
        <interface counters snipped for brevity>

```

Example 5-5: Show Interface Tunnel 2.

Example 5-6 shows that we have attached IPSec transform set ipsec-prop-vpn-090d5536c0b79203c-0 to IPSec profile ipsec-vpn-090d5536c0b79203c-0 (attached to Tunnel 1). It also shows that we have attached IPSec transform set ipsec-prop-vpn-090d5536c0b79203c-1 to IPSec profile ipsec-vpn-090d5536c0b79203c-2 (attached to Tunnel 2).

```
NWKT-WAN-Edge01#show crypto ipsec profile
IPSEC profile default
    Security association lifetime: 4608000 kilobytes/3600 seconds
    Responder-Only (Y/N): N
    PFS (Y/N): N
    Mixed-mode : Disabled
    Transform sets={
        default: { esp-aes esp-sha-hmac } ,
    }

IPSEC profile ipsec-vpn-090d5536c0b79203c-0
    Security association lifetime: 4608000 kilobytes/3600 seconds
    Responder-Only (Y/N): N
    PFS (Y/N): Y
    DH group: group2
    Mixed-mode : Disabled
    Transform sets={
        ipsec-prop-vpn-090d5536c0b79203c-0: { esp-aes esp-sha-hmac }
    }

IPSEC profile ipsec-vpn-090d5536c0b79203c-1
    Security association lifetime: 4608000 kilobytes/3600 seconds
    Responder-Only (Y/N): N
    PFS (Y/N): Y
    DH group: group2
    Mixed-mode : Disabled
    Transform sets={
        ipsec-prop-vpn-090d5536c0b79203c-1: { esp-aes esp-sha-hmac }
    }
```

Example 5-6: Show Crypto Ipsec Profile.

Example 5-7 shows that both ipsec transform-sets ipsec-prop-vpn-090d5536c0b79203c-0 and ipsec-prop-vpn-090d5536c0b79203c-1 use the esp-aes encryprion algorithm and esp-sha-hmac hashing algorithm for authentication.

```
NWKT-WAN-Edge01#show crypto ipsec transform-set
Transform set default: { esp-aes esp-sha-hmac }
    will negotiate = { Transport, },
Transform set ipsec-prop-vpn-090d5536c0b79203c-0: { esp-aes esp-sha-hmac }
    will negotiate = { Tunnel, },
Transform set ipsec-prop-vpn-090d5536c0b79203c-1: { esp-aes esp-sha-hmac }
    will negotiate = { Tunnel, },
```

Example 5-7: Show Crypto Ipsec Transform-set.

Example 5-8 shows our ISAKMP Profiles. The first profile is bound to local interface GigabitEthernet 1 and remote peer 3.9.101.25/32 (Tunnel 1). Key Ring keyring-vpn-090d5536c0b79203c-0 defines the pre-share key we are using with this remote peer. The second profile is also bound to the local interface GigabitEthernet 1 but to remote IP address 35.177.110.203/32 (Tunnel 2). Key Ring keyring-vpn-090d5536c0b79203c-1 defines the pre-share key we are using with this remote peer.

```
NWKT-WAN-Edge01#show crypto isakmp profile
IKEv1 PROFILE isakmp-vpn-090d5536c0b79203c-0
Ref Count = 2
    Identities matched are:
        ip-address 3.9.101.25 255.255.255.255
    Certificate maps matched are:
        keyring(s): keyring-vpn-090d5536c0b79203c-0
        trustpoint(s): <all>
    Interface binding: GigabitEthernet1 :

IKEv1 PROFILE isakmp-vpn-090d5536c0b79203c-1
Ref Count = 2
    Identities matched are:
        ip-address 35.177.110.203 255.255.255.255
    Certificate maps matched are:
        keyring(s): keyring-vpn-090d5536c0b79203c-1
        trustpoint(s): <all>
    Interface binding: GigabitEthernet1 :
```

Example 5-8: Show Crypto ISAKMP Profile.

Figure 5-28 shows that both tunnels used for VPN NWKT-HQ-VPC01 are up on the VGW.

Tunnel Number	Outside IP Address	Inside IPv4 CIDR	Inside IPv6 CIDR	Status	Status Last Changed	Details
Tunnel 1	3.9.101.25	169.254.88.240/30	-	UP	October 18, 2021 at 1:50:33 P...	2 BGP ROUTES
Tunnel 2	35.177.110.203	169.254.159.56/30	-	UP	October 18, 2021 at 1:50:26 P...	2 BGP ROUTES

Figure 5-28: Tunnel details on VGW.

Control-Plane Verification

Example 5-9 shows that NWKT-WAN-Edge01 has two BGP peers. It has received one prefix from both peers.

```
NWKT-WAN-Edge01#show ip bgp summary
BGP router identifier 192.168.10.1, local AS number 65123
BGP table version is 7, main routing table version 7
<snipped for brevity>
BGP activity 3/0 prefixes, 4/0 paths, scan interval 60 secs

Neighbor      V      AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down State/PfxRcd
169.254.88.241  4      64512    498     527       7     0     0 01:22:24        1
169.254.159.57  4      64512     6      8       7     0     0 00:00:24        1
```

Example 5-9: Show IP BGP Summary.

Example 5-10 shows the BGP table of NWKT-WAN-Edge01. It verifies that it has chosen a route via 169.254.88.241 as the best path due to the lower metric value (100). That illustrates that VGW uses metric 100 (carried within MED path attribute) to BGP updates sent over Tunnel 1 and metric 200 to BGP updates sent over Tunnel 2. To avoid asymmetric routing, we can use the same BGP egress policy signaling by setting the lower MED for BGP Updates sent over Tunnel 1 and higher metric for BGP Updates sent over Tunnel 2. We can also use BGP AS-Path prepending for BGP Updates sent over Tunnel 2. Note that MED and AS-Path prepending is the only option for BGP traffic engineering that VGW accepts.

```
NWKT-WAN-Edge01#show ip bgp
BGP table version is 6, local router ID is 192.168.10.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop            Metric LocPrf Weight Path
        0.0.0.0          0.0.0.0                  0  i
*>  10.10.0.0/16    169.254.88.241       100   0 64512 i
*   10.10.0.0/16    169.254.159.57       200   0 64512 i
*>  172.16.10.0/24  0.0.0.0                  0   32768 i
```

Example 5-10: Show IP BGP.

Example 5-11 shows that NWKT-WAN-Edge01 has installed only the best BGP route from the BGP table to the routing table. BGP also supports flow-based ECMP (Equal Cost Multi-Path), and we can implement it on CGW CSR1000v. However, AWS VGW doesn't use ECMP. That is why we don't use BGP ECMP on CGW either.

```
NWKT-WAN-Edge01#show ip route 10.10.0.218
Routing entry for 10.10.0.0/16
  Known via "bgp 65123", distance 20, metric 100
  Tag 64512, type external
  Last update from 169.254.88.241 01:24:48 ago
  Routing Descriptor Blocks:
    * 169.254.88.241, from 169.254.88.241, 01:24:48 ago
      Route metric is 100, traffic share count is 1
      AS Hops 1
      Route tag 64512
      MPLS label: none
```

Example 5-11: Show IP BGP.

Figure 5-29 shows verifies that the Route Table NWKT-PUB-RT has received route to 172.16.10.0/24 from the VGW.

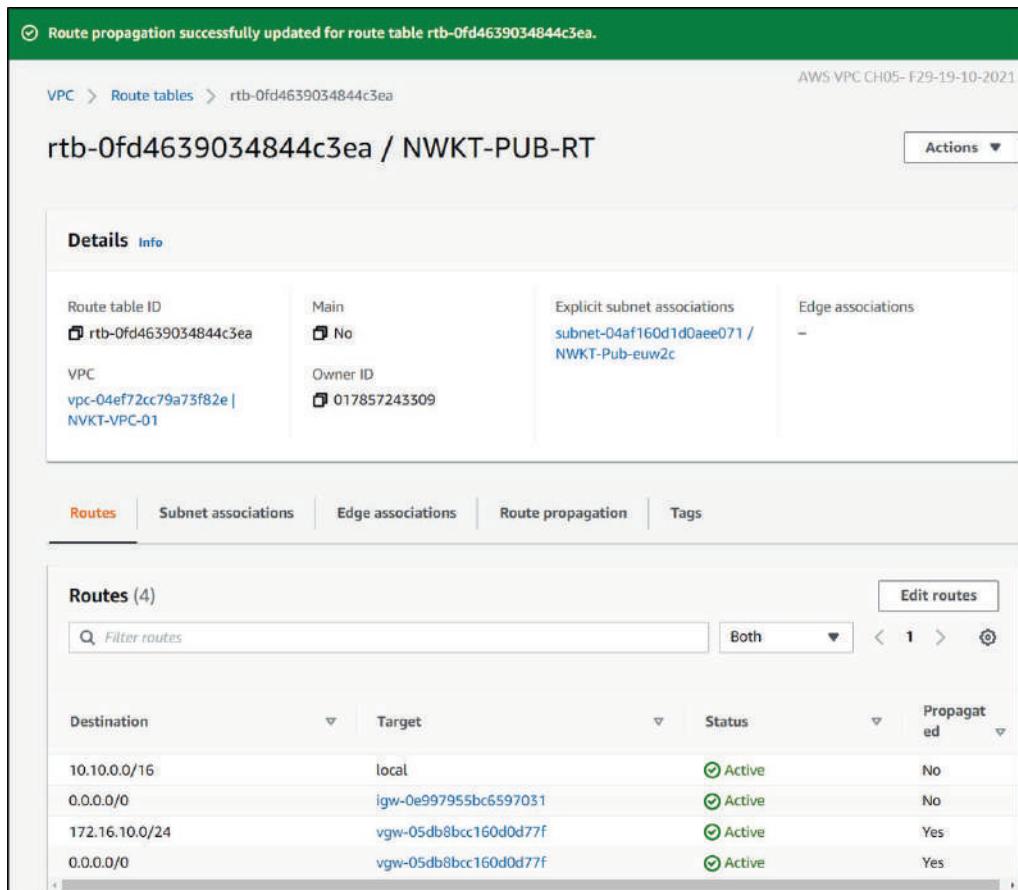


Figure 5-29: CGW Configuration.

Data-Plane Verification

Example 5-12 verifies that the Data-Plane is working. We can ping from CGW to EC2 instance 10.10.0.218 in VPC using the IP address 172.16.10.1 (loopback 172) as a source IP.

```
NWKT-WAN-Edge01#ping 10.10.0.218 source loopback 172
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.0.218, timeout is 2 seconds:
Packet sent with a source address of 172.16.10.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 35/35/36 ms
```

Example 5-12: Data-Plane Verification.

Billing

Figure 5-30 summarizes the costs so far. We started the first billable component (Linux t2.micro instance with EIP) 285 hours ago (11 days). We also use NAT GW for Internet access to the t2.micro launched on private subnets (I have deleted those later). We have also run an AWS Reachability Analyzer four times. There is also a small fee for a VPN connection. Note that our data transfer is based on a couple of ssh connection and ICMP request/reply. That is why there are no data transfer fees on the bill. The total cost so far is only 8.83\$, which in my mind is a reasonable amount.

AWS Service Charges		\$8.83
▼ Data Transfer		\$0.00
▶ EU (London)		\$0.00
▼ Elastic Compute Cloud		\$4.21
▶ EU (London)		\$4.21
Amazon Elastic Compute Cloud NatGateway		\$0.10
\$0.05 per GB Data Processed by NAT Gateways	0.000001 GB	\$0.00
\$0.05 per NAT Gateway Hour	2.000 Hrs	\$0.10
Amazon Elastic Compute Cloud running Linux/UNIX		\$3.76
\$0.0132 per On Demand Linux t2.micro Instance Hour	285.179 Hrs	\$3.76
EBS		\$0.34
\$0.116 per GB-month of General Purpose SSD (gp2) provisioned storage - EU (London)	2.970 GB-Mo	\$0.34
▼ Key Management Service		\$0.00
▶ EU (London)		\$0.00
AWS Key Management Service eu-west-2-KMS-Requests		\$0.00
\$0.03 per 10000 KMS requests in EU (London)	1.000 Requests	\$0.00
▼ Virtual Private Cloud		\$2.95
▶ EU (London)		\$2.95
No Instance Type		\$2.95
\$0.05 per VPN Connection-Hour	51.000 Hrs	\$2.55
\$0.1 per analysis processed by VPC Reachability Analyzer	4.000 ReachabilityAnalyses	\$0.40
Taxes		
VAT to be collected		\$1.67

Figure 5-30: Billing Information.

Chapter 6: Transit Gateway

Introduction

Chapter 5 discusses AWS to On-Prem DC VPN connection with Virtual Private Gateway (VGW). VGW consists of two devices on which we can build a dedicated VPN tunnel from the on-prem Customer Gateway (CGW). If we use two CGWs for high availability, we will have four IPSec tunnels per VPC. This solution doesn't scale well with a multi-VPC design where we have more than two VPCs. For example, ten VPCs solutions with a dual-homed on-prem VPN connection require 40 IPSec tunnels. Other than that, we most probably use dynamic routing, and we need to establish 40 BGP peering between VGW and CGW. It will soon become an operational nightmare. The other additional complexity with the VGW solution is that it does not provide an inter-VPC connection, eventually VGW is a VPC-specific solution. It means that to allow intra-VPC traffic flows, we need to implement VPC peering or use AWS Private-Link. That, in turn, increases to the solution's overall complexity. Note that we can use VGW as Cloud Hub for on-prem Datacenters.

AWS Transit Gateway (TGW) simplifies the overall complexity. It is a distributed, AWS region-specific system, managed by Amazon. We can connect regional VPCs to TGW using attachments and associate those with TGW Main Route Table or with dedicated Route Table. Using dedicated TGW RT, we can group VPCs with a uniform routing policy into the same RT. For example, there might be VPCs which only should access to on-prem DC resources. TGW also simplifies on-prem connection. We can implement dual-homed on-prem CGWs and establish VPN connection with a single regional TGW no matter how many VPCs we have in region. TGW also supports direct connect. Besides, we can build a global Inter-Region Backbone by establishing a peering relationship between TGWs located in geolocation Regions.

Figure 6-1 illustrates our example Transit Gateway implementation. We have three VPCs, with each having one subnet. The subnet in the left-most VPC, NWKT-Prod, is a public subnet allowing instances to use Internet Gateway for Internet access. Both subnets 10.11.0.0/24 and 10.12.0.0/24 in VPCs NWKT-Test and NWKT-Dev are private subnets without Internet access. We are going to implement Transit Gateway into AWS and attach all three VPCs to it. We will also establish a VPN connection between TGW and CGW to allow on-prem Datacenter connection to EC2 instances on each VPC.

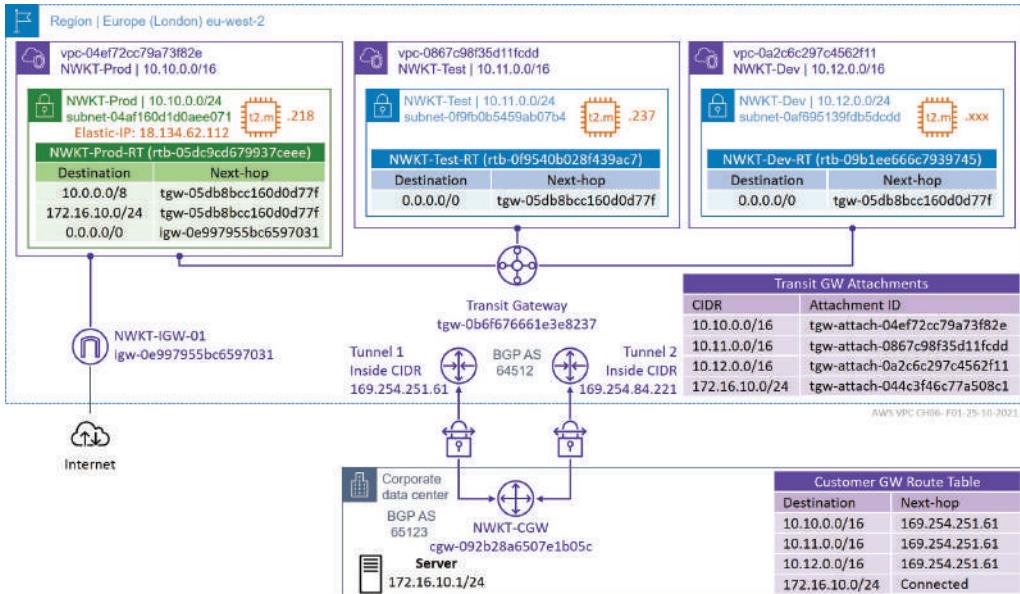


Figure 6-1: Transit Gateway Example Solution.

You can have five TGWs per account (adjustable) and five TGWs per VPC. A TGW can have 5000 attachments by default (adjustable). The maximum bandwidth per VPC is 50 Gbps (non-adjustable). One VPN attachment support 5 000 000 packets per second. The maximum bandwidth per VPN Tunnel is 1,25 Gbps with up to 140 000 packets per second. TGW supports ECMP over VPN, which gives you 2,5 Gbps per VPN and 280 000 packets per second.

Create Transit Gateway

First, we launch a Transit Gateway (1). Then we create VPC attachments for all three VPCs (2). Third, static routes to subnet-specific Route Tables on each VPCs (3). The Next-Hop (NH) for a default route is TGW on subnets NWKT-Test 10.11.0.0/24 and NWKT-Prod 10.10.0.0/24, while in subnet NWKT-Prod 10.10.0.0/24 Next-hop is IGW. The Next-Hop for the destination network 172.16.10.0/24 (on-prem DC subnet) in subnet NWKT-Prod 10.10.0.0/24 is TGW. This routing policy allows both Inter-VPC and VPC to on-prem DC communication.

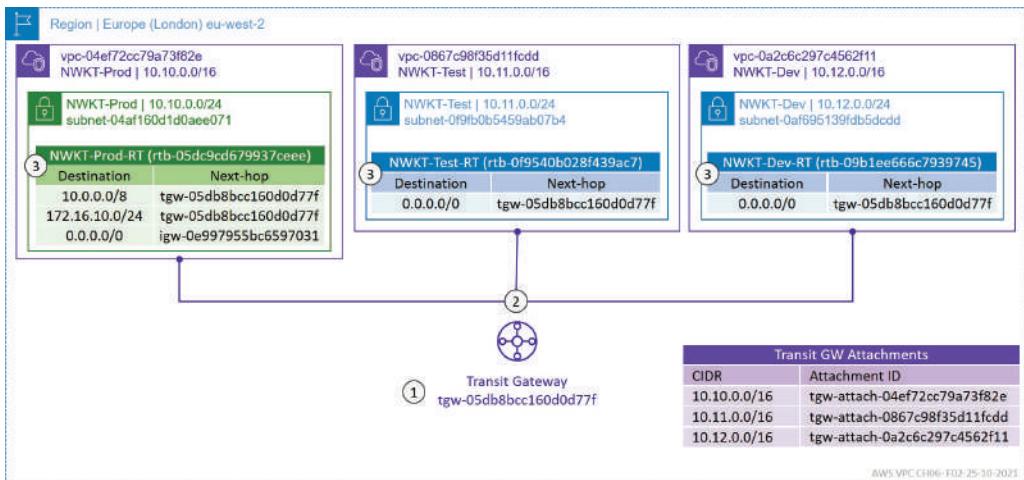


Figure 6-2: Creating Transit Gateway.

Launch TGW

Navigate to the VPC Dashboard and select Transit Gateway.

Note: Your Instances will launch in the Europe region. AWS VPC CH06-F03-25-10-2023

Resources by Region C Refresh Resources

You are using the following Amazon VPC resources

VPCs	Europe 4	NAT Gateways	Europe 0
Subnets	Europe 6	VPC Peering Connections	Europe 0
Route Tables	Europe 9	Network ACLs	Europe 4
Internet Gateways	Europe 2	Security Groups	Europe 5
Egress-only Internet Gateways	Europe 0	Customer	0

Figure 6-3: Creating Transit Gateway: Phase-1.

Click the *Create transit gateway* button.

Name	Transit gateway ID	Owner ID	State
No transit gateways found			

AWS VPC CH06-F04-25-10-2023

Figure 6-4: Creating Transit Gateway: Phase-2.

Figures 6-5 and 6-6 are in the same view in the AWS Console. Fill in the Name tag field. You can select the BGP ASN number by yourself. In this example, we are letting Amazon assign its default ASN (64512). Leave all other settings and selections to their defaults. Click the *Create transit gateway* button.

Create transit gateway Info

A transit gateway (TGW) is a network transit hub that interconnects attachments (VPCs and VPNs) within the same AWS account or across AWS accounts.

Details - optional

Name tag Info
Creates a tag with the key set to Name and the value set to the specified string.

NWKT-TGW

Description Info
Set the description of your transit gateway to help you identify it in the future.

description

Configure the transit gateway

Amazon side Autonomous System Number (ASN) Info

ASN

DNS support Info

VPN ECMP support Info

Default route table association Info

Default route table propagation Info

Multicast support Info

AWS VPC CH06- F05-25-10-2021

Figure 6-5: Creating Transit Gateway: Phase-3.1.

Configure cross-account sharing options

Auto accept shared attachments [Info](#)

Transit gateway CIDR blocks

CIDR - optional [Info](#)

10.0.0.0/24

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.

Key	Value - optional
<input type="text"/> Name <input type="button" value="X"/>	<input type="text"/> NWKT-TGW <input type="button" value="X"/> <input type="button" value="Remove"/>

You can add 49 more tags.

AWS VPC CH06-F06-25-10-2021

Figure 6-6: Creating Transit Gateway: Phase-3.2.

AWS Console notifies you about successful TGW creation. The state is first Pending (figure 6-7). When the TGW is up and running, the stage changes to Available (figure 6-8).

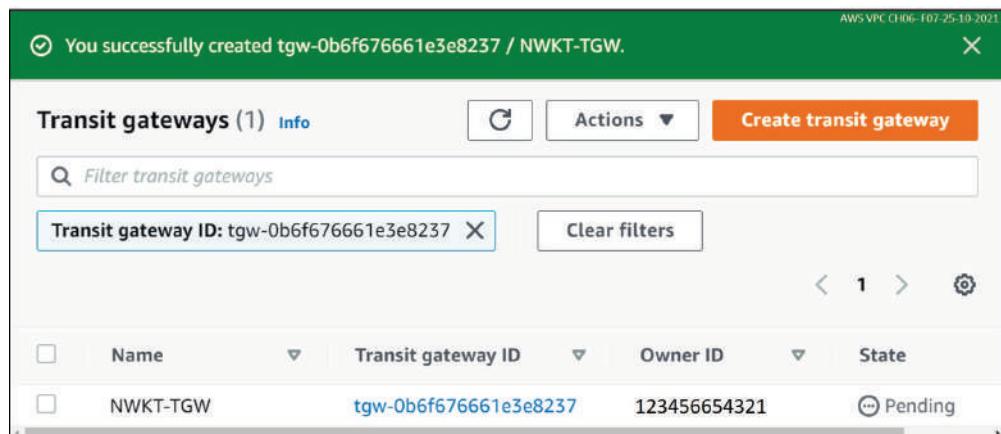


Figure 6-7: Creating Transit Gateway: Phase-3.3.

The screenshot shows the AWS VPC console with a success message at the top: "You successfully created tgw-0b6f676661e3e8237 / NWKT-TGW." Below this, the "Transit gateways (1/1)" list is displayed. The table has columns: Name, Transit gateway ID, Owner ID, and State. One entry is shown: "NWKT-TGW" with ID "tgw-0b6f676661e3e8237", Owner ID "123456654321", and State "Available". A "Create transit gateway" button is visible at the top right. Below the table, there are tabs for "Details", "Sharing", and "Tags". The "Details" tab is selected, showing the following configuration:

Transit gateway ID	<input type="checkbox"/> tgw-0b6f676661e3e8237	State	<input checked="" type="checkbox"/> Available	Amazon ASN	<input type="checkbox"/> 64512	DNS support	Enable
Transit gateway ARN	<input type="checkbox"/> arn:aws:ec2:eu-west-2:017857243309:transit-gateway/tgw-0b6f676661e3e8237	Default association route table	<input type="checkbox"/> Enable	Association route table ID	tgw-rtb-0a53e2ba10b858708	Auto accept shared attachments	Disable
		Default propagation route table	<input type="checkbox"/> Enable	Propagation route table ID	tgw-rtb-0a53e2ba10b858708	VPN ECMP support	Enable
Owner ID		Transit gateway CIDR					

Figure 6-8: Creating Transit Gateway: Phase-3.4.

Create Transit Gateway Attachment

Navigate back to the VPC Dashboard and select the Transit Gateway Attachment hyperlink. Then select the Create transit gateway attachment button.

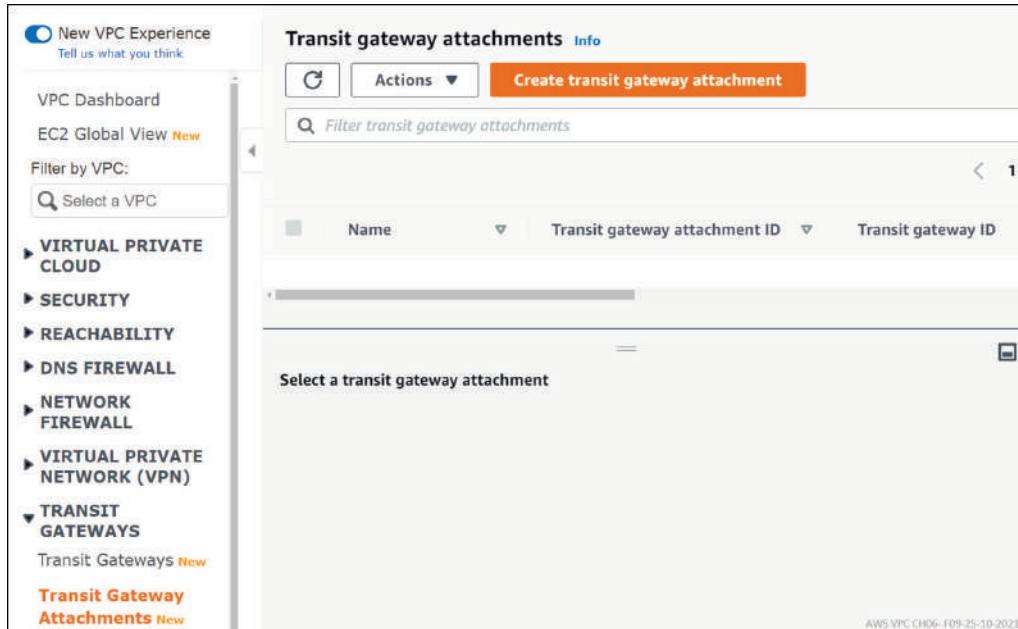


Figure 6-9: Creating Transit Gateway Attachment: Phase-1.

Figures 6-10, 6-11, and 6-12 are in the same console view. Fill in the optional Name tag field. Then select our previously created Transit Gateway NWKT-TGW (tgw-0b6f676661e3e8237) from the Transit gateway ID drop-down menu (figure 6-9) and select VPC as an attachment type. Select VPC NWKT-Prod (vpc-04ef72cc79a73f82e) from the VPC ID drop-down menu. Connect TGW to the Availability Zone (AZ) eu-west-2c and select the only available subnet NWKT-Prod (subnet-04af160d1d0aee071). Then click the *Create transit gateway attachment* button. Do the same association process also for VPC NWKT-Test and NWKT-Dev.

Create transit gateway attachment [Info](#)

A transit gateway (TGW) is a network transit hub that interconnects attachments (VPCs and VPNs) within the same AWS account or across AWS accounts.

Details

Name tag - *optional*
Creates a tag with the key set to Name and the value set to the specified string.

Transit gateway ID [Info](#)

Attachment type [Info](#)

AWS VPC CH06- F10-25-10-2021

Figure 6-10: Creating Transit Gateway Attachment: Phase-1.1.

VPC attachment

Select and configure your VPC attachment.

DNS support [Info](#)

IPv6 support [Info](#)

VPC ID
Select the VPC to attach to the transit gateway.

Subnet IDs [Info](#)
Select the subnets in which to create the transit gateway VPC attachment.

<input type="checkbox"/> eu-west-2a	No subnet available
<input type="checkbox"/> eu-west-2b	No subnet available
<input checked="" type="checkbox"/> eu-west-2c	<input type="text" value="subnet-04af160d1d0aee071 (NWKT-Prod)"/>

X

AWS VPC CH06- F11-25-10-2021

Figure 6-11: Creating Transit Gateway Attachment: Phase-1.2.

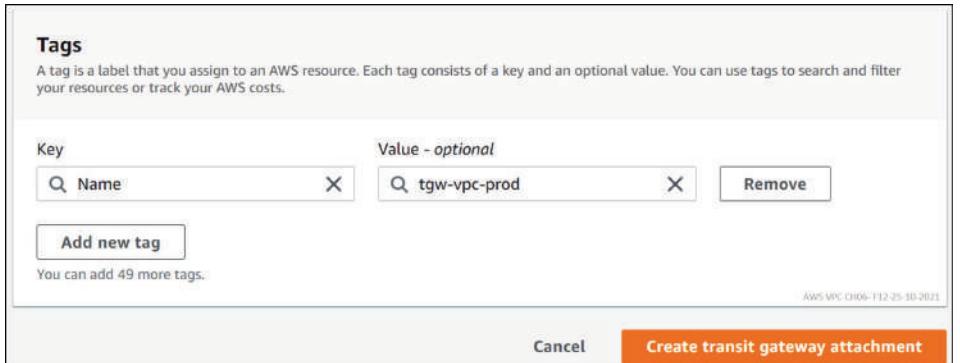


Figure 6-12: Creating Transit Gateway Attachment: Phase-1.3.

Figure 6-13 verifies that we have successfully attached VPC NWKT-Prod (vpc-04ef72cc79a73f82e) to NWKT-TGW (tgw-0b6f676661e3e8237) by using the subnet NWKT-Prod (subnet-04af160d1d0aee071) as an attachment point. We also attach two other VPC to TGW by using the same method.

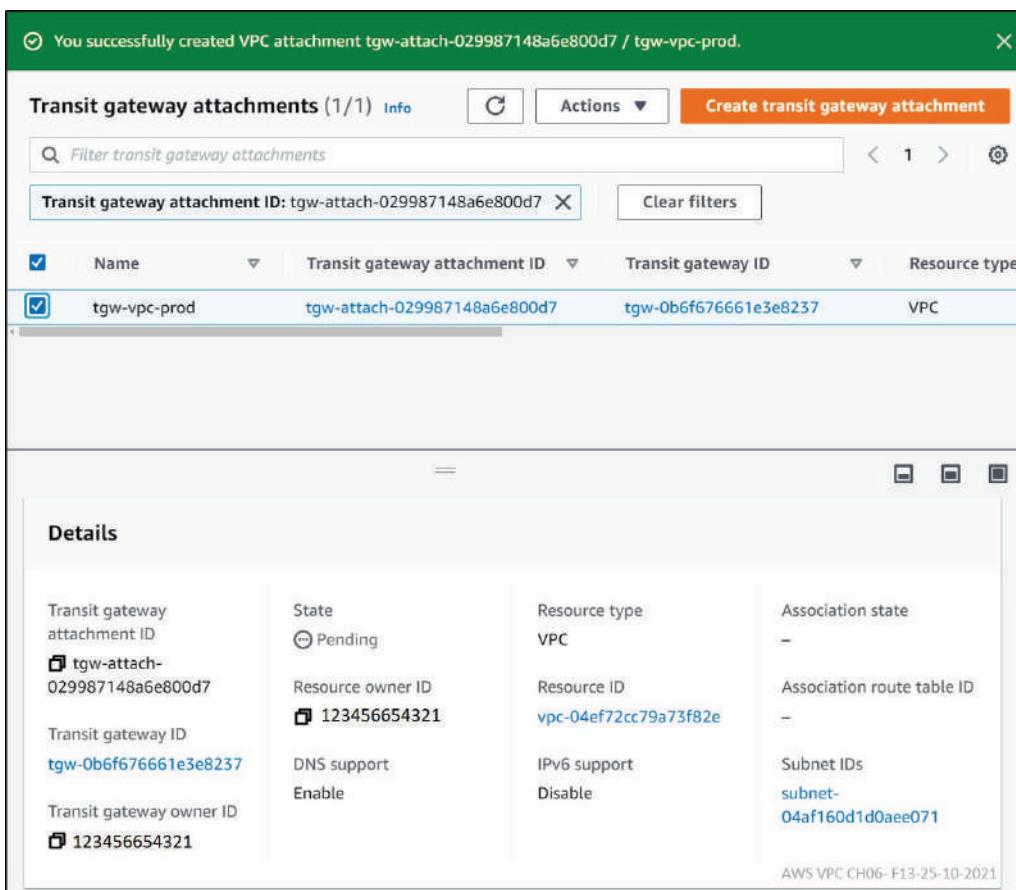


Figure 6-13: Creating Transit Gateway Attachment: Phase-1.4.

After attaching all three VPC to TGW, we can verify that routes TGW have propagated VPC CIDRs into the Route Table. To do that, navigate to the TGW Route table window. I have named the default TGW Route Table to TGW-MAIN-RT.

The screenshot shows the AWS Transit Gateway route tables list and a detailed view of the TGW-MAIN-RT table.

Transit gateway route tables (1/1) Info

Name	Transit gateway route table ID	Transit gateway ID	State
TGW-MAIN-RT	tgw-rtb-0a53e2ba10b858708	tgw-0b6f676661e3e8237	Available

tgw-rtb-0a53e2ba10b858708 / TGW-MAIN-RT

Details Associations Propagations Prefix list references Routes Tags

Details

Transit gateway route table ID tgw-rtb-0a53e2ba10b858708	State Available	Default association route table Yes	Default propagation route table Yes
Transit gateway ID tgw-0b6f676661e3e8237			

AWS VPC CH06-F14-25-10-2021

Figure 6-14: Creating Transit Gateway Attachment: State Verification.

The association tab shows Attachment Id, Attachment VPC, attached Resource Id, and the state (figure 6-15). The first Attachment Id, ending digits 88c1, is the VPC NWKT-Test attachment. The second one is the VPC NWKT-Dev attachment. The last one is the VPC NWKT-Prod attachment.

The screenshot shows the AWS VPC console interface. At the top, there's a header bar with the title "Transit gateway route tables (1/1)" and a "Create transit gateway route table" button. Below the header is a search bar labeled "Filter transit gateway route tables". The main content area displays a table with three columns: "Name", "Transit gateway route table ID", and "Transit gateway ID". A single row is selected, showing "TGW-MAIN-RT", "tgw-rtb-0a53e2ba10b858708", and "tgw-0b6f676661e3e8237" respectively, with a status of "Available". Below the table, the route table ID and name are repeated. At the bottom of the page, there are tabs for "Details", "Associations", "Propagations", "Prefix list references", "Routes", and "Tags", with "Associations" being the active tab. The "Associations" tab shows a table with four columns: "Attachment ID", "Resource t...", "Resource ID", and "State". Three associations are listed, all marked as "Associated": "tgw-attach-09c66a4def88488c1" (VPC, vpc-0a2c6c297c4562f11), "tgw-attach-0c134eb4773d8485b" (VPC, vpc-0867c98f35d11fcdd), and "tgw-attach-029987148a6e800d7" (VPC, vpc-04ef72cc79a73f82e).

Name	Transit gateway route table ID	Transit gateway ID	State
TGW-MAIN-RT	tgw-rtb-0a53e2ba10b858708	tgw-0b6f676661e3e8237	Available

Attachment ID	Resource t...	Resource ID	State
tgw-attach-09c66a4def88488c1	VPC	vpc-0a2c6c297c4562f11	Associated
tgw-attach-0c134eb4773d8485b	VPC	vpc-0867c98f35d11fcdd	Associated
tgw-attach-029987148a6e800d7	VPC	vpc-04ef72cc79a73f82e	Associated

Figure 6-15: Creating Transit Gateway Attachment: Association Verification.

The Propagations tab shows that the propagation is enabled in each of three attachments. This means that TGW will add CIDR ranges of VPCs into the associated Route Table. TGW installs CIDRs into the default Route Table of TGW. However, you can create additional Route Tables based on your segmentation policy and associate CIDRs with those.

The screenshot shows the AWS Transit Gateway Route Tables interface. At the top, there's a header with 'Transit gateway route tables (1/1)' and a 'Create transit gateway route table' button. Below the header is a search bar labeled 'Filter transit gateway route tables'. The main table has columns: Name, Transit gateway route table ID, Transit gateway ID, and State. One row is selected, showing 'TGW-MAIN-RT', 'tgw-rtb-0a53e2ba10b858708', 'tgw-0b6f676661e3e8237', and 'Available'. Below the table, the route table details for 'tgw-rtb-0a53e2ba10b858708 / TGW-MAIN-RT' are shown. The 'Propagations' tab is selected, displaying a sub-table with columns: Attachment ID, Resource type, Resource ID, and State. Three entries are listed, all marked as 'Enabled': 'tgw-attach-029987148a6e800d7' (VPC, vpc-04ef72cc79a73f82e), 'tgw-attach-09c66a4def88488c1' (VPC, vpc-0a2c6c297c4562f11), and 'tgw-attach-0c134eb4773d8485b' (VPC, vpc-0867c98f35d11fcdd).

Figure 6-16: Creating Transit Gateway Attachment: Propagation Verification.

The information in the Routes tab verifies that the route propagation works. TGW has installed all three VPC-specific CIDRs ranges into the default Route Table. Figure 6-17 shows that the Next-Hop for the CIDR 10.10.0.0/16 is attachment ending digits 800d7, and the CIDR is owned by VPC NWKT-Test (last digits 73f82e).

Routes (3)							
	CIDR	Attachment ID	Resource ID	R...	Route t...	Route state	Actions
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-029987148a6e800d7	vpc-04ef72cc79a73f82e	VPC	Propagated	Active	
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0c134eb4773d8485b	vpc-0867c98f35d11fcdd	VPC	Propagated	Active	
<input type="checkbox"/>	10.12.0.0/16	tgw-attach-09c66a4def88488c1	vpc-0a2c6c297c4562f11	VPC	Propagated	Active	

Figure 6-17: Creating Transit Gateway Attachment: Route Verification.

Update Subnet Route Tables

As the next step, we update subnet-specific Route Tables in within each VPC. Figure 6-18 shows the NWKT-Prod-RT where we have associated the subnet NWKT-Prod (10.10.0.0/24) before adding a route to TGW. Figure 6-19 illustrates the same Route Table after the update. The TGW is the Next-Hop for both subnets 10.11.0.0/24 and 10.12.0.0/24, while the IGW is the Next-Hop for all other destinations. Note that we haven't set up a VPN connection to the on-prem Datacenter yet. That is why we haven't added a route to network 172.16.10.0/24 at this phase.

Route tables (1/9) Info			
			Actions
<input type="checkbox"/>	NWKT-Test-RT	rtb-0f9540b028f439ac7	subnet-0f9fb0b5459ab07b4 / NWKT-Test
<input type="checkbox"/>	NWKT-PUB-RT	rtb-0fd4639034844c3ea	–
<input checked="" type="checkbox"/>	NWKT-Prod-RT	rtb-05dc9cd679937ceee	subnet-04af160d1d0aae071 / NWKT-Prod
<input type="checkbox"/>	NWKT-PRI-RT	rtb-0e7261b40b0d5237b	–
<input type="checkbox"/>	NWKT-MAIN-RT	rtb-069ar98ar692271fe	–

rtb-05dc9cd679937ceee / NWKT-Prod-RT					
Details	Routes	Subnet associations	Edge associations	Route propagation	Tags
					AWS-VPC-CH06-F18-25-10-2021
Routes (1)					
<input type="button" value="Edit routes"/> <div style="display: flex; justify-content: space-between;"> <input type="button" value="Filter routes"/> Both 1 </div>					
Destination	Target	Status	Propagated		
10.10.0.0/16	local	Active	No		

Figure 6-18: Subnet-Specific Route Tables Before Update.

Routes (4)					Edit routes
<input type="text"/> Filter routes			Status	Propagated	
Destination	Target	Status	Propagated	Actions	
10.10.0.0/16	local	Active	No		
10.11.0.0/16	tgw-0b6f676661e3e8237	Active	No		
10.12.0.0/16	tgw-0b6f676661e3e8237	Active	No		
0.0.0.0/0	igw-0e997955bc6597031	Active	No	AWS VPC CHD6- F19-25-10-2021	

Figure 6-19: Subnet-Specific Route Tables After Update.

Example 6-1 shows how we can verify VPC to TGW attachments by using AWS CLI command.

```
aws ec2 describe-transit-gateway-vpc-attachments --output table

-----+-----+-----+-----+-----+
|           DescribeTransitGatewayVpcAttachments |           |
+-----+-----+-----+-----+-----+
||          TransitGatewayVpcAttachments          ||          |
+-----+-----+-----+-----+-----+
|| CreationTime          | 2021-10-24T07:53:42.000Z |          |
|| State                | available               |          |
|| TransitGatewayAttachmentId | tgw-attach-029987148a6e800d7 |          |
|| TransitGatewayId      | tgw-0b6f676661e3e8237 |          |
|| VpcId                | vpc-04ef72cc79a73f82e |          |
|| VpcOwnerId            | 123456654321           |          |
+-----+-----+-----+-----+-----+
||          Options          ||          |
+-----+-----+-----+-----+-----+
||| DnsSupport           | enable                  |          |
||| Ipv6Support          | disable                 |          |
+-----+-----+-----+-----+-----+
||          SubnetIds          ||          |
+-----+-----+-----+-----+-----+
||| subnet-04af160d1d0aee071 |          |          |
+-----+-----+-----+-----+-----+
||          Tags          ||          |
+-----+-----+-----+-----+-----+
||| Key        | Name       |          |
||| Value      | tgw-vpc-Prod |          |
+-----+-----+-----+-----+-----+
||          TransitGatewayVpcAttachments          ||          |
+-----+-----+-----+-----+
|| CreationTime          | 2021-10-24T07:55:56.000Z |          |
|| State                | available               |          |
|| TransitGatewayAttachmentId | tgw-attach-09c66a4def88488c1 |          |
+-----+-----+-----+-----+-----+
|| VpcId                | vpc-0a2c6c297c4562f11 |          |
|| VpcOwnerId            | 123456654321           |          |
+-----+-----+-----+-----+-----+
||          Options          ||          |
+-----+-----+-----+-----+-----+
||| DnsSupport           | enable                  |          |
||| Ipv6Support          | disable                 |          |
+-----+-----+-----+-----+-----+
```

```

| |+-----+-----+ | | | |
| ||          SubnetIds           | |
| ||+-----+-----+ |
| ||  subnet-0af695139fdb5dcd | |
| ||+-----+-----+ |
| ||          Tags               | |
| ||+-----+-----+ |
| ||  Key      | Name        | |
| ||  Value    | tgw-vpc-Dev | |
| ||+-----+-----+ |
| ||          TransitGatewayVpcAttachments | |
| +-----+-----+ |
| | CreationTime   | 2021-10-24T07:55:18.000Z | |
| | State         | available | |
| | TransitGatewayAttachmentId | tgw-attach-0c134eb4773d8485b | |
| | TransitGatewayId     | tgw-0b6f676661e3e8237 | |
| | VpcId          | vpc-0867c98f35d11fcdd | |
| | VpcOwnerId     | 123456654321 | |
| +-----+-----+ |
| |          Options            | | | |
| |+-----+-----+ |
| ||  DnsSupport  | enable | |
| ||  Ipv6Support | disable | |
| |+-----+-----+ |
| ||          SubnetIds           | |
| |+-----+-----+ |
| ||  subnet-0f9fb0b5459ab07b4 | |
| |+-----+-----+ |
| ||          Tags               | |
| |+-----+-----+ |
| ||  Key      | Name        | |
| ||  Value    | tgw-vpc-Test | |
| |+-----+-----+ |

```

Example 6-1: VPC to TGA Attachment Verification from the AWS CLI.

Data-Plane Testing

Example 6-2 verifies that we have an IP connectivity between EC2 instances 10.10.0.218 (VPC NWKT-Prod) and 10.11.0.237 (VPC NWKT-Test).

```

[ec2-user@ip-10-10-0-218 ~]$ ping 10.11.0.237
PING 10.11.0.237 (10.11.0.237) 56(84) bytes of data.
64 bytes from 10.11.0.237: icmp_seq=1 ttl=254 time=0.727 ms
64 bytes from 10.11.0.237: icmp_seq=2 ttl=254 time=0.716 ms
64 bytes from 10.11.0.237: icmp_seq=3 ttl=254 time=0.712 ms
64 bytes from 10.11.0.237: icmp_seq=4 ttl=254 time=0.829 ms
64 bytes from 10.11.0.237: icmp_seq=5 ttl=254 time=0.670 ms
64 bytes from 10.11.0.237: icmp_seq=6 ttl=254 time=0.710 ms
64 bytes from 10.11.0.237: icmp_seq=7 ttl=254 time=0.911 ms
64 bytes from 10.11.0.237: icmp_seq=8 ttl=254 time=0.791 ms
^C
--- 10.11.0.237 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7151ms
rtt min/avg/max/mdev = 0.670/0.758/0.911/0.077 ms

```

Example 6-2: VPC to TGA Attachment Data-Plane Testing.

Create VPN Connection

Next, we build a VPN connection between Transit Gateway (TGW) and pre-defined Customer Gateway (CGW). The first step is to create a VPN attachment and define the CGW in TGW. After that, we configure the on-prem Datacenter WAN-Edge-01 (our CGW) routers' IPSec and BGP settings based on the configuration file generated during the TGW VPN configuration. The previous chapter explained the file download process.

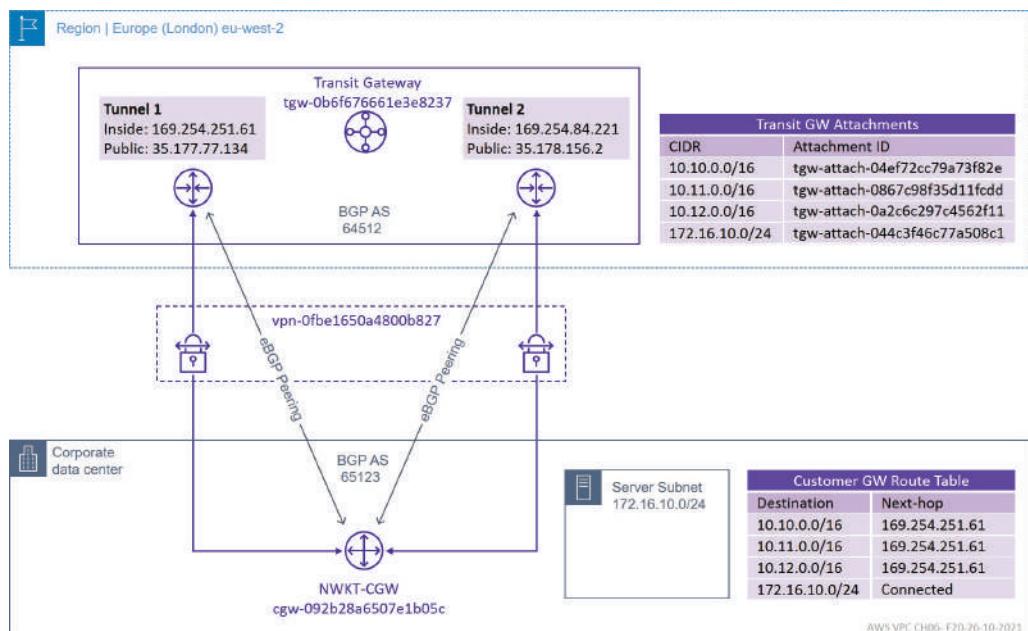


Figure 6-20: VPN Connection Between the Transit Gateway and the Customer Gateway.

Configure VPN on TGW

VPN connection configuration follows the same principles that we used when we attach VPCs to TGW. First, select the TGW from the Transit gateway ID drop-down menu. Then we choose the Attachment type from the Attachment type VPN drop-down menu. Next, we choose our pre-created CGW from the Customer gateway ID drop-down menu. In order to send BGP Update messages about CIDRs installed into TGWs Route Table (the one we are using with the VPN = default), we need to select the Dynamic Routing options.

Create transit gateway attachment Info

A transit gateway (TGW) is a network transit hub that interconnects attachments (VPCs and VPNs) within the same AWS account or across AWS accounts.

Details

Transit gateway ID Info
tgw-0b6f676661e3e8237 (NWKT-TGW)

Attachment type Info
VPN

VPN Attachment
Create a new customer gateway or select an existing customer gateway that you would like to connect to the transit gateway via a VPN connection.

Customer Gateway Info
 Existing
 New

Customer Gateway ID Info
cgw-092b28a6507e1b05c (NWKT-CGW)

Routing options Info
 Dynamic (requires BGP)
 Static

Enable Acceleration (Improve performance of VPN tunnels via AWS Global Accelerator and the AWS global network) Info

AWS VPC CH06- F21-26-10-2021

Figure 6-21: Configuring VPN Between TGW and CGW – Endpoints and Routing.

We use Amazon-generated Inside IP addresses and Pre-Shared keys for both tunnels. Click the Create transit gateway attachment button to proceed.

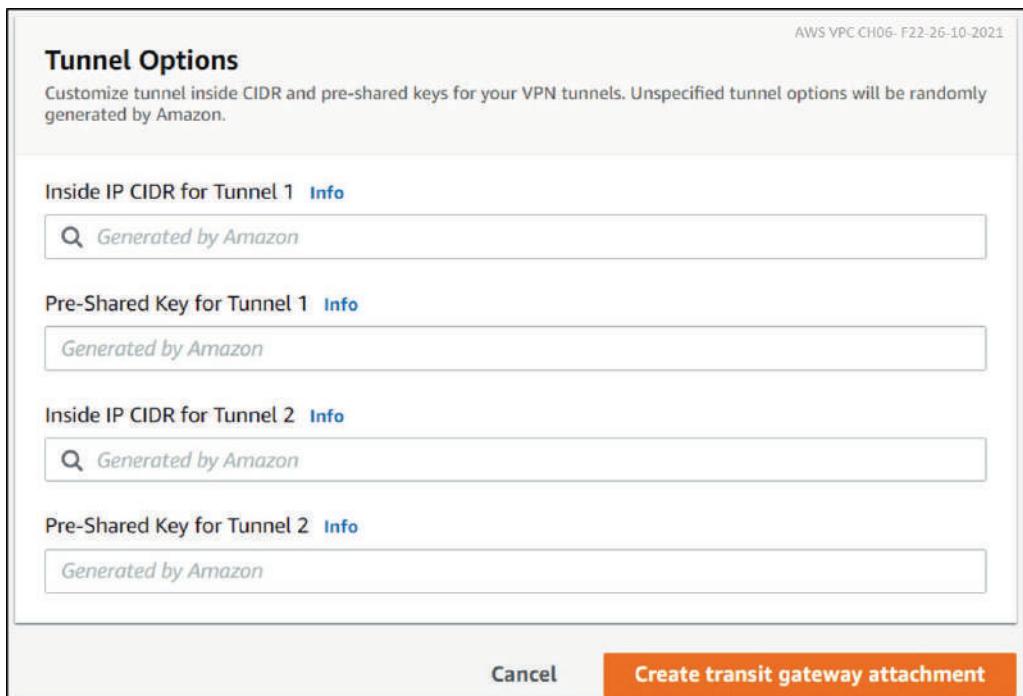


Figure 6-22: Configuring VPN Between TGW and CGW – Tunnel settings.

You will get a notification about successful VPN attachment. Figure 6-23 shows that VPN resource `vpn-0fbe1650a4800b827` is attached to TGW `tgw-0b6f676661e3e8237` using an attachment `tgw-attach-0e4c3f46c77a508c1`. VPN is associated with TGWs main Route Table `tgw-trb-0a53e2ba10b858708` (TGW routes ingress packets from VPN attachment based main Route Table).

You successfully created VPN attachment vpn-0fbe1650a4800b827 / tgw-VPN.
The VPN connection was successfully created. The new attachment may take up to a minute to appear in the transit gateway attachments grid. Refresh the grid to see the attachment. To manage this VPN connection, visit [VPN Connections](#).

Transit gateway attachments (1/4) Info				
	Name	Transit gateway attachment ID	Transit gateway ID	Resource type
<input type="checkbox"/>	tgw-vpc-Prod	tgw-attach-029987148a6e800d7	tgw-0b6f676661e3e8237	VPC
<input type="checkbox"/>	tgw-vpc-Dev	tgw-attach-09c65a4def88488c1	tgw-0b6f676661e3e8237	VPC
<input type="checkbox"/>	tgw-vpc-Test	tgw-attach-0c134eb4773d8485b	tgw-0b6f676661e3e8237	VPC
<input checked="" type="checkbox"/>	-	tgw-attach-0e4c3f46c77a508c1	tgw-0b6f676661e3e8237	VPN

tgw-attach-0e4c3f46c77a508c1

[Details](#) [Tags](#)

Details

Transit gateway attachment ID tgw-attach-0e4c3f46c77a508c1	State Available	Resource type VPN	Resource ID vpn-0fbe1650a4800b827
Transit gateway ID tgw-0b6f676661e3e8237	Association route table ID tgw-rtb-0a53e2ba10b858708	Association state Associated	Resource owner ID 123456654321
Transit gateway owner ID 123456654321			

AWS VPC CH06- F23-26-10-2021

Figure 6-23: Configuring VPN Between TGW and CGW – Verification.

Click the [vpn-0fbe1650a4800b827](#) hyperlink under Resource ID header in Details tab (figure 6-23) for moving to VPN Connection page.

Figure 6-24 shows that we have configured a VPN connection between TGW and CGW and its state available. It also shows the CGW's public IP address, and that we are using a pre-shared key for authentication. Local and remote IPv4 CIDRs are set to 0.0.0.0/0. We can change CIDRs, if needed, by choosing the *Modify VPN Connection Options* from the *Action* drop-down menu. Figure 6-26 and 6-27 shows the process. If you change the CIDRs, download the CGW configuration after you have done modifications.

VPN ID	State	Customer Gateway Address	
vpn-0fbe1650a4800b827	available	91.153.26.147	
Virtual Private Gateway	-	Customer Gateway	
Transit Gateway	tgw-0b6f676661e3e8237	Customer Gateway Address	
Type	ipsec.1	Category	VPN
VPC	-	Routing	Dynamic
Acceleration Enabled	false	Authentication Type	Pre Shared Key
Local IPv4 Network Cidr	0.0.0.0/0	Remote IPv4 Network Cidr	0.0.0.0/0
Local IPv6 Network Cidr	-	Remote IPv6 Network Cidr	-

Figure 6-24: Configuring VPN Between TGW and CGW – VPN Status on TGW.

We haven't configured the CGW device in on-prem-DC at this phase, and that is why both tunnels are down.

Tunnel Num	Outside IP Address	Inside IPv4 CIDR	Status	Status Last Changed	Details
Tunnel 1	35.177.77.134	169.254.251.60/30	DOWN	October 24, 2021 at 11:54:35 A...	IPSEC IS DOWN
Tunnel 2	35.178.156.2	169.254.84.220/30	DOWN	October 24, 2021 at 11:54:32 A...	IPSEC IS DOWN

Figure 6-25: Configuring VPN Between TGW and CGW – Tunnel Status: Down.

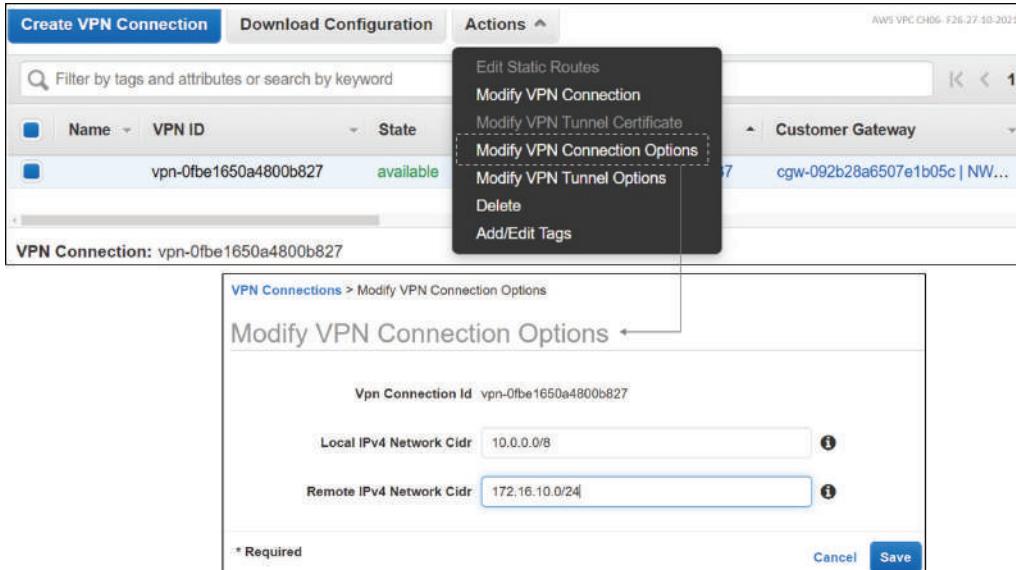


Figure 6-26: Configuring VPN Between TGW and CGW – Modifying CIDRs.

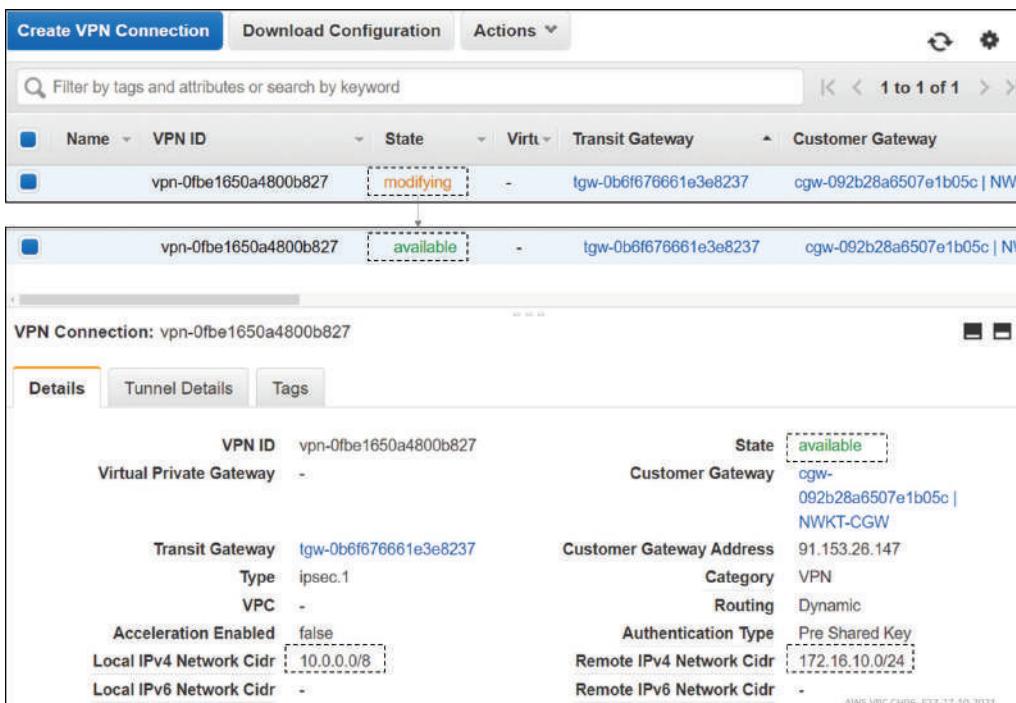


Figure 6-27: Configuring VPN Between TGW and CGW – Modifying CIDRs.

Configure VPN on CGW

Example 6-3 shows the CGW configuration. Chapter 5 explained how you can download the configuration.

```
NWKT-WAN-Edge-01
!
crypto keyring keyring-vpn-0fbe1650a4800b827-1
    local-address GigabitEthernet1
    pre-shared-key address 35.178.156.2 key _7NsYsj7vd0MeKuCZZGDThtsyajxwisT
crypto keyring keyring-vpn-0fbe1650a4800b827-0
    local-address GigabitEthernet1
    pre-shared-key address 35.177.77.134 key GdOrYGG3jvzCAYlXhhYDcFuu01VD4RFu
!
crypto isakmp policy 200
    encr aes
    authentication pre-share
    group 2
    lifetime 28800
!
crypto isakmp policy 201
    encr aes
    authentication pre-share
    group 2
    lifetime 28800
crypto isakmp keepalive 10 10
crypto isakmp profile isakmp-vpn-0fbe1650a4800b827-0
    keyring keyring-vpn-0fbe1650a4800b827-0
    match identity address 35.177.77.134 255.255.255.255
    local-address GigabitEthernet1
crypto isakmp profile isakmp-vpn-0fbe1650a4800b827-1
    keyring keyring-vpn-0fbe1650a4800b827-1
    match identity address 35.178.156.2 255.255.255.255
    local-address GigabitEthernet1
!
crypto ipsec security-association replay window-size 128
!
crypto ipsec transform-set ipsec-prop-vpn-0fbe1650a4800b827-0 esp-aes esp-sha-hmac
    mode tunnel
crypto ipsec transform-set ipsec-prop-vpn-0fbe1650a4800b827-1 esp-aes esp-sha-hmac
    mode tunnel
crypto ipsec df-bit clear
!
!
crypto ipsec profile ipsec-vpn-0fbe1650a4800b827-0
    set transform-set ipsec-prop-vpn-0fbe1650a4800b827-0
    set pfs group2
!
crypto ipsec profile ipsec-vpn-0fbe1650a4800b827-1
    set transform-set ipsec-prop-vpn-0fbe1650a4800b827-1
    set pfs group2
!
!
!
interface Tunnel1
    ip address 169.254.251.62 255.255.255.252
    ip tcp adjust-mss 1379
    tunnel source GigabitEthernet1
```

```

tunnel mode ipsec ipv4
tunnel destination 35.177.77.134
tunnel protection ipsec profile ipsec-vpn-0fbe1650a4800b827-0
ip virtual-reassembly
!
interface Tunnel1
ip address 169.254.84.222 255.255.255.252
ip tcp adjust-mss 1379
tunnel source GigabitEthernet1
tunnel mode ipsec ipv4
tunnel destination 35.178.156.2
tunnel protection ipsec profile ipsec-vpn-0fbe1650a4800b827-1
ip virtual-reassembly
router bgp 65123
bgp log-neighbor-changes
neighbor 169.254.84.221 remote-as 64512
neighbor 169.254.84.221 timers 10 30 30
neighbor 169.254.251.61 remote-as 64512
neighbor 169.254.251.61 timers 10 30 30
!
address-family ipv4
network 172.16.10.0 mask 255.255.255.0
neighbor 169.254.84.221 activate
neighbor 169.254.84.221 default-originate
neighbor 169.254.84.221 soft-reconfiguration inbound
neighbor 169.254.251.61 activate
neighbor 169.254.251.61 default-originate
neighbor 169.254.251.61 soft-reconfiguration inbound
exit-address-family

```

Example 6-3: Configuring VPN and BGP on CGW.

Control-Plane and Data-Plane Verification

Example 6-4 and figure 6-28 show that after CGW configuration, tunnels are up on both CGW and TGW.

NWKT-WAN-Edge-01#sh ip int bri					
Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet1	192.168.100.18	YES	DHCP	up	up
GigabitEthernet2	unassigned	YES	unset	administratively down	down
GigabitEthernet3	unassigned	YES	unset	administratively down	down
GigabitEthernet4	unassigned	YES	unset	administratively down	down
Loopback172	172.16.10.1	YES	manual	up	up
Tunnel1	169.254.251.62	YES	manual	up	up
Tunnel2	169.254.84.222	YES	manual	up	up

Example 6-4: Tunnels State Verification on CGW.

AWS VPC CH06-F2B-27-10-2023						
Tunnel Num	Outside IP Address	Inside IPv4 CIDR	Status	Status Last Changed	Details	
Tunnel 1	35.177.77.134	169.254.251.60/30	UP	October 24, 2021 at 4:22:11 P...	2 BGP ROUTES	
Tunnel 2	35.178.156.2	169.254.84.220/30	UP	October 24, 2021 at 4:21:50 P...	2 BGP ROUTES	

Figure 6-28: Tunnels State Verification on TGW.

Example 6-5 verifies that BGP peering between CGW and TGW is established over IPSec tunnel 1 (169.254.251.61) and tunnel 2 (169.254.84.221).

```
NWK-T-WAN-Edge-01#sh ip bgp summ
BGP router identifier 192.168.100.18, local AS number 65123
BGP table version is 6, main routing table version 6
4 network entries using 992 bytes of memory
7 path entries using 952 bytes of memory
2/1 BGP path/bestpath attribute entries using 560 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 2528 total bytes of memory
BGP activity 4/0 prefixes, 7/0 paths, scan interval 60 secs

Neighbor      V      AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down State/PfxRcd
169.254.84.221  4      64512     16      18       6    0    0 00:02:08      3
169.254.251.61  4      64512     32      35       6    0    0 00:04:46      3
```

Example 6-5: BGP Peering Verification on CGW.

Example 6-6 verifies that CGW has received BGP Updates about VPC CIDRs from the TGW over both tunnels. CGW has selected routes received from the 169.254.251.61 as a best path because it is the oldest one (example 6-5 shows the time-stamps for peering).

```
NWK-T-WAN-Edge-01#sh ip bgp
BGP table version is 6, local router ID is 192.168.100.18
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop            Metric LocPrf Weight Path
        0.0.0.0          0.0.0.0
* 10.10.0.0/16      169.254.84.221      100      0 64512 i
*>                  169.254.251.61      100      0 64512 i
* 10.11.0.0/16      169.254.84.221      100      0 64512 i
*>                  169.254.251.61      100      0 64512 i
* 10.12.0.0/16      169.254.84.221      100      0 64512 i
*>                  169.254.251.61      100      0 64512 i
```

Example 6-6: BGP Local RIB Verification.

Example 6-7 verifies that CGW has installed routes learned from the TGW into the routing table.

```
NWKT-WAN-Edge-01#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2
      i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
      ia - IS-IS inter area, * - candidate default, U - per-user static route
      o - ODR, P - periodic downloaded static route, H - NHRP, 1 - LISP
      a - application route
      + - replicated route, % - next hop override, p - overrides from Pfr

Gateway of last resort is 192.168.100.1 to network 0.0.0.0

S*   0.0.0.0/0 [254/0] via 192.168.100.1
      10.0.0.0/16 is subnetted, 3 subnets
B     10.10.0.0 [20/100] via 169.254.251.61, 00:05:20
B     10.11.0.0 [20/100] via 169.254.251.61, 00:05:20
B     10.12.0.0 [20/100] via 169.254.251.61, 00:05:20
      169.254.0.0/16 is variably subnetted, 4 subnets, 2 masks
C     169.254.84.220/30 is directly connected, Tunnel2
L     169.254.84.222/32 is directly connected, Tunnel2
C     169.254.251.60/30 is directly connected, Tunnel1
L     169.254.251.62/32 is directly connected, Tunnel1
      192.168.100.0/24 is variably subnetted, 2 subnets, 2 masks
C     192.168.100.0/24 is directly connected, GigabitEthernet1
L     192.168.100.18/32 is directly connected, GigabitEthernet1
```

Example 6-7: Routing Table Verification.

Figures 6-29 and 6-30 shows that TGW has learned route to on-prem Datacenter network 172.16.10.0/24 via attachments tgw-attach-0e4c3f46c77a508c1 over VPN connections vpn-0fbe1650a4800b827 (Tunnel 1: 35.177.77.134 and Tunnel 2: 35.178.156.2). Note that I didn't change the remote CIDR and that is why we have 0.0.0.0/0 entry learned via VPN. Remember to add the route to 172.16.10.0/24 to the Route Table associated with the subnet NWKT-Prod (figure 6-31).

Routes (1/5)							AWS VPC CH06-F29-27.10.2021
	CIDR	Attachment ID	Resource ID	Resource...	Route type	Action	Create static route
<input type="checkbox"/>	0.0.0.0/0	2 Attachments	2 Resources	VPN	Propagated		
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-029987148a6e800d7	vpc-04ef72cc79a73f82e	VPC	Propagated		
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0c134eb4773d8485b	vpc-0867c98f35d11fcdd	VPC	Propagated		
<input type="checkbox"/>	10.12.0.0/16	tgw-attach-09c66a4def88488c1	vpc-0a2c6c297c4562f11	VPC	Propagated		
<input checked="" type="checkbox"/>	172.16.10.0/24	2 Attachments	tgw-attach-0e4c3f46c77a508c1 X	VPN	Propagated		

Figure 6-29: TGW Routing Table.

Routes (1/5)						AWS VPC CH06- F31-27-10-2021	
	CIDR	Attachment ID	Resource ID	Resour...	Route type	Actions	Create static route
<input type="checkbox"/>	0.0.0.0/0	2 Attachments	2 Resources	VPN	Propagated		
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-029987148a6e800d7	ypc-04ef72cc79a73f82e	VPC	Propagated		
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0c134eb4773d8485b	ypc-0867c98f35d11fcdd	VPC	Propagated		
<input type="checkbox"/>	10.12.0.0/16	tgw-attach-09c66a4def88488c1	ypc-0a2c6c297c4562f11	VPC	Propagated		
<input checked="" type="checkbox"/>	172.16.10.0/24	2 Attachments	2 Resources	vpn-0fbe1650a4800b827(35.177.77.134) vpn-0fbe1650a4800b827(35.178.156.2)	Gated		

Figure 6-30: TGW Routing Table.

rtb-05dc9cd679937ceee / NWKT-Prod-RT						AWS VPC CH06- F31-27-10-2021
Details Info						
Route table ID	Main	Explicit subnet associations	Edge associations			
<input type="checkbox"/> rtb-05dc9cd679937ceee	<input type="checkbox"/> No	subnet-04af160d1d0aee071 / NWKT-Prod	-			
VPC	Owner ID	<input type="checkbox"/> 123456654321				
vpc-04ef72cc79a73f82e NWKT-Prod						
Routes		Subnet associations	Edge associations	Route propagation	Tags	
Routes (5)						Edit routes
<input type="checkbox"/> Filter routes						Both
						< 1 >
Destination						Propagated
172.16.10.0/24	tgw-0b6f676661e3e8237	Active	No			
10.10.0.0/16	local	Active	No			
10.11.0.0/16	tgw-0b6f676661e3e8237	Active	No			
10.12.0.0/16	tgw-0b6f676661e3e8237	Active	No			
0.0.0.0/0	igw-0e997955bc6597031	Active	No			

Figure 6-31: Subnet NWKT-Prod Route Table.

Examples 6-8 and 6-9 verifies that we have IP connectivity between on-prem Datacenter and VPCs.

```
NWKT-WAN-Edge-01#ping 10.11.0.237 source loopback 172
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.11.0.237, timeout is 2 seconds:
Packet sent with a source address of 172.16.10.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 35/35/36 ms
```

Example 6-8: *Data-Plane Verification Between 172.16.10.1 – 10.11.0.237 (NWKT-Test).*

```
NWKT-WAN-Edge-01#ping 10.10.0.218 source loopback 172
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.0.218, timeout is 2 seconds:
Packet sent with a source address of 172.16.10.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 35/35/36 ms
```

Example 6-9: *Data-Plane Verification Between 172.16.10.1 – 10.10.0.218 (NWKT-Prod).*

Transit Gateway Pricing

Figure 6-31 shows the cost components related to Transit Gateway. In addition to VPN connection hours, the Transit Gateway cost structure includes attachment-based hourly costs, which is 0.06\$ per hour. Other than that, AWS charges 0.05\$ per VPN connection hour (same as VGW solution), and attachment-based data processing charges 0.02\$ per GB. The design based on the VGW and VPC peering solutions is cheaper if you only have a couple of VPCs. That is because there are no attachment-based costs.

AWS Service Charges		\$16.42
▼ Data Transfer		\$0.00
▶ EU (London)		\$0.00
▼ Elastic Compute Cloud		\$5.80
▶ EU (London)		\$5.80
▶ Key Management Service		\$0.00
▼ Virtual Private Cloud		\$7.52
▶ EU (London)		\$7.52
No Instance Type		\$7.52
\$0.02 per GB Data Processed by Transit Gateway VPC Attachment	0.000017 GigaBytes	\$0.00
\$0.02 per GB Data Processed by Transit Gateway VPN Attachment	0.000002 GigaBytes	\$0.00
\$0.05 per VPN Connection-Hour	106.000 Hrs	\$5.30
\$0.06 per Transit Gateway VPC Attachment Hour	21.000 hour	\$1.26
\$0.06 per Transit Gateway VPN Attachment Hour	6.000 hour	\$0.36
\$0.1 per analysis processed by VPC Reachability Analyzer	6.000 ReachabilityAnalyses	\$0.60
Taxes		
VAT to be collected		\$3.10

AWS VPC CH06_F12-27-10-2021

Figure 6-32: TGW Cost Components.

Chapter 7: VPC Segmentation with Transit Gateway

Introduction

The previous chapter discusses an on-premise and VPC connection architecture with Transit Gateways (TGW). This chapter introduces a VPC segmentation model based on the TGW's Route Tables (RT). Let's start by defining our segmentation policy. First, traffic to and from VPCs NWKT-Prod and NWKT-Test to the VPC NWKT-Shared and on-premise DC over VPN is allowed. Packet flows between VPCs NWKT-Prod and NWKT-Test are denied. Traffic flows between the VPC NWKT-Shared and on-prem DC over VPC are not allowed. Figure 7-1 illustrates what we are going to do in order to full fill these policy requirements. I have preconfigured all attachments, and by default, they are all associated with the default Route Table of TGW. As the first step, we create dedicated Route Tables for each VPC (NWKT-Shared, NWKT-Prod, and NWKT-Test) and VPN connection. Then we detach VPCs and VPN attachments from the default Route Table and attach them into dedicated, resource-specific Route Tables. Next, we are going to modify routes in each Route Table to implement a segmentation policy. VPCs NWK-test and NWKT-Prod share the Route Table. To allow traffic flows between on-premise DC and these two VPCs, we need to propagate (=import) routes from the TGW-London-VPN-RT into TGW-London-VPC-RT and another way around. Then we'll do the same operation between TGW-London-VPC-RT and TGW-London-Shared-RT.

Figure7-1 illustrates the situation after we have done all necessary propagation. After propagation, each route table includes routes and their associated next hop. For example, TGW-London-VPC-RT has two routing entries, a) destination 172.16.10.0/24 with the next-hop pointing to the VPN attachment, and b) 10.12.0.0/16 pointing to the VPC attachment NWK-Shared. The propagated route from the VPC Route Table is the CIDR range associated with the attached VPC, not the mask of any particular subnet within the VPC. The TGW Route Table propagation process is like an import process, based on BGP Route-Targets, in traditional router virtualization (within a router).

The difference is that we don't have to export routes from TGWs Route Table to install them into the other Route Table within the same TGW. In the traditional router virtualization, the BGP process adds an extended BGP community Route-Target to NLRI (Network Layer Reachability Information) during the export process. Then, we use exported route-target when we import NLRI into another VRF. You can add 20 Route Tables per TGW. TGW can have up to 10 000 static routes distributed among its RTs. TGW can advertise 5000 routes and receive 1000 routes from its peers. You can add one static route for a prefix towards single attachment.

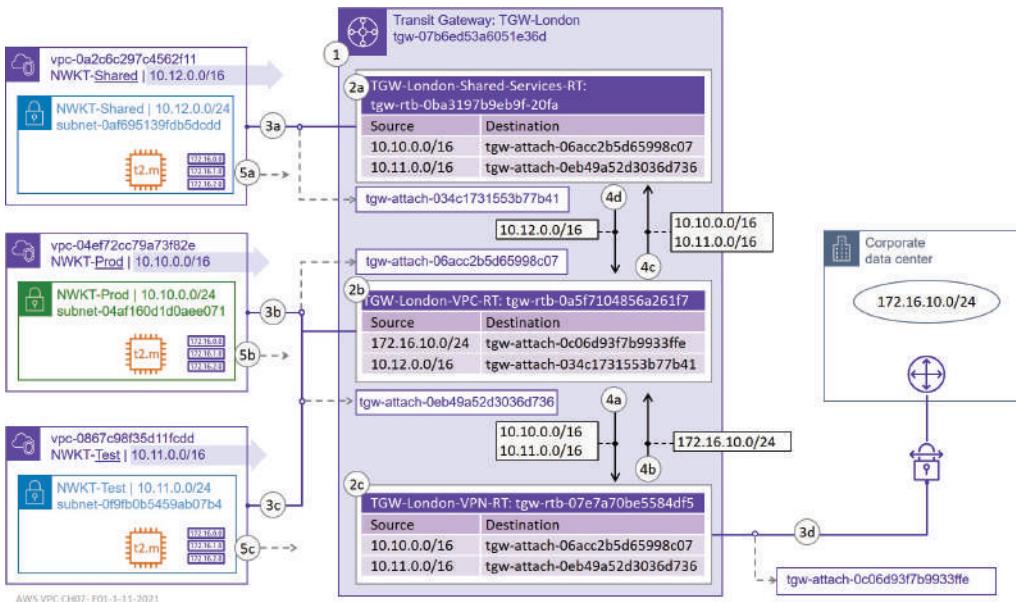


Figure 7-1: Transit Gateway Segmentation Overall Model.

Figure 7-2 shows that our TGW-London (tgw-07b6ed53a6051e36d) has one associated Route Table tgw-rtb-0648db5cfa12dc1a2. The same Route Table is also the Propagation Route Table. Note that the propagation process is explained later in this chapter.

The screenshot shows the AWS Transit Gateways console. At the top, there is a search bar labeled "Filter transit gateways" and a navigation bar with icons for back, forward, and refresh, along with "Actions" and "Create transit gateway" buttons. Below the header, a table lists one transit gateway:

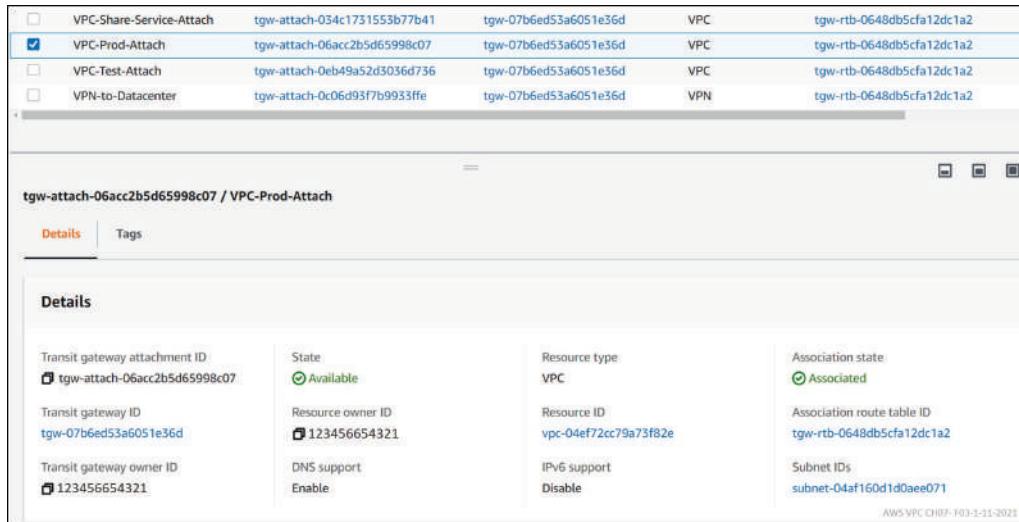
Name	Transit gateway ID	State	Amazon ASN
TGW-London	tgw-07b6ed53a6051e36d	Available	64512

Below the table, the specific transit gateway "tgw-07b6ed53a6051e36d / TGW-London" is selected. The "Details" tab is active, showing the following configuration:

Transit gateway ID tgw-07b6ed53a6051e36d	State Available	Amazon ASN 64512	DNS support Enable
Transit gateway ARN arn:aws:ec2:eu-west-2:017857243309:transit-gateway/tgw-07b6ed53a6051e36d	Default association route table Enable	Association route table ID tgw-rtb-0648db5cfa12dc1a2	Auto accept shared attachments Disable
Owner ID 123456654321	Default propagation route table Enable	Propagation route table ID tgw-rtb-0648db5cfa12dc1a2	VPN ECMP support Enable
Description eu-west-2 TGW	Transit gateway CIDR blocks -	Multicast support Disable	AWS VPC CH07- F02-1-11-2021

Figure 7-2: TGW-London.

Figure 7-3 verifies that TGW-London has three VPC attachments and one VPN attachment. We haven't added any resource-specific Route Tables yet, and all attachments are associated with the default Route Table. Figure below shows the details of the attachment VPC-Prod-Attach (tgw-attach-06acc2b5d65998c07), which associates the VPC NWKT-Prod (vpc-04ef72cc79a73f82e) to this TGW's default Route Table (tgw-rtb-0648db5cfa12dc1a2). It also shows the subnet Id (subnet-04af160d1d0aee071) of the subnet NWKT-Prod (10.10.0.0/24).



The screenshot shows a table of resource associations. The columns are: Attachment ID, Name, Resource ID, Resource Type, and Association state. There are four rows:

- VPC-Share-Service-Attach, tgw-attach-034c1731553b77b41, tgw-07b6ed53a6051e36d, VPC, tgw-rtb-0648db5cfa12dc1a2
- VPC-Prod-Attach**, tgw-attach-06acc2b5d65998c07, tgw-07b6ed53a6051e36d, VPC, tgw-rtb-0648db5cfa12dc1a2
- VPC-Test-Attach, tgw-attach-0eb49a52d3036d736, tgw-07b6ed53a6051e36d, VPC, tgw-rtb-0648db5cfa12dc1a2
- VPN-to-Datacenter, tgw-attach-0c06d93f7b9933ff, tgw-07b6ed53a6051e36d, VPN, tgw-rtb-0648db5cfa12dc1a2

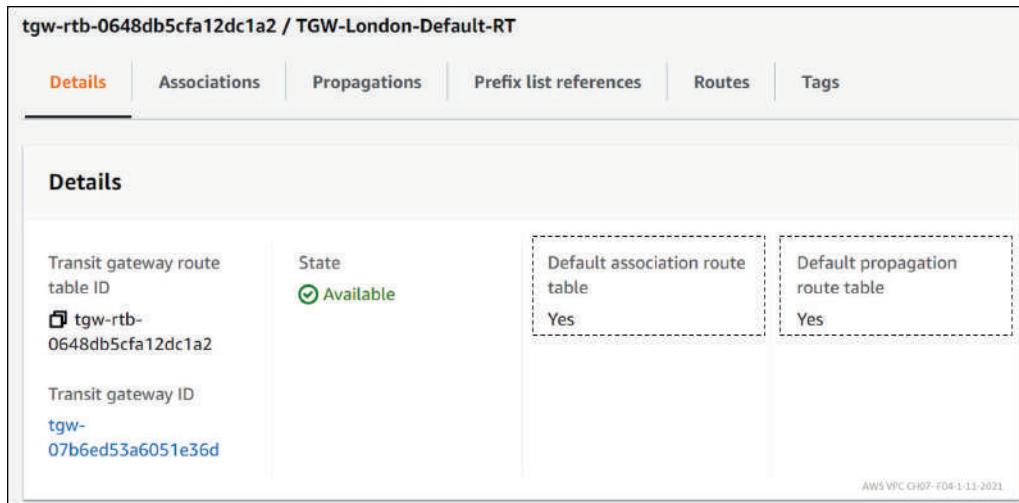
Below the table, a detailed view for the VPC-Prod-Attach row is shown. The title is "tgw-attach-06acc2b5d65998c07 / VPC-Prod-Attach". The "Details" tab is selected. The table contains the following data:

Transit gateway attachment ID	State	Resource type	Association state
tgw-attach-06acc2b5d65998c07	Available	VPC	Associated
Transit gateway ID	Resource owner ID	Resource ID	Association route table ID
tgw-07b6ed53a6051e36d	123456654321	vpc-04ef72cc79a73f82e	tgw-rtb-0648db5cfa12dc1a2
Transit gateway owner ID	DNS support	IPv6 support	Subnet IDs
123456654321	Enable	Disable	subnet-04af160d1d0ae071

AWS VPC CH07-F03-1-11-2021

Figure 7-3: TGW-London Resource Associations.

Next, we'll take a look at the TGW's default Route Table. Among the TGW Route Table Id and TGW IDs, the Details tab in figure 7-4 shows that all attachments are associated with this RT by default. It also shows that this is the default propagation RT.



The screenshot shows a "Details Sheet" for the TGW-London-Default-RT. The tabs at the top are: Details, Associations, Propagations, Prefix list references, Routes, and Tags. The "Details" tab is selected. The table contains the following data:

Transit gateway route table ID	State	Default association route table	Default propagation route table
tgw-rtb-0648db5cfa12dc1a2	Available	Yes	Yes
Transit gateway ID			
tgw-07b6ed53a6051e36d			

AWS VPC CH07-F04-1-11-2021

Figure 7-4: TGW-London-Default-RT – Details Sheet.

The Association tab shows the associated resource attachments with their resource type and resource Id. You can find the resource id and resource name mapping from figure 7-1.

Associations (4) <small>Info</small>					
<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State	
<input type="checkbox"/>	tgw-attach-06acc2b5d65998c07	VPC	vpc-04ef72cc79a73f82e	<input checked="" type="checkbox"/> Associated	
<input type="checkbox"/>	tgw-attach-034c1731553b77b41	VPC	vpc-0a2c6c297c4562f11	<input checked="" type="checkbox"/> Associated	
<input type="checkbox"/>	tgw-attach-0c06d93f7b9933ffe	VPN	vpn-00ce8797da115243a	<input checked="" type="checkbox"/> Associated	
<input type="checkbox"/>	tgw-attach-0eb49a52d3036d736	VPC	vpc-0867c98f35d11fcdd	<input checked="" type="checkbox"/> Associated	

Figure 7-5: TGW-London-Default-RT – Associations Sheet.

The Propagations sheet shows that the propagation is enabled by default in the RT. For example, on-prem Customer Gateway (CGW) advertises Network Layer Reachability Information (NLRI) about subnet 172.16.10.0/24 to TGW using BGP Update messages over a secure VPN connection. After validating the BGP Update message, TGW propagates this (=installs) route to default RT.

Propagations (4) <small>Info</small>					
<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State	
<input type="checkbox"/>	tgw-attach-034c1731553b77b41	VPC	vpc-0a2c6c297c4562f11	<input checked="" type="checkbox"/> Enabled	
<input type="checkbox"/>	tgw-attach-06acc2b5d65998c07	VPC	vpc-04ef72cc79a73f82e	<input checked="" type="checkbox"/> Enabled	
<input type="checkbox"/>	tgw-attach-0c06d93f7b9933ffe	VPN	vpn-00ce8797da115243a	<input checked="" type="checkbox"/> Enabled	
<input type="checkbox"/>	tgw-attach-0eb49a52d3036d736	VPC	vpc-0867c98f35d11fcdd	<input checked="" type="checkbox"/> Enabled	

Figure 7-6: TGW-London-Default-RT – Propagations Sheet.

The Routes sheet in figure 7-7 verifies that TGW has installed all VPC CIDR ranges into RT and subnet that it has learned from the CGW. Note that TGW hasn't installed individual subnets (10.10.0.0/24, 10.11.0.0/24, 10.12.0.0/24). Remember that TGW accepts 1000 routes from its peer.

CIDR	Attachment ID	Resource ID	Resource ...	Route type
0.0.0.0/0	2 Attachments	2 Resources	VPN	Propagated
10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC	Propagated
10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC	Propagated
10.12.0.0/16	tgw-attach-034c1731553b77b41	vpc-0a2c6c297c4562f11	VPC	Propagated
172.16.10.0/24	2 Attachments	2 Resources	VPN	Propagated

Figure 7-7: TGW-London-Default-RT – Routes Sheet.

Example 7-1 shows the CGW's RT. It verifies that TGW has advertised VPC CIDR ranges to CGW over both VPN tunnels. CGW has first installed information into the BGP table and from there to its routing table.

NWKT-WAN-Edge-01#show ip bgp						
<snipped >						
Network	Next Hop	Metric	LocPrf	Weight	Path	
0.0.0.0	0.0.0.0			0	i	
*	10.10.0.0/16	169.254.241.209	100	0	64512 i	
*		169.254.175.105	100	0	64512 i	
*	10.11.0.0/16	169.254.241.209	100	0	64512 i	
*		169.254.175.105	100	0	64512 i	
*	10.12.0.0/16	169.254.241.209	100	0	64512 i	
*		169.254.175.105	100	0	64512 i	
*	172.16.10.0/24	0.0.0.0	0	32768	i	

Example 7-1: CGW's BGP Table.

NWKT-WAN-Edge-01#sh ip route bgp beg Gateway						
Gateway of last resort is 192.168.100.1 to network 0.0.0.0						
10.0.0.0/16 is subnetted, 3 subnets						
B	10.10.0.0 [20/100]	via 169.254.175.105, 00:19:14				
B	10.11.0.0 [20/100]	via 169.254.175.105, 00:19:14				
B	10.12.0.0 [20/100]	via 169.254.175.105, 00:19:14				

Example 7-2: CGW's Routing Table.

Create Route Table for Attachments

In this section, we create three TGW route tables: 1) a shared RT for VPCs NWKT-Prod and NWKT-test attachments, 2) dedicated RTs for VPC NWKT-Shared attachment, and 3) dedicated RT for VPN Connection attachment. Then we detach all four attachments from the default RT and associate them into correct route tables. Figure 7-8 illustrates the attachment-to-RT mapping, not the actual routes we later propagate into each RT.

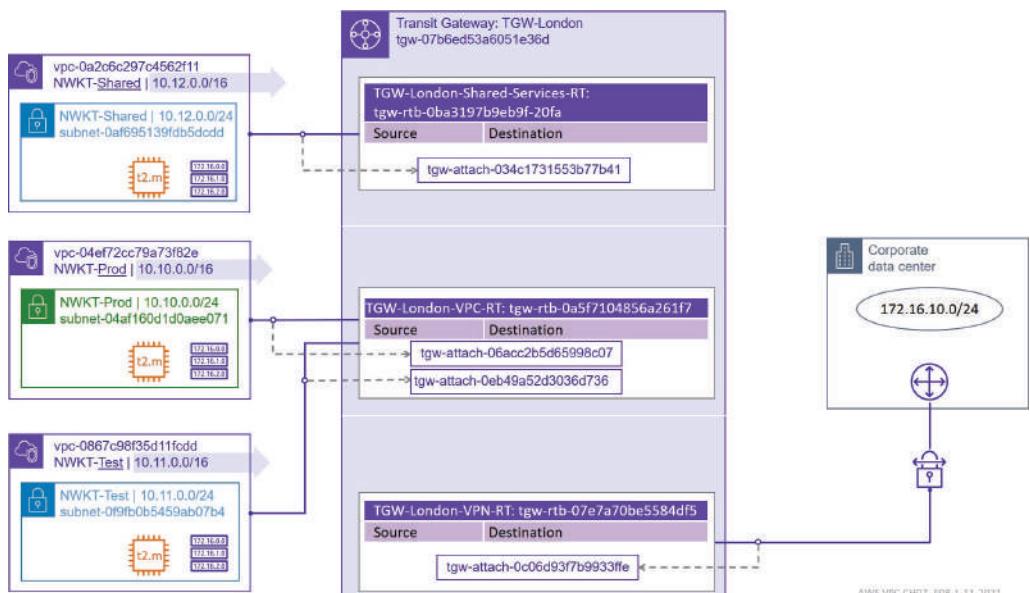


Figure 7-8: TGW's Route Table Model.

Create TGW Route Table

Navigate to the VPC dashboard and click Transit Gateway Route Tables under the Transit Gateway menu. Give the name to RT and select TGW from the Transit Gateway Id drop-down menu. Next, click the *Create transit gateway route table* button.

The screenshot shows the 'Create transit gateway route table' wizard. At the top, there's a breadcrumb navigation: VPC > Transit gateway route tables > Create transit gateway route table. To the right, a timestamp reads 'AWS VPC CH07-F09-1-11-2021'. The main section is titled 'Create transit gateway route table' with an 'Info' link. Below it, a sub-instruction says 'Use transit gateway route tables to configure routing for your transit gateway attachments.' A 'Details' tab is selected, showing a 'Name tag - optional' field containing 'TGW-London-VPC-RT'. A 'Transit gateway ID' dropdown menu is set to 'tgw-07b6ed53a6051e36d (TGW-London)'. The 'Tags' section follows, with a note about tags being labels for AWS resources. It includes a table for adding tags, where a single tag 'Name: TGW-London-VPC-RT' is listed. An 'Add new tag' button is available. A note says 'You can add 49 more tags.' At the bottom right are 'Cancel' and 'Create transit gateway route table' buttons, the latter being orange.

Figure 7-9: Create TGW Route Table.

Figure 7-10 shows the situation after we have created all three additional route tables. All route tables now have their unique TGW RT Ids. The Details tab shows the route table TGW-London-VPC specific information.

Transit gateway route tables (1/4) Info					
Actions Create transit gateway route table					
<input type="checkbox"/>	Name	Transit gateway route table ID	Transit gateway ID	State	⋮
<input type="checkbox"/>	TGW-London-Default-RT	tgw-rtb-0648db5cfa12dc1a2	tgw-07b6ed53a6051e36d	Available	Available
<input type="checkbox"/>	TGW-London-VPN-RT	tgw-rtb-07e7a70be55a84df5	tgw-07b6ed53a6051e36d	Available	Available
<input checked="" type="checkbox"/>	TGW-London-VPC-RT	tgw-rtb-0a5f7104856a261f7	tgw-07b6ed53a6051e36d	Available	Available
<input type="checkbox"/>	TGW-London-Shared-Service-RT	tgw-rtb-0ba3197b9eb9f20fa	tgw-07b6ed53a6051e36d	Available	Available

tgw-rtb-0a5f7104856a261f7 / TGW-London-VPC-RT

Details	Associations	Propagations	Prefix list references	Routes	Tags
Details					
Transit gateway route table ID tgw-rtb-0a5f7104856a261f7	State Available	Default association route table No	Default propagation route table No		
Transit gateway ID tgw-07b6ed53a6051e36d					

AWS VPC CH07- F11-1-11-2021

Figure 7-10: Transit Gateway Route Tables - Details.

Figure 7-11 verifies that there is no association with the RT at this phase.

tgw-rtb-0a5f7104856a261f7 / TGW-London-VPC-RT					
Details	Associations	Propagations	Prefix list references	Routes	Tags
Associations Info					
<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State	⋮
No associations This route table does not have any associated attachments.					
Create association					

AWS VPC CH07- F11-1-11-2021

Figure 7-11: Transit Gateway Route Tables - Associations.

Detach Attachments from the Default RT

Go to the Association sheet of TGW-London-Default-RT. Select the attachment and click the *Delete association* button.

Attachment ID	Resource type	Resource ID	State
<input checked="" type="checkbox"/> tgw-attach-06acc2b5d65998c07	VPC	vpc-04ef72cc79a73f82e	Associated
<input type="checkbox"/> tgw-attach-034c1731553b77b41	VPC	vpc-0a2c6c297c4562f11	Associated
<input type="checkbox"/> tgw-attach-0c06d93f7b9933ffe	VPN	vpn-00ce8797da115243a	Associated
<input type="checkbox"/> tgw-attach-0eb49a52d3036d736	VPC	vpc-0867c98f35d11fcdd	Associated

Figure 7-12: Transit Gateway Route Tables – Delete Associations.

Confirm that you want to delete the RT-to-attachments association by clicking the *Delete association* button.



Figure 7-13: Transit Gateway Route Tables – Delete Associations - Confirmation.

Figure 7-14 shows the attachment associations of TGW-London-Default-RT after we have deleted all associations.

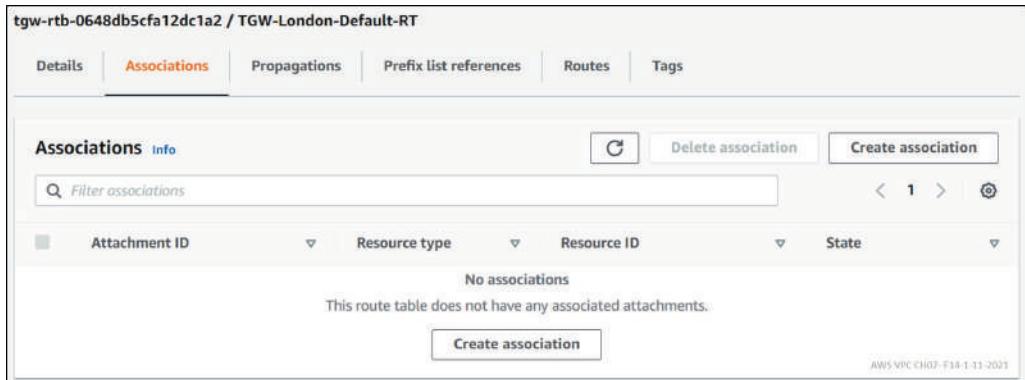


Figure 7-14: Transit Gateway Route Tables – Delete Associations - Verification.

If we now check the CGW's BGP table, there is no AWS VPC route in its BGP table.

```
NWKT-WAN-Edge-01#show ip bgp
BGP table version is 23, local router ID is 192.168.100.18
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
                  r RIB-failure, S Stale, m multipath, b backup-path, f RT-
Filter,
                  x best-external, a additional-path, c RIB-compressed,
                  t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network          Next Hop            Metric LocPrf Weight Path
  0.0.0.0          0.0.0.0
*>  172.16.10.0/24  0.0.0.0          0          32768  I
NWKT-WAN-Edge-01#sh ip route | beg Gateway
Gateway of last resort is 192.168.100.1 to network 0.0.0.0
S*  0.0.0.0/0 [254/0] via 192.168.100.1
    169.254.0.0/16 is variably subnetted, 4 subnets, 2 masks
C    169.254.175.104/30 is directly connected, Tunnel1
L    169.254.175.106/32 is directly connected, Tunnel1
C    169.254.241.208/30 is directly connected, Tunnel2
L    169.254.241.210/32 is directly connected, Tunnel2
    172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C    172.16.10.0/24 is directly connected, Loopback172
L    172.16.10.1/32 is directly connected, Loopback172
    192.168.100.0/24 is variably subnetted, 3 subnets, 2 masks
C    192.168.100.0/24 is directly connected, GigabitEthernet1
S    192.168.100.1/32 [254/0] via 192.168.100.1, GigabitEthernet1
L    192.168.100.18/32 is directly connected, GigabitEthernet1
```

Example 7-3: CGW BGP Table.

Associate Attachments with RT

Examples 7-15, 7-16, and 7-17 show the associations process of how we associate VPC attachments used with VPC NWKT-Test and NWKT-Prod with TGW-London-VPC-RT. Start the process by clicking the *Create association* button.

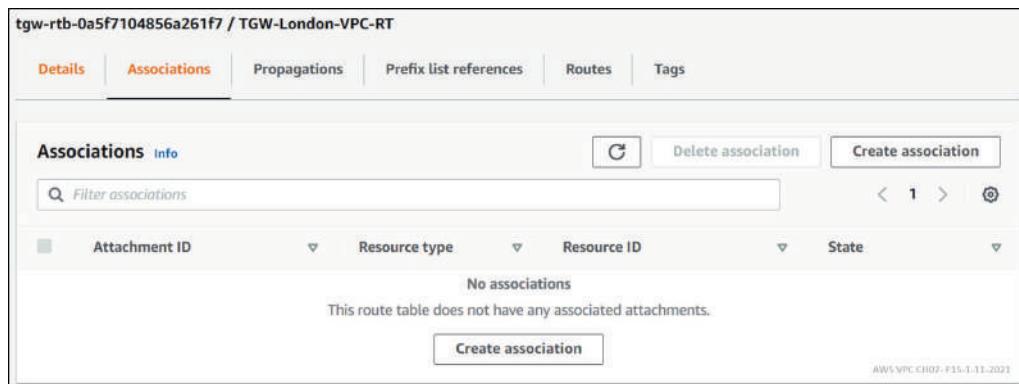


Figure 7-15: Transit Gateway Route Tables –Create Associations.

Select the VPC-Prod-Attach from the *Choose attachment to associate* drop-down menu. Then, click the *Create association* button. Repeat the association process with the attachment VPC-Test-Attach.

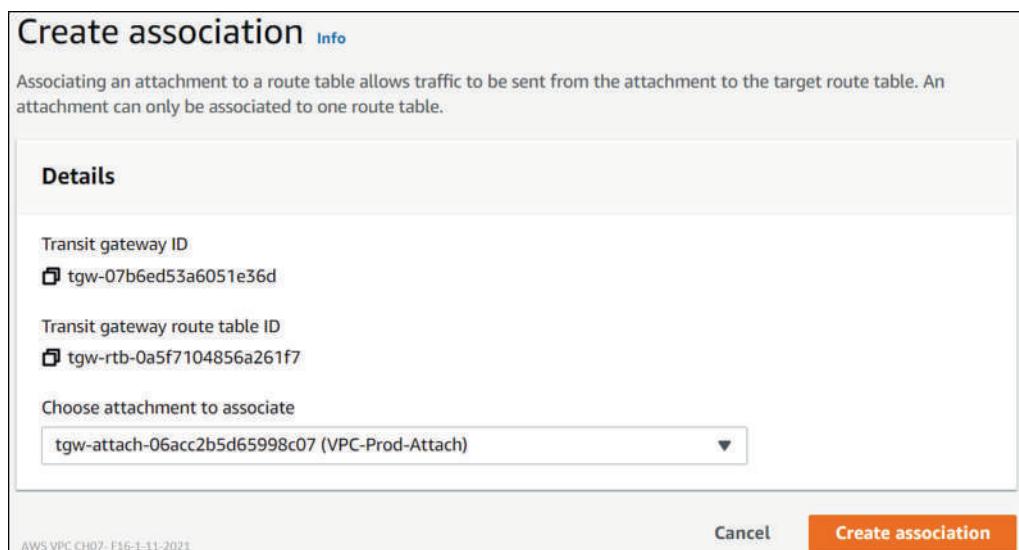


Figure 7-16: Transit Gateway Route Tables –Create Associations.

Figure 7-17 shows how the processes proceed from the *Associating* to *Associated*.

The screenshot shows a two-step process for creating associations between a Transit Gateway Router Table (RT) and VPCs.

Step 1: The top section shows two VPC attachments in the "Associating" state. The attachments are:

- tgw-attach-0eb49a52d3036d736 (VPC: vpc-0867c98f35d11fcdd)
- tgw-attach-06acc2b5d65998c07 (VPC: vpc-04ef7cc79a73f82e)

Step 2: The bottom section shows the same attachments now in the "Associated" state. The attachments are:

- tgw-attach-0eb49a52d3036d736 (VPC: vpc-0867c98f35d11fcdd)
- tgw-attach-06acc2b5d65998c07 (VPC: vpc-04ef7cc79a73f82e)

AWS VPC CH07-F17-1-11-2021

Figure 7-17: Transit Gateway Route Tables –Create Associations.

Figures 7-18 and 7-19 show associations of TGW-London-Shared-Services-RT and TGW-London-VPN-RT respectively.

The screenshot shows one association between a Transit Gateway Router Table (RT) and a VPC.

Association Details:

- Attachment ID: tgw-attach-034c1731553b77b41
- Resource type: VPC
- Resource ID: vpc-0a2c6c297c4562f11
- State: Associated

AWS VPC CH07-F18-1-11-2021

Figure 7-18: TGW RT TGW-London-Shared-Services-RT Associations.

The screenshot shows one association between a Transit Gateway Router Table (RT) and a VPN.

Association Details:

- Attachment ID: tgw-attach-0c06d93f7b9933ff
- Resource type: VPN
- Resource ID: vpn-00ce8797da115243a
- State: Associated

AWS VPC CH07-F19-1-11-2021

Figure 7-19: TGW RT TGW-London-VPN-RT Associations.

Route Table Propagation

In this section, we create IP connectivity between on-prem and AWS VPCs NWKT-Test and NWKT-Prod by updating TGW route tables. Figure 7-20 doesn't show VPC NWKT-Prod for simplicity though it is associated with the same RT as VPC NWKT-Test. We start by propagating routes, which TGW has learned from the CGW, into the route table TGW-London-VPC-RT. Then we repeat the process the other way around. After propagation, TGW starts routing ingress packets with the destination IP 172.16.10.xxx from VPC NWKT-Test (and NWKT-Prod) to VPN attachment tgw-attach-0c06d93f7b9933ffe. The next-hop for the ingress packets from the subnet 172.16.10.0/24 towards CIDR range 10.11.0.0/16 is the attachment associated with VPC NWKT-test.

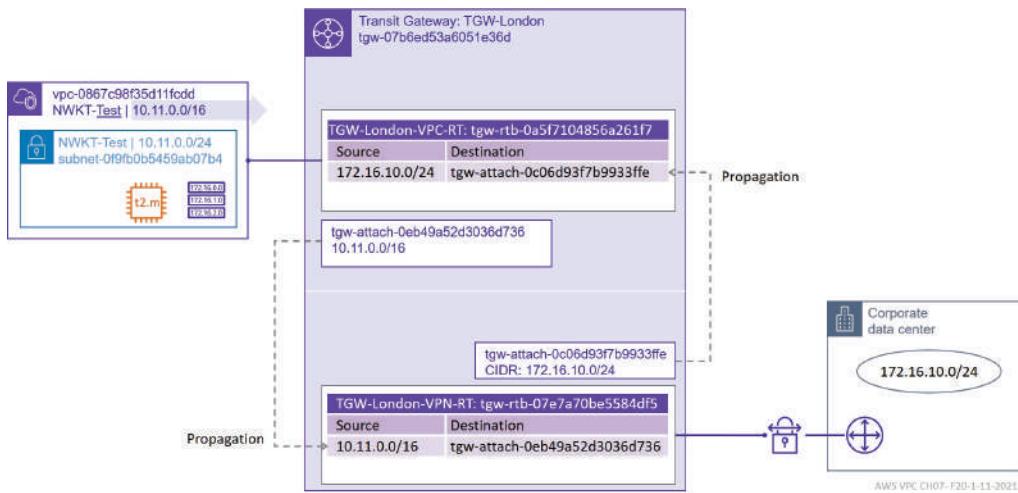


Figure 7-20: Route Propagation.

Create Propagation

First, we import routes to TGW-London-VPC-RT (figure 7-21). Navigate to the route table TGW-London-VPC-RT and select the Propagations sheet. Then, click the Create propagation button.

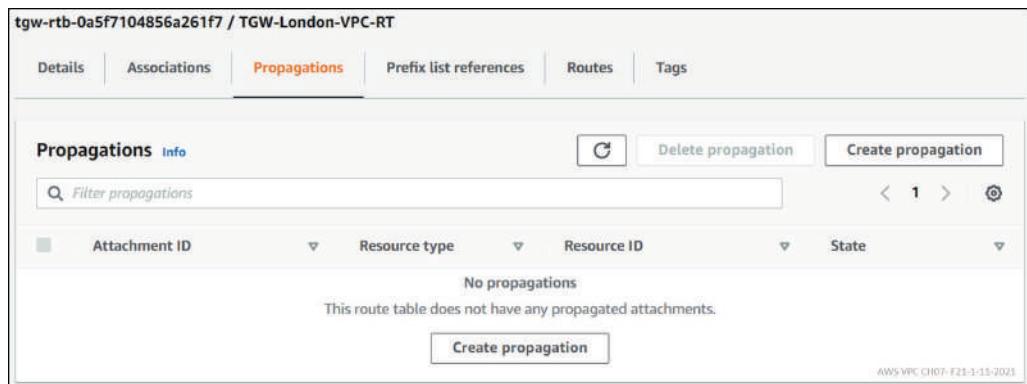


Figure 7-21: Route Propagation – Phase-1.

Select the VPN-to-Datacenter (tgw-attach-0c06d93f7b9933ffe) from the *Choose attachment to propagate* drop-down menu (figure 7-22). Click the *Create propagation* button to proceed.

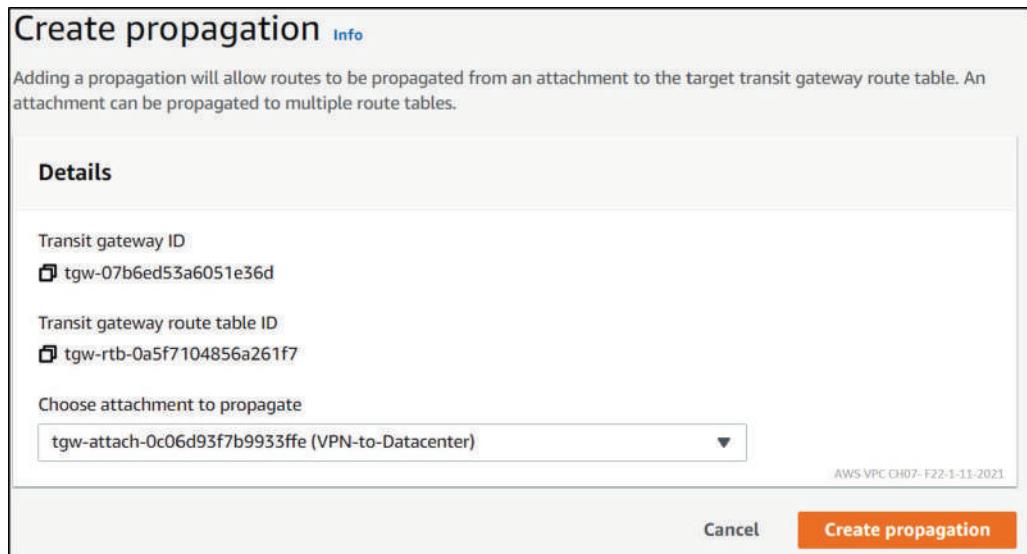


Figure 7-22: Route Propagation – Phase-2.

Figure 7-23 shows that now the attachment `tgw-attach-0c06d93f7b9933ffe` is listed on the propagation sheet.

Attachment ID	Resource type	Resource ID	State
<code>tgw-attach-0c06d93f7b9933ffe</code>	VPN	<code>vpn-00ce8797da115243a</code>	Enabled

Figure 7-23: Route Propagation – Phase-3.

Figure 7-24 shows what propagation did. The Routes sheet shows that the next-hop for CIDR `172.16.10.0/24` points to the VPN connection attachment. Note that figure 7-24 shows routing entries about twice. That way, I was able to display both attachment ids and resource ids. Example 7-4 verifies that we still need to propagate routes into TGW-London-VPN-RT's route table to get a bi-directional connection.

CIDR	Attachment ID	Resource ID	Resource type	Route
<code>0.0.0.0/0</code>	2 Attachments	2 Resources	VPN	Prc
<code>172.16.10.0/24</code>	2 Attachments	<code>tgw-attach-0c06d93f7b9933ffe</code>	VPN	Prc
		<code>tgw-attach-0c06d93f7b9933ffe</code>		

Figure 7-24: Route Propagation – Phase-4.

```
NWKT-WAN-Edge-01#sh ip route bgp | beg Gateway
Gateway of last resort is 192.168.100.1 to network 0.0.0.0
NWKT-WAN-Edge-01#
```

Example 7-4: Route Propagation – The First Check on CGW.

Next, we propagate routes from the attachment VPC-Test-Attach to TGW-London-VPN-RT (tgw-rtb-07e7a70be55a84df5) using the same process we did in the previous example. Go to the TGW-London-VPN-RT, select the *Propagations* sheet, and click the *Create propagation* button.

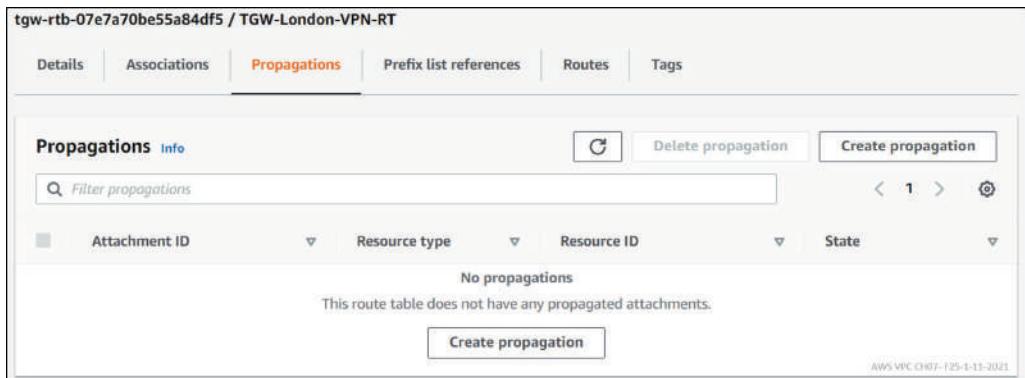


Figure 7-25: Route Propagation – Phase-5.

Select the VPC-Test-Attach from the Choose attachment to propagate from the drop-down menu. Next, click the Create propagation button.

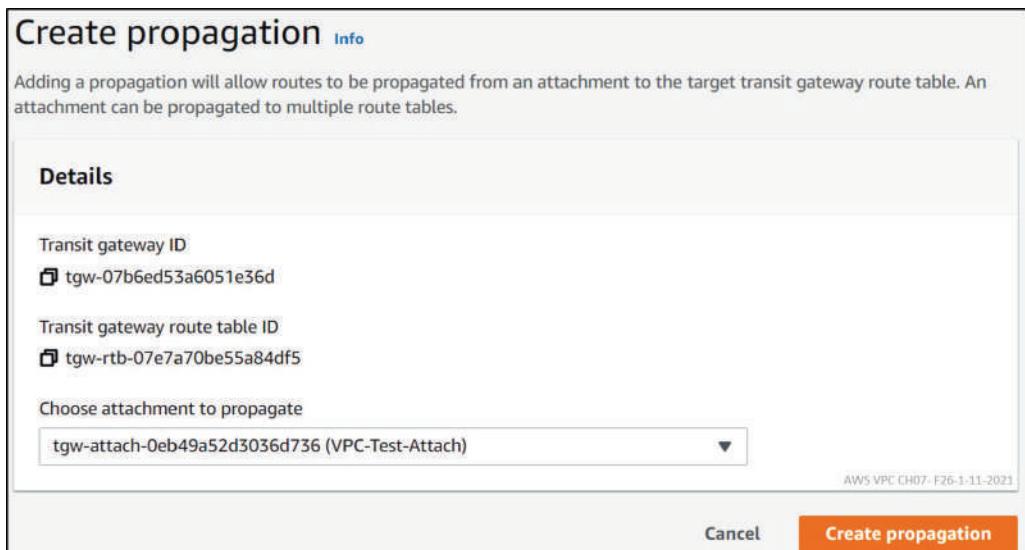


Figure 7-26: Route Propagation – Phase-6.

Now we can see that the attachment tgw-attach-0eb49a52d3036d736 is listed in the Propagations sheet.

Attachment ID	Resource type	Resource ID	State
tgw-attach-0eb49a52d3036d736	VPC	vpc-0867c98f35d11fcdd	Enabled

Figure 7-27: Route Propagation – Phase-7.

We can also see that there is routing information about CIDR 10.11.0.0/16 in the Routes tab.

CIDR	Attachment ID	Resource ID	Route type
10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	Propagated

Figure 7-28: Route Propagation – Phase-8.

Example 7-5 verifies that TGW starts advertising CIDR 10.11.0.0/16 NLRI to CGW after installing route to TGW-London-VPN-RT.

```
NWKT-WAN-Edge-01#sh ip route bgp | beg Gateway
Gateway of last resort is 192.168.100.1 to network 0.0.0.0
    10.0.0.0/16 is subnetted, 1 subnets
B        10.11.0.0 [20/100] via 169.254.241.209, 00:00:30
NWKT-WAN-Edge-01#
```

Example 7-5: Route Propagation – The Second Check on CGW.

Figures 7-29, and 7-30 show the route propagations and routes in the RT TGW-London-VPN-RT after route propagation related to VPC NWKT-Prod attachment.

tgw-rtb-07e7a70be55a84df5 / TGW-London-VPN-RT						AWS VPC CH07-F29-1-11-2021
Details	Associations	Propagations	Prefix list references	Routes	Tags	
Propagations (2) <small>Info</small>						<input type="button" value="Delete propagation"/> <input type="button" value="Create propagation"/>
<input type="button" value="Filter propagations"/>						< 1 >
<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State		
<input type="checkbox"/>	tgw-attach-06acc2b5d65998c07	VPC	vpc-04ef72cc79a73f82e	Enabled		
<input type="checkbox"/>	tgw-attach-0eb49a52d3036d736	VPC	vpc-0867c98f35d11fcdd	Enabled		

Figure 7-29: Route Propagation – Phase-8.

tgw-rtb-07e7a70be55a84df5 / TGW-London-VPN-RT						AWS VPC CH07-F30-1-11-2021
Details	Associations	Propagations	Prefix list references	Routes	Tags	
Routes (2)						<input type="button" value="Actions"/> <input type="button" value="Create static route"/>
<input type="button" value="Filter routes"/>						< 1 >
<input type="checkbox"/>	CIDR	Attachment ID	Resource ID	Resource ...	Route type	
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC	Propagated	
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC	Propagated	

Figure 7-30: Route Propagation – Phase-9.

Example 7-6 shows that CGW has installed route to 10.10.0.0/16 to its routing table.

```
NWKT-WAN-Edge-01#sh ip route bgp | beg Gateway
Gateway of last resort is 192.168.100.1 to network 0.0.0.0
    10.0.0.0/16 is subnetted, 2 subnets
B        10.10.0.0 [20/100] via 169.254.241.209, 00:00:37
B        10.11.0.0 [20/100] via 169.254.241.209, 00:02:37
```

Example 7-6: Route Propagation – The Third Check on CGW.

However, we still can't ping from the on-prem Datacenter CGW to our example EC2 instance running on VPC NWKT-Prod (subnet 10.10.0.0/24).

```
NWKTK-WAN-Edge-01#ping 10.10.0.218 source 172.16.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.0.218, timeout is 2 seconds:
Packet sent with a source address of 172.16.10.1
.....
Success rate is 0 percent (0/5)NWKTK-WAN-Edge-01#
```

Example 7-7: Route Propagation – The Fifth Check on CGW.

The reason for this is that there is no route back to 172.16.10.0/24 in the subnet NWKT-Prod (subnet-04af160d1d0aee071) routing table (NWKT-Prod-Public/rtb-05dc9cd679937ceee).

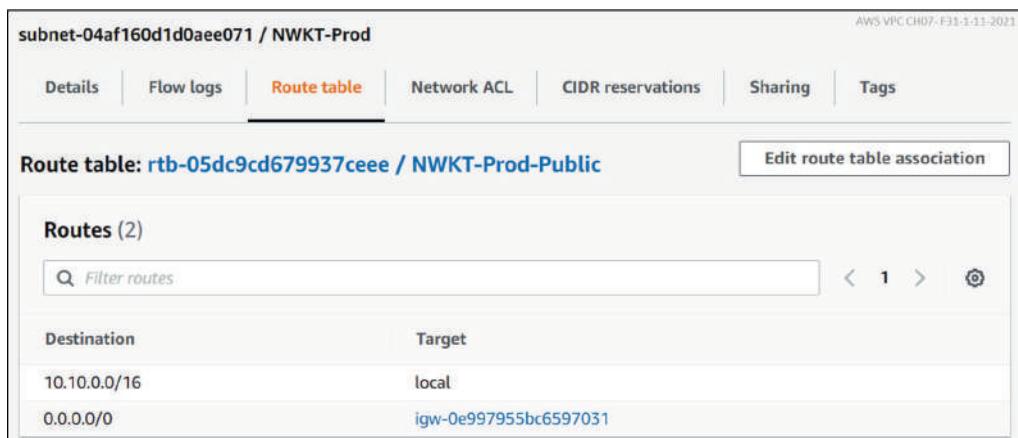


Figure 7-31: Route Table of Subnet NWKT-Prod.

Navigate to the route table NWKT-Prod-Public, and select the Routes sheet. Next, click the Edit routes button.

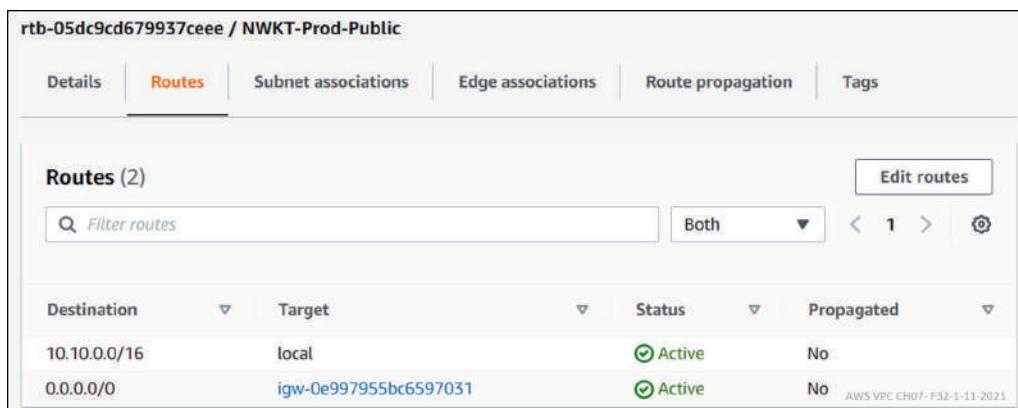


Figure 7-32: Route Table of Subnet NWKT-Prod.

After we have created a static route to 172.16.10.0/24 with TGW-London (tgw-07b6ed53a6051e36d) as a next-hop (figure 7-33), ping from 172.16.10.1 (CGW) to 10.10.0.218 (EC2 instance) start working.

Routes (3)				Edit routes
Destination	Target	Status	Propagated	
172.16.10.0/24	tgw-07b6ed53a6051e36d	✓ Active	No	
10.10.0.0/16	local	✓ Active	No	
0.0.0.0/0	igw-0e997955bc6597031	✓ Active	No	AWS VPC CH07- F33-1-11-2021

Figure 7-33: Route Table of Subnet NWKT-Prod.

```
NWKT-WAN-Edge-01#ping 10.10.0.218 source 172.16.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.0.218, timeout is 2 seconds:
Packet sent with a source address of 172.16.10.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 35/35/36 ms
NWKT-WAN-Edge-01#
```

Example 7-8: Route Propagation – The Sixth Check on CGW - Pass.

At this phase, we have done half of the jobs. We still need to establish a connection from VPCs NWKT-Prod and NWKT-Test to VPC NWKT-Shared. The process is the same we did in previous examples. Figures 7-34 verifies that we have propagated routes that TGW has learned via attachment tgw-attach-034c1731553b77b41 (=NWKT-Shared/vpc-0a2c6c297c4562f11) into the route table TGW-London-VPC-RT.

tgw-rtb-0a5f7104856a261f7 / TGW-London-VPC-RT						AWS VPC CH07- F33-1-11-2021
Details	Associations	Propagations	Prefix list references	Routes	Tags	
Propagations (2) Info						
<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State		
<input type="checkbox"/>	tgw-attach-034c1731553b77b41	VPC	vpc-0a2c6c297c4562f11	✓ Enabled		
<input type="checkbox"/>	tgw-attach-0c06d93f7b9933ffe	VPN	vpn-00ce8797da115243a	✓ Enabled		

Figure 7-34: Shared-Service Route Propagation into the VPC RT - Propagation.

Figure 7-35, in turn, shows that TGW has also installed route toward 10.12.0.0/16 into the TGW-London-VPC-RT. The next-hop is the attachment tgw-attach-034c1731553b77b41 (attachment related to VPC NWKT-Shared).

tgw-rtb-0a5f7104856a261f7 / TGW-London-VPC-RT

AWS VPC CH07: F35-1-11-2021

Details Associations Propagations Prefix list references **Routes** Tags

▶ Filter routes by CIDR (2)

Routes (3)

Filter routes Actions Create static route

CIDR	Attachment ID	Resource ID	Resource type
0.0.0.0/0	2 Attachments	2 Resources	VPN
10.12.0.0/16	tgw-attach-034c1731553b77b41	vpc-0a2c6c297c4562f11	VPC
172.16.10.0/24	2 Attachments	2 Resources	VPN

Figure 7-35: Shared-Service Route Propagation into the VPC RT - Routes.

Figures 7-36, and 7-37 show the propagation process and routes from the TGW-London-Shared-Service-RT perspective.

tgw-rtb-0ba3197b9eb9f20fa / TGW-London-Shared-Service-RT

AWS VPC CH07: F36-1-11-2021

Details Associations **Propagations** Prefix list references Routes Tags

Propagations (2) Info

Filter propagations Delete propagation Create propagation

Attachment ID	Resource type	Resource ID	State
tgw-attach-06acc2b5d65998c07	VPC	vpc-04ef72cc79a73f82e	Enabled
tgw-attach-0eb49a52d3036d736	VPC	vpc-0867c98f35d11fcdd	Enabled

Figure 7-36: VPC RT Propagation into the Shared-Service Route - Propagation.

tgw-rtb-0ba3197b9eb9f20fa / TGW-London-Shared-Service-RT						AWS VPC CH07-F37-1-13-2021															
Details	Associations	Propagations	Prefix list references	Routes	Tags																
► Filter routes by CIDR (2)																					
Routes (2) <div style="display: flex; justify-content: space-between;"> <input type="button" value="⟳"/> <input type="button" value="Actions ▾"/> <input type="button" value="Create static route"/> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th><input type="checkbox"/></th> <th>CIDR</th> <th>Attachment ID</th> <th>Resource ID</th> <th>Resource type</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>10.10.0.0/16</td> <td>tgw-attach-06acc2b5d65998c07</td> <td>vpc-04ef72cc79a73f82e</td> <td>VPC</td> </tr> <tr> <td><input type="checkbox"/></td> <td>10.11.0.0/16</td> <td>tgw-attach-0eb49a52d3036d736</td> <td>vpc-0867c98f35d11fcdd</td> <td>VPC</td> </tr> </tbody> </table>							<input type="checkbox"/>	CIDR	Attachment ID	Resource ID	Resource type	<input type="checkbox"/>	10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC	<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC
<input type="checkbox"/>	CIDR	Attachment ID	Resource ID	Resource type																	
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC																	
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC																	

Figure 7-37: VPC RT Propagation into the Shared-Service Route - Routes.

Figure 7-38 verifies that routes propagated from the TGW-London-Shared-Service-RT into TGW-London-VPC-RT are not propagated further to TGW-London-VPN-RT. It is like the iBGP routing policy in traditional networking. BGP learned NLRIIs learned from one iBGP peer are not advertised to another iBGP peer. Also, routes imported from one VRF to another VRF are not exported out of the VRF that imports routes. These processes prevent routing update loops.

tgw-rtb-07e7a70be55a84df5 / TGW-London-VPN-RT						AWS VPC CH07-F38-1-13-2021															
Details	Associations	Propagations	Prefix list references	Routes	Tags																
► Filter routes by CIDR (2)																					
Routes (2) <div style="display: flex; justify-content: space-between;"> <input type="button" value="⟳"/> <input type="button" value="Actions ▾"/> <input type="button" value="Create static route"/> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th><input type="checkbox"/></th> <th>CIDR</th> <th>Attachment ID</th> <th>Resource ID</th> <th>Resource type</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>10.10.0.0/16</td> <td>tgw-attach-06acc2b5d65998c07</td> <td>vpc-04ef72cc79a73f82e</td> <td>VPC</td> </tr> <tr> <td><input type="checkbox"/></td> <td>10.11.0.0/16</td> <td>tgw-attach-0eb49a52d3036d736</td> <td>vpc-0867c98f35d11fcdd</td> <td>VPC</td> </tr> </tbody> </table>							<input type="checkbox"/>	CIDR	Attachment ID	Resource ID	Resource type	<input type="checkbox"/>	10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC	<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC
<input type="checkbox"/>	CIDR	Attachment ID	Resource ID	Resource type																	
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC																	
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC																	

Figure 7-38: Route Table TGW-London-VPN-RT.

CGW on the on-prem Datacenter doesn't have a route to 10.12.0.0/16 due to the build-in route-propagation loop prevention mechanism.

```
NWKT-WAN-Edge-01#sh ip route bgp | beg Gateway
Gateway of last resort is 192.168.100.1 to network 0.0.0.0
    10.0.0.0/16 is subnetted, 2 subnets
B        10.10.0.0 [20/100] via 169.254.241.209, 00:13:30
B        10.11.0.0 [20/100] via 169.254.241.209, 00:15:30
NWKT-WAN-Edge-01#
```

Example 7-9: Route Table of CGW.

As the last step, we need to add static routes to subnet-specific route tables. Figure 7-39 shows routing entries in the subnet NWKT-Prod RT. Figure 7-40 shows routing entries in the NWKT-Test RT. Figure 7-40 shows routing entries in the NWKT-Test RT. Figure 7-41 shows routing entries in the NWKT-Shared RT.

subnet-04af160d1d0aee071 / NWKT-Prod		AWS VPC CH07-F99-1-11-2021														
Details	Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags										
Route table: rtb-05dc9cd679937ceee / NWKT-Prod-Public						Edit route table association										
Routes (4)																
<input type="text"/> Filter routes < 1 > ⌂																
<table border="1"> <thead> <tr> <th>Destination</th><th>Target</th></tr> </thead> <tbody> <tr> <td>172.16.10.0/24</td><td>tgw-07b6ed53a6051e36d</td></tr> <tr> <td>10.12.0.0/24</td><td>tgw-07b6ed53a6051e36d</td></tr> <tr> <td>10.10.0.0/16</td><td>local</td></tr> <tr> <td>0.0.0.0/0</td><td>igw-0e997955bc6597031</td></tr> </tbody> </table>							Destination	Target	172.16.10.0/24	tgw-07b6ed53a6051e36d	10.12.0.0/24	tgw-07b6ed53a6051e36d	10.10.0.0/16	local	0.0.0.0/0	igw-0e997955bc6597031
Destination	Target															
172.16.10.0/24	tgw-07b6ed53a6051e36d															
10.12.0.0/24	tgw-07b6ed53a6051e36d															
10.10.0.0/16	local															
0.0.0.0/0	igw-0e997955bc6597031															

Figure 7-39: Routing Entries in Route Table NWKT-Prod-Public.

subnet-0f9fb0b5459ab07b4 / NWKT-Test		AWS VPC CH07-F99-1-11-2021														
Details	Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags										
Route table: rtb-0f9540b028f439ac7 / NWKT-Test						Edit route table association										
Routes (4)																
<input type="text"/> Filter routes < 1 > ⌂																
<table border="1"> <thead> <tr> <th>Destination</th><th>Target</th></tr> </thead> <tbody> <tr> <td>172.16.10.0/24</td><td>tgw-0cefc925035437d16</td></tr> <tr> <td>10.12.0.0/24</td><td>tgw-07b6ed53a6051e36d</td></tr> <tr> <td>10.11.0.0/16</td><td>local</td></tr> <tr> <td>0.0.0.0/0</td><td>tgw-0cefc925035437d16</td></tr> </tbody> </table>							Destination	Target	172.16.10.0/24	tgw-0cefc925035437d16	10.12.0.0/24	tgw-07b6ed53a6051e36d	10.11.0.0/16	local	0.0.0.0/0	tgw-0cefc925035437d16
Destination	Target															
172.16.10.0/24	tgw-0cefc925035437d16															
10.12.0.0/24	tgw-07b6ed53a6051e36d															
10.11.0.0/16	local															
0.0.0.0/0	tgw-0cefc925035437d16															

Figure 7-40: Routing Entries in Route Table NWKT-Prod-Public.

subnet-0af695139fdb5dcdd / NWKT-Shared		AWS VPC CH07-F43-1-11-2021												
Details	Flow logs	Route table	Network ACL	CIDR reservations	Sharing	Tags								
Route table: rtb-09b1ee666c7939745 / NWKT-Shared					Edit route table association									
Routes (3)														
<input type="text"/> Filter routes														
<table border="1"> <thead> <tr> <th>Destination</th><th>Target</th></tr> </thead> <tbody> <tr> <td>10.10.0.0/24</td><td>tgw-07b6ed53a6051e36d</td></tr> <tr> <td>10.11.0.0/24</td><td>tgw-07b6ed53a6051e36d</td></tr> <tr> <td>10.12.0.0/16</td><td>local</td></tr> </tbody> </table>							Destination	Target	10.10.0.0/24	tgw-07b6ed53a6051e36d	10.11.0.0/24	tgw-07b6ed53a6051e36d	10.12.0.0/16	local
Destination	Target													
10.10.0.0/24	tgw-07b6ed53a6051e36d													
10.11.0.0/24	tgw-07b6ed53a6051e36d													
10.12.0.0/16	local													

Figure 7-41: Routing Entries in Route Table NWKT-Shared.

Summary

Figure 7-42 illustrates pieces of puzzles that we used to implement the our VPC segmentation with Transit Gateway. Our segmentation policy stated that services running within the VPC NWKT-Shared has to be accessed from VPCs NWKT-Prod, and NWKT-Test. Traffic flows between VPCs NWKT-Test, and NWKT-Prod, however, is restricted. Service we are running in on-prem Datacenter are accessible from VPCs NWKT-Prod, and NWKT-Test but not from the VPC NWKT-Shared.

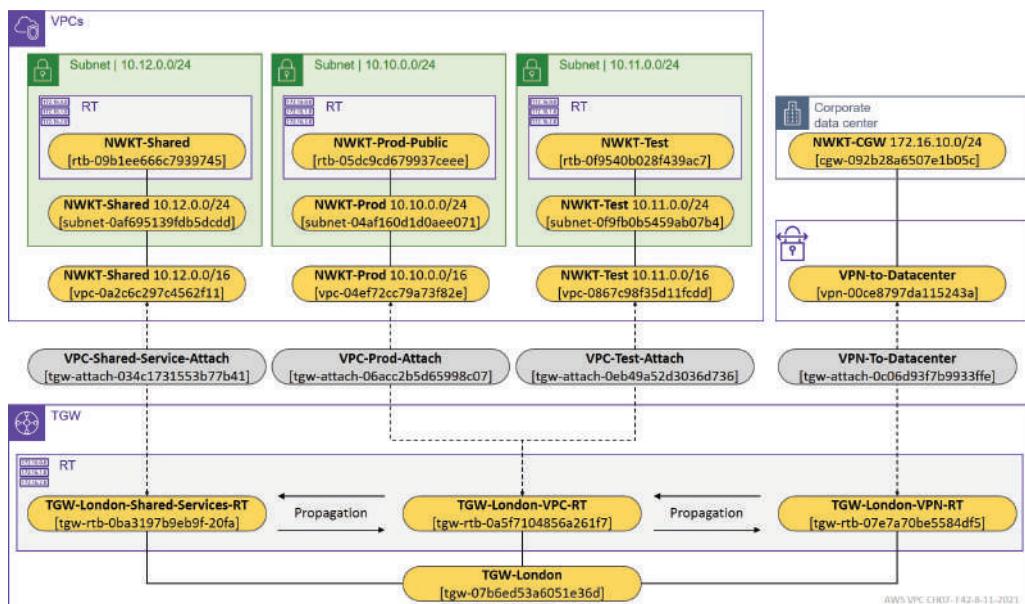


Figure 7-42: Segmentation with TGW Route Tables Summary Diagram.

Chapter 8: Transit Gateway Peering

Introduction

The previous chapter explained how we implement a VPC segmentation policy with Transit Gateway using route tables. This chapter introduces how we can connect VPCs in different AWS regions using TGW peering. Figure 8-1 illustrates an example environment that we are using in this chapter. Our first intent is to allow traffic flows between VPC NWKT-Prod in AWS Region eu-west-2 (London) and VPC Stockholm-VPC in AWS region eu-north-1 (Stockholm). Our second need is to allow packet flows from the on-prem Datacenter connected to NWKT-TGW (via VPN) to both VPCs NWKT-Prod and Stockholm-VPC. London TGW has dedicated route tables for VPC, VPN, and TGW Peering. Stockholm-TGW, in turn, uses the default route table for both VPC and TGW Peering.

When setting up the peering connection between TGWs, we send a peering request from Stockholm TGW to London TGW (1a). Then, we accept the peering request on London TGW (1b). TGW Peering, at the time of writing, supports only static routes. As the next step, we add static routes towards subnets 10.10.0.0/16 and 172.16.10.0/24 into the Stockholm-TGW-Default-RT (2a) with peering attachment as a next-hop. We also add static routes for these two subnets in the London TGW route table Stockholm-London-RT but with their respective attachments. We also need to add static routes to 10.10.0.0/16 and 172.16.10.0/24 into subnet Stockholm-Subnet-1's route table (3a). Besides, we have to add a static route toward 10.100.0.0/16 into subnet NWKT-Prod's route table (3b). The next-hop is the local TGW attachment (not the TGW peering attachment) in both examples. Even though I previously mentioned that TGW peering doesn't support propagation, there is one exception. All routes in the route table related to VPN connection are advertised to CGW automatically (3c).

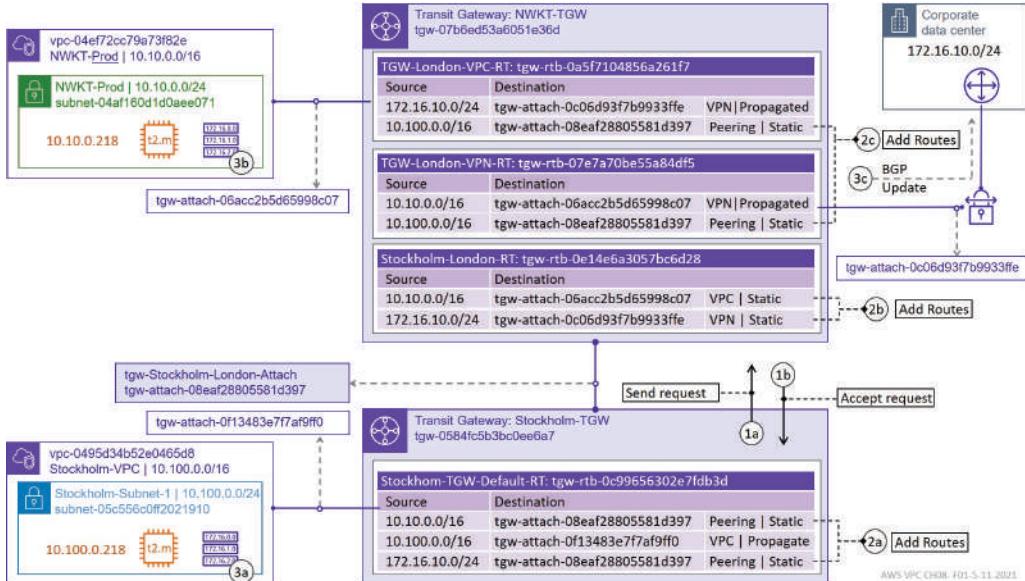


Figure 8-1: TGW peering Example Environment.

Before moving on to the Transit Gateway Peering section, let's look at the Stockholm Region resources that I have already put in place. Figure 8-2 shows our VPC Stockholm-VPC and its CIDR range of 10.100.0.0/16.

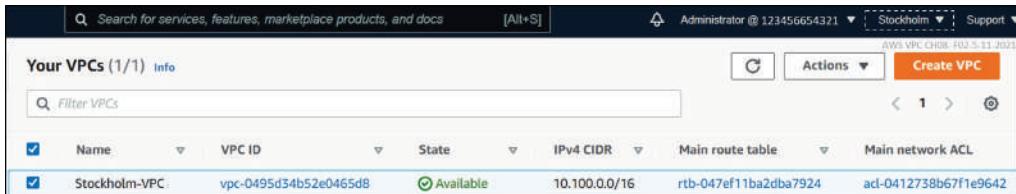


Figure 8-2: Stockholm-VPC: Configuration.

Within Stockholm-VPC, we have a subnet 10.100.0.0/24 (Stockholm-Subnet-1). I have launched an EC2 instance (IP address 10.100.0.218) in this subnet.

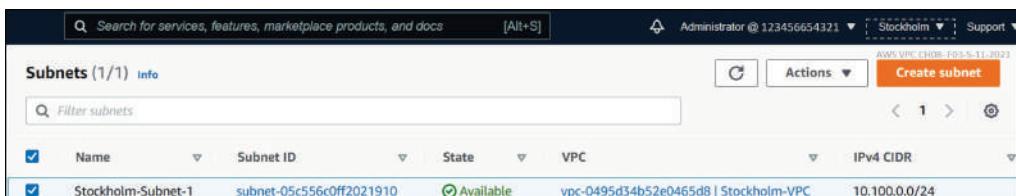


Figure 8-3: Stockholm-VPC: Subnet.

We have a pre-configured TGW, Stockholm-TGW on Stockholm Region. I have attached the Stockholm-VPC to the default route table of the TGW. The VPC-to-TGW association is also shown in figure 8-5.

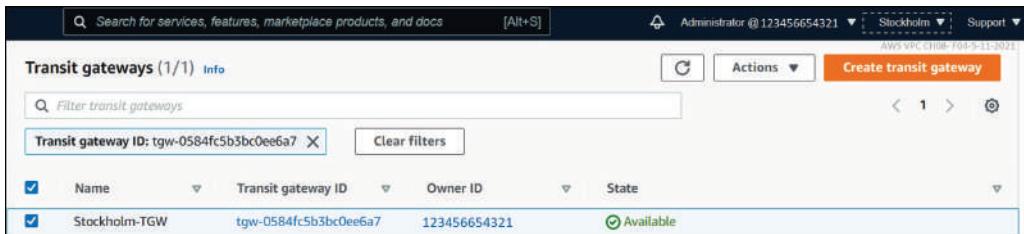


Figure 8-4: Stockholm-VPC: Transit Gateway.

Create TGW Peering

The TGW peering configuration has two phases. As a first step, we create a TGW attachment using the resource type *Peering Connection* in Stockholm-TGW. During the process, we define the remote AWS region (London) and TGW, which we want to peer (London-TGW). In the second phase, the administrator of the London-TGW has to accept our peering request.

TGW Peering Connection Request (Stockholm-TGW)

Navigate to the Transit Gateway Attachment under the Transit Gateways header (figure 6-3 in chapter 6). Figure 8-5 shows that we have already attached the Stockholm-VPC to Stockholm-TGW. To start the TGW peering configuration, click the *Create transit gateway attachment* button.

The screenshot shows the AWS CloudFormation console with the following details:

- Transit gateway attachments (1/1) Info**
- Actions** button
- Create transit gateway attachment** button
- Filter transit gateway attachments** input field
- Transit gateway attachment ID: tgw-attach-Of13483e7f7af9ff0 X**
- Clear filters** button
- Table Headers:** Name, Transit gateway attachment ID, Transit gateway ID, Resource type, Resource ID
- Table Data:**

<input checked="" type="checkbox"/>	Stockholm-VPC-Attach	tgw-attach-Of13483e7f7af9ff0	tgw-0584fc5b3bc0ee6a7	VPC	vpc-0495d34b52e0465d8
-------------------------------------	----------------------	------------------------------	-----------------------	-----	-----------------------
- Attachment Details View:**
 - Name tag:** tgw-attach-Of13483e7f7af9ff0 / Stockholm-VPC-Attach
 - Details Tab:** Shows the following configuration:

Transit gateway attachment ID: tgw-attach-Of13483e7f7af9ff0	State: Available	Resource type: VPC	Association state: Associated
Transit gateway ID: tgw-0584fc5b3bc0ee6a7	Resource owner ID: 123456654321	Resource ID: vpc-0495d34b52e0465d8	Association route table ID: tgw-rtb-0c99656302e7fdb3d
Transit gateway owner ID: 123456654321	DNS support: Enable	IPv6 support: Disable	Subnet IDs: subnet-05c556c0ff2021910
 - Tags Tab:** Not visible in the screenshot.
- AWS VPC CH08- F05-5-11-2021** timestamp at the bottom right.

Figure 8-5: TGW Peering Configuration on Stockholm-TGW: Step-1.

The following three figures are all in the same window, but I have divided those into three separate figures. First, we fill the *Details* section. Give the *Name tag* (optional) for peering connection. Then select the Stockholm-TGW (requester) from the *Transit gateway ID* drop-down menu. Next, select the *Peering Connection* from the *Attachment type* drop-down menu.

The screenshot shows the 'Create transit gateway attachment' dialog with the following fields:

- Title:** Create transit gateway attachment [Info](#)
- Description:** A transit gateway (TGW) is a network transit hub that interconnects attachments (VPCs and VPNs) within the same AWS account or across AWS accounts.
- Details Section:**
 - Name tag - optional:** Creates a tag with the key set to Name and the value set to the specified string. Value: Stockholm-London-Attach
 - Transit gateway ID:** [Info](#) dropdown menu showing: tgw-0584fc5b3bc0ee6a7 (Stockholm-TGW)
 - Attachment type:** [Info](#) dropdown menu showing: Peering Connection
- AWS VPC CH08- F06-5-11-2021** timestamp at the bottom right.

Figure 8-6: TGW Peering Configuration on Stockholm-TGW: Step-2.1.

Next, we move to the Peering connection attachment section. I have created London TGW using the same account as in Stockholm, so we check the *My account* radio button. Then we select Europe (London) (eu-west-2) from the *Region* drop-down menu. Next, you navigate to the London VPC dashboard and copy the London TGW id from there and paste it to the *Transit gateway (accepter)* field.

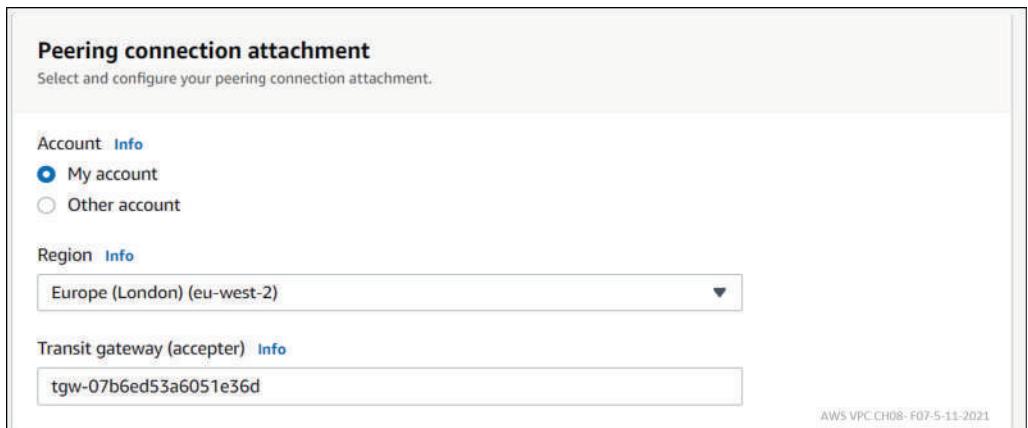


Figure 8-7: TGW Peering Configuration on Stockholm-TGW: Step-2.2.

The name we gave in the Details section is shown in the *Tags* section. You can add more tags if needed. When you have given all the information, click the *Create transit gateway attachment* button.

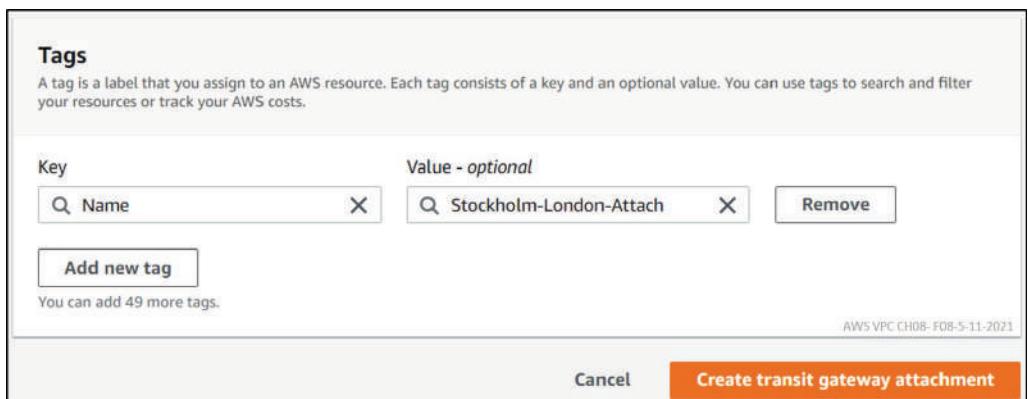


Figure 8-8: TGW Peering Configuration on Stockholm-TGW: Step-2.3.

Figure 8-9 shows that before we accept the connection request on London TGW, the State on Stockholm is *Pending Acceptance*. The Details section shows Requester TGW ID (tgw-0584fc5b3bc0ee6a7/Stockholm-TGW) and the region (Stockholm (eu-north-1). Besides, you can see the same information related to Acceptor TGW.

Name	Transit gateway attachment ID	Transit gateway ID	Resource type	Resource ID
Stockholm-London-Attach	tgw-attach-08eaf28805581d397	tgw-0584fc5b3bc0ee6a7	Peering	tgw-07b6ed53a6051e36d

Details

Transit gateway attachment ID tgw-attach-08eaf28805581d397	State Pending Acceptance	Resource type Peering	Association route table ID -
Requester transit gateway ID tgw-0584fc5b3bc0ee6a7	Requester region Stockholm (eu-north-1)	Requester owner ID 123456654321	Association state -
Acceptor transit gateway ID tgw-07b6ed53a6051e36d	Acceptor region London (eu-west-2)	Acceptor owner ID 123456654321	

Figure 8-9: TGW Peering Configuration on Stockholm-Pending Acceptance.

TGW Attachment - London: Accept

To finalize the TGW peering, select London from the AWS Region selection drop-down menu from the main bar (black background). Then navigate to the Transit gateway attachments window. You can see that the *Stockholm-London-Attach* state is *Pending Acceptance*. After selecting it, choose the *Accept transit gateway attachment* from the Actions drop-down menu.

Name	Transit gateway attachment ID	Transit gateway ID	Association route table
<input checked="" type="checkbox"/> Stockholm-London-Attach	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	
<input type="checkbox"/>	tgw-attach-01c6bcfa2cfae7da0	tgw-0cef925035437d16	Peering
<input type="checkbox"/>	tgw-attach-004c075de4de5f722	tgw-07b6ed53a6051e36d	VPC

tgw-attach-08eaf28805581d397 / Stockholm-London-Attach

Details Tags

Details

Transit gateway attachment ID tgw-attach-08eaf28805581d397	State Pending Acceptance	Resource type Peering	Association route table ID -
Requester transit gateway ID tgw-0584fc5b3bc0ee6a7	Requester region Stockholm (eu-north-1)	Requester owner ID 123456654321	Association state -
Acceptor transit gateway ID tgw-07b6ed53a6051e36d	Acceptor region London (eu-west-2)	Acceptor owner ID 123456654321	

AWS VPC CH08- F10-5-11-2021

Figure 8-10: Accepting TGW Peering on London TGW – Accept TGW attachment (1).

After accepting the request, you will get a confirmation notification that you have to accept.

Are you sure that you want to accept this transit gateway peering attachment tgw-attach-08eaf28805581d397?

Cancel **Accept**

Figure 8-11: Accepting TGW Peering on London TGW – Accept TGW attachment (2).

Figure 8-12 shows how the state is changed from *Pending Acceptance* to *Pending*. Then, from *Pending* to *Available* (figure 8-13). Figure 8-13 also shows that the TGW

attachment is associated with the route table TGW-London-Default-RT (tgw-rtb-0648db5cfa12dc1a2) by default. We'll change the association later to dedicated RT.

The screenshot shows the AWS CloudFormation console with the search bar set to 'Search for services, features, marketplace products, and docs'. The user is signed in as 'Administrator @123456654321' in the London region. The main table lists three transit gateway attachments:

Name	Transit gateway attachment ID	Transit gateway ID	Resource type	Association route table
<input checked="" type="checkbox"/> Stockholm-London-Attach	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	Peering	-
<input type="checkbox"/> -	tgw-attach-01c6bcfa2cfae7da0	tgw-0cefc925035437d16	Peering	-
<input type="checkbox"/> Not-Used	tgw-attach-004c075de4de5f722	tgw-07b6ed53a6051e36d	VPC	-

Details for the 'Stockholm-London-Attach' row are shown in the modal:

- Transit gateway attachment ID:** tgw-attach-08eaf28805581d397
- State:** Pending
- Requester transit gateway ID:** tgw-0584fc5b3b0ee6a7
- Requester region:** Stockholm (eu-north-1)
- Acceptor transit gateway ID:** tgw-07b6ed53a6051e36d
- Acceptor region:** London (eu-west-2)
- Resource type:** Peering
- Requester owner ID:** 123456654321
- Acceptor owner ID:** 123456654321
- Association route table ID:** -
- Association state:** -

AWS VPC CH08- F12-5-11-2021

Figure 8-12: Accepting TGW Peering on London TGW – Pending.

The screenshot shows the AWS CloudFormation console with the search bar set to 'Search for services, features, marketplace products, and docs'. The user is signed in as 'Administrator @123456654321' in the London region. The main table lists three transit gateway attachments:

Name	Transit gateway attachment ID	Transit gateway ID	Resource type	Association route table
<input checked="" type="checkbox"/> Stockholm-London-Attach	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	Peering	tgw-rtb-0648db5cfa12dc1a2
<input type="checkbox"/> -	tgw-attach-01c6bcfa2cfae7da0	tgw-0cefc925035437d16	Peering	-
<input type="checkbox"/> Not-Used	tgw-attach-004c075de4de5f722	tgw-07b6ed53a6051e36d	VPC	-

Details for the 'Stockholm-London-Attach' row are shown in the modal:

- Transit gateway attachment ID:** tgw-attach-08eaf28805581d397
- State:** Available
- Requester transit gateway ID:** tgw-0584fc5b3b0ee6a7
- Requester region:** Stockholm (eu-north-1)
- Acceptor transit gateway ID:** tgw-07b6ed53a6051e36d
- Acceptor region:** London (eu-west-2)
- Resource type:** Peering
- Requester owner ID:** 123456654321
- Acceptor owner ID:** 123456654321
- Association route table ID:** tgw-rtb-0648db5cfa12dc1a2
- Association state:** Associated

AWS VPC CH08- F13-5-11-2021

Figure 8-13: Accepting TGW Peering on London TGW - Available.

Figure 8-14 verifies that the TGW peering state is *Available* in Stockholm-TGW. The tgw-attachment is associated with the default route table. That is ok in Stockholm because we are not implementing any TGW route table-based segmentation.

Name	Transit gateway attachment ID	Transit gateway ID	Resource type	Resource ID
Stockholm-London...	tgw-attach-08eaf28805581d397	tgw-0584fc5b3bc0ee6a7	Peering	tgw-07b6ed53a6051e36d

tgw-attach-08eaf28805581d397 / Stockholm-London-Attach

Details Tags

Details

Transit gateway attachment ID tgw-attach-08eaf28805581d397	State Available	Resource type Peering	Association route table ID tgw-rtb-0c99656302e7fdb3d
Requester transit gateway ID tgw-0584fc5b3bc0ee6a7	Requester region Stockholm (eu-north-1)	Requester owner ID 123456654321	Association state Associated
Acceptor transit gateway ID tgw-07b6ed53a6051e36d	Acceptor region London (eu-west-2)	Acceptor owner ID 123456654321	

Figure 8-14: Accepting TGW Peering on Stockholm TGW - Available.

RT of Stockholm-TGW

The next step we need to do after creating TGW Peering attachment is adding static routes to route tables. First, we add static routes towards subnets 10.10.0.0/16 and 172.16.10.0/24 to Stockholm-TGW-Default-RT. The next-hop for these destination subnets is TGW peering attachment (tgw-attach-08eaf28805581d397). Figure 8-15 shows the Stockholm-TGW-Default-RT and its associated resources, which is the local VPC and the TGW peering attachments.

Transit gateway route tables (1/1) Info						Actions	Create transit gateway route table	
<input checked="" type="checkbox"/>	Name	Transit gateway route table ID	Transit gateway ID	State	Default association			
<input checked="" type="checkbox"/>	Stockholm-TGW-Default-RT	tgw-rtb-0c99656302e7fdb3d	tgw-0584fc5b3bc0ee6a7	Available	Yes			
<hr/>								
tgw-rtb-0c99656302e7fdb3d / Stockholm-TGW-Default-RT								
Details	Associations	Propagations	Prefix list references	Routes	Tags			
<hr/>								
Associations (2) Info						Actions	Delete association	Create association
<input checked="" type="checkbox"/>	Attachment ID	Resource type	Resource ID	State				
<input checked="" type="checkbox"/>	tgw-attach-0f13483e7f7af9ff0	VPC	vpc-0495d34b52e0465d8	Associated				
<input checked="" type="checkbox"/>	tgw-attach-08eaf28805581d397	Peering	tgw-07b6ed53a6051e36d	Associated				AWS VPC CH08-F16-5-11-2021

Figure 8-15: Route Table Update – Stockholm-TGW Default RT.

Figure 8-16 shows that the only route is the automatically propagated VPC CIDR 10.100.0.0/16. Figure 8-17 shows the route table routing entries after we have added two static routes. As a recap, at the time of writing, TGW peering doesn't support route propagation.

tgw-rtb-0c99656302e7fdb3d / Stockholm-TGW-Default-RT						AWS.VPC.CH08-F16-5-11-2021	
Details	Associations	Propagations	Prefix list references	Routes	Tags		
<hr/>							
▶ Filter routes by CIDR (2)							
Routes (1/1)						Actions	Create static route
<input checked="" type="checkbox"/>	CIDR	Attachment ID	Resource ID	Resource type	Route type		
<input checked="" type="checkbox"/>	10.100.0.0/16	tgw-attach-0f13483e7f7af9ff0	vpc-0495d34b52e0465d8	VPC	Propagated		

Figure 8-16: Route Table Update – Stockholm-TGW Default RT (1).

tgw-rtb-0c99656302e7fdb3d / Stockholm-TGW-Default-RT						AWS VPC CH08- F17-5-11-2021																				
Details	Associations	Propagations	Prefix list references	Routes	Tags																					
Filter routes by CIDR (2)																										
Routes (3)																										
<input type="button" value="C"/> Actions ▾ <input type="button" value="Create static route"/>																										
<input type="text"/> Filter routes																										
<table border="1"> <thead> <tr> <th>CIDR</th> <th>Attachment ID</th> <th>Resource ID</th> <th>Resource type</th> <th>Route type</th> </tr> </thead> <tbody> <tr> <td>10.10.0.0/16</td> <td>tgw-attach-08eaf28805581d397</td> <td>tgw-07b6ed53a6051e36d</td> <td>Peering</td> <td>Static</td> </tr> <tr> <td>10.100.0.0/16</td> <td>tgw-attach-0f13483e7f7af9ff0</td> <td>vpc-0495d34b52e0465d8</td> <td>VPC</td> <td>Propagated</td> </tr> <tr> <td>172.16.10.0/24</td> <td>tgw-attach-08eaf28805581d397</td> <td>tgw-07b6ed53a6051e36d</td> <td>Peering</td> <td>Static</td> </tr> </tbody> </table>							CIDR	Attachment ID	Resource ID	Resource type	Route type	10.10.0.0/16	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	Peering	Static	10.100.0.0/16	tgw-attach-0f13483e7f7af9ff0	vpc-0495d34b52e0465d8	VPC	Propagated	172.16.10.0/24	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	Peering	Static
CIDR	Attachment ID	Resource ID	Resource type	Route type																						
10.10.0.0/16	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	Peering	Static																						
10.100.0.0/16	tgw-attach-0f13483e7f7af9ff0	vpc-0495d34b52e0465d8	VPC	Propagated																						
172.16.10.0/24	tgw-attach-08eaf28805581d397	tgw-07b6ed53a6051e36d	Peering	Static																						

Figure 8-17: Route Table Update – Stockholm-TGW Default RT (2).

RT of London-TGW

Now the Stockholm-TGW knows how to route packets to subnets 10.10.0.0/16 and 172.16.0.0/24. Next, we modify London-TGW route tables. In London-TGW, the TGW peering attachment is associated with *TGW-London-Default-RT* by default. We first delete this association (figure 8-18) and then associate it with the correct RT *Stockholm-London-RT* (figure 8-19).

tgw-rtb-0648db5cfa12dc1a2 / TGW-London-Default-RT						AWS VPC CH08- F18-5-11-2021								
Details	Associations	Propagations	Prefix list references	Routes	Tags									
Associations (1/1) Info														
<input type="button" value="C"/> Delete association <input type="button" value="Create association"/>														
<input type="text"/> Filter associations														
<table border="1"> <thead> <tr> <th>Attachment ID</th> <th>Resource type</th> <th>Resource ID</th> <th>State</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> tgw-attach-08eaf28805581d397</td> <td>Peering</td> <td>tgw-0584fc5b3bc0ee6a7</td> <td> Associated</td> </tr> </tbody> </table>							Attachment ID	Resource type	Resource ID	State	<input checked="" type="checkbox"/> tgw-attach-08eaf28805581d397	Peering	tgw-0584fc5b3bc0ee6a7	Associated
Attachment ID	Resource type	Resource ID	State											
<input checked="" type="checkbox"/> tgw-attach-08eaf28805581d397	Peering	tgw-0584fc5b3bc0ee6a7	Associated											

Figure 8-18: Route Table Update – Delete Association on London-TGW.

tgw-rtb-0e14e6a3057bc6d28 / Stockholm-London-RT						AWS VPC CH08- F19-5-11-2021								
Details	Associations	Propagations	Prefix list references	Routes	Tags									
Associations (1/1) Info														
<input type="button" value="C"/> Delete association <input type="button" value="Create association"/>														
<input type="text"/> Filter associations														
<table border="1"> <thead> <tr> <th>Attachment ID</th> <th>Resource type</th> <th>Resource ID</th> <th>State</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> tgw-attach-08eaf28805581d397</td> <td>Peering</td> <td>tgw-07b6ed53a6051e36d</td> <td> Associated</td> </tr> </tbody> </table>							Attachment ID	Resource type	Resource ID	State	<input checked="" type="checkbox"/> tgw-attach-08eaf28805581d397	Peering	tgw-07b6ed53a6051e36d	Associated
Attachment ID	Resource type	Resource ID	State											
<input checked="" type="checkbox"/> tgw-attach-08eaf28805581d397	Peering	tgw-07b6ed53a6051e36d	Associated											

Figure 8-19: Route Table Update – Create A New Association on London-TGW.

There are no routing entries in Stockholm-London-RT at this phase (figure 8-20).

The screenshot shows the AWS VPC Route Table Update interface for the route table tgw-rtb-0e14e6a3057bc6d28 / Stockholm-London-RT. The 'Routes' tab is selected. A search bar labeled 'Filter routes' is present. Below it is a table header with columns: CIDR, Attachment ID, Resource ID, Resource type, and Route type. A message 'No routes' and 'This route table does not have any routes.' is displayed. A 'Create static route' button is located at the bottom right of the table area.

Figure 8-20: Route Table Update – London-TGW (1).

London-TGW routes traffic flows received via TGW peering connection based on route table Stockholm-London-RT, we need to add static routes towards 10.10.0.0/16 and 172.16.10.0/24 to this RT too. The next-hop for destination 10.10.0.0/16 is the VPC attachment, while for the subnet 172.16.10.0/24, it is the VPN attachment. Note that I have also added a route to 10.100.0.0/16 into the Stockholm-London-RT, though it is not required. TGW uses this route table for routing the ingress traffic from the peering connection. Traffic to region local VPC CIDR 10.100.0.0/16 from Stockholm is never sent to London over TGW peering.

The screenshot shows the AWS VPC Route Table Update interface for the route table tgw-rtb-0e14e6a3057bc6d28 / Stockholm-London-RT. The 'Routes' tab is selected. A search bar labeled 'Filter routes' is present. Below it is a table showing three routes. The table has columns: CIDR, Attachment ID, Resource ID, Resource type, and Route type. The routes listed are:

CIDR	Attachment ID	Resource ID	Resource type	Route type
10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC	Static
10.100.0.0/16	tgw-attach-08eaf28805581d397	tgw-0584fc5b3bc0ee6a7	Peering	Static
172.16.10.0/24	tgw-attach-0c05d93f7b9933ffe	vpn-00ce8797da115243a...	VPN	Static

Figure 8-21: Route Table Update – London-TGW (2).

RT of TGW-London-VPC-RT

The VPC NWKT-Prod in London (eu-west-2) has its dedicated TGW route table TGW-London-VPC-RT (tgw-rtb-0a5f7104856a261f7). Figure 8-22 shows the route table after adding a static route to Stockholm CIDR 10.100.0.0/17 into it. The next-hop points to TGW Peering attachment tgw-attach-08eaf28805581d397.

Routes (4)						
	CIDR	Attachment ID	Resource ID	Resource type	Route type	Actions
<input type="checkbox"/>	0.0.0.0/0	2 Attachments	2 Resources	VPN	Propagated	
<input type="checkbox"/>	10.100.0.0/16	tgw-attach-08eaf28805581d397	tgw-0584fc5b3bc0ee6a7	Peering	Static	
<input type="checkbox"/>	10.12.0.0/16	tgw-attach-034c1731553b77b41	vpc-0a2c6c297c4562f11	VPC	Propagated	
<input type="checkbox"/>	172.16.10.0/24	2 Attachments	2 Resources	VPN	Propagated	

Figure 8-22: TGW-London-VPC-RT.

RT of TGW-London-VPN-RT

Figure 8-23 shows the TGW-London-VPN-RT after adding the static route.

Routes (3)						
	CIDR	Attachment ID	Resource ID	Resource type	Route type	Actions
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-06acc2b5d65998c07	vpc-04ef72cc79a73f82e	VPC	Propagated	
<input type="checkbox"/>	10.100.0.0/16	tgw-attach-08eaf28805581d397	tgw-0584fc5b3bc0ee6a7	Peering	Static	
<input type="checkbox"/>	10.11.0.0/16	tgw-attach-0eb49a52d3036d736	vpc-0867c98f35d11fcdd	VPC	Propagated	

Figure 8-23: TGW-London-VPN-RT.

We are using dynamic routing (eBGP) between London-TGW and on-prem Datacenter CGW devices. VPN connection is attached to TGW-London-VPN-RT, and TGW will include all routing entries to BGP Updates by default. Example 8-1 shows that the CGW device NWKT-WAN-Edge-01 has received BGP updates from London-TGW over both VPN tunnels.

```
NWKT-WAN-Edge-01#sh ip bgp
BGP table version is 26, local router ID is 192.168.100.18
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
      Network          Next Hop            Metric LocPrf Weight Path
      0.0.0.0          0.0.0.0             0          0 i
*   10.10.0.0/16    169.254.175.105    100        0 64512 i
*>  10.11.0.0/16    169.254.241.209    100        0 64512 i
*>  10.11.0.0/16    169.254.175.105    100        0 64512 i
*>  10.100.0.0/16   169.254.241.209    100        0 64512 i
*>  10.100.0.0/16   169.254.175.105    100        0 64512 i
*>  172.16.10.0/24  0.0.0.0              0          32768 i
NWKT-WAN-Edge-01#
```

Example 8-1: The BGP Table of the CGW Device NWKT-WAN-Edge-01.

RT of Stockholm-EC2-RT

After updating London-TGW and Stockholm-TG route tables, we still need to update subnet-specific route tables. We start by adding static routes to Stockholm-EC2-RT. This route table is associated with the subnet 10.100.0.0/24, where we run our EC2 instance (10.100.0.218). The next-hop for both destination 10.10.0.0/16 and 172.16.10.0/24 is Stockholm-TGW (tgw-0584fc5b3bc0ee6a7).

rtb-047ef11ba2dba7924 / Stockholm-EC2-RT						
Details	Routes	Subnet associations	Edge associations	Route propagation	Tags	
Routes (3)						
<input type="button" value="Edit routes"/> <div style="display: flex; justify-content: space-between;"> <input type="text"/> Filter routes Both < 1 > </div>						
Destination	Target	Status	Propagated			
172.16.10.0/24	tgw-0584fc5b3bc0ee6a7		No			
10.10.0.0/24	tgw-0584fc5b3bc0ee6a7		No			
10.100.0.0/16	local		No			

Figure 8-24: Route Table of the Subnet 10.100.0.0/24 (Stockholm).

RT of NWKT-Prod-Public

As the last configuration change, we add a static route to the subnet 10.10.0.0/24 (NWKT-Prod) route table. The figure shows that the next-hop for the destination 10.100.0.0/16 is London-TGW. Figure 8-25 shows the Route table sheet in the subnet NWKT-prod management view, while the previous figure 8-24 shows the Routes tab in the Stockholm-EC2-RT route table. You can verify information from more than one source.

Route table: rtb-05dc9cd679937ceee / NWKT-Prod-Public		Edit route table association
Routes (5)		
<input type="text"/> Filter routes		
Destination	Target	
172.16.10.0/24	tgw-07b6ed53a6051e36d	
10.12.0.0/24	tgw-07b6ed53a6051e36d	
10.10.0.0/16	local	
10.100.0.0/16	tgw-07b6ed53a6051e36d	
0.0.0.0/0	igw-0e997955bc6597031	

Figure 8-25: Route Table of the Subnet 10.10.0.0/24 (London).

Verify IP Connection

After routing table updates, we have an IP connectivity from on-prem Datacenter subnet 172.16.10.0/24 to EC2 10.100.0.218 running on Stockholm.

```
NWKT-WAN-Edge-01#
NWKT-WAN-Edge-01#ping 10.100.0.218 source 172.16.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.0.218, timeout is 2 seconds:
Packet sent with a source address of 172.16.10.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 67/71/84 ms
NWKT-WAN-Edge-01#
```

Example 8-2: Ping from on-prem Datacenter to Stockholm EC2 (10.100.0.218).

Example 8-3 shows that we can also ping from EC2 instance 10.10.0.218 running on subnet NWKT-Prod on VPC in AWS London region to EC2 10.100.0.218 we have launched on Stockholm subnet.

```
[ec2-user@ip-10-10-0-218 ~]$ ping 10.100.0.218
PING 10.100.0.218 (10.100.0.218) 56(84) bytes of data.
64 bytes from 10.100.0.218: icmp_seq=1 ttl=252 time=34.9 ms
64 bytes from 10.100.0.218: icmp_seq=2 ttl=252 time=33.8 ms
64 bytes from 10.100.0.218: icmp_seq=3 ttl=252 time=33.8 ms
64 bytes from 10.100.0.218: icmp_seq=4 ttl=252 time=33.7 ms
64 bytes from 10.100.0.218: icmp_seq=5 ttl=252 time=33.9 ms
^C
--- 10.100.0.218 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 33.729/34.051/34.944/0.450 ms
[ec2-user@ip-10-10-0-218 ~]$
```

Example 8-3: Ping from subnet 10.10.0.0/24 to Stockholm EC2 (10.100.0.218).

TGW Peering Pricing

When EC2 in London VPC sends data to EC2 in Stockholm, you will be charged 0,02 \$/1GB for London-TGW ingress traffic flows. You will also be charged 0,02\$/1GB that London-TGW sends over TGW Peering Connection to Stockholm-TGW. However, there is no fee for TGW ingress data flow received from TGW peers or sent to VPC. That means that you will pay 0,04\$ for 1 GB of data sent from London to Stockholm.

Summary

Figure 8-26 shows the building block we have in this chapter. We started by creating VPC Peering Connection between TGWs running in the Stockholm region and London region. Then we update all necessary TGW route tables. Next, we updated subnet-specific route tables. Remember that we also have to update Security Groups associated with our EC2 instance needs to allow traffic flows. In addition, make sure that the Network ACL (NACL) protecting the subnet has to permit our application traffic.

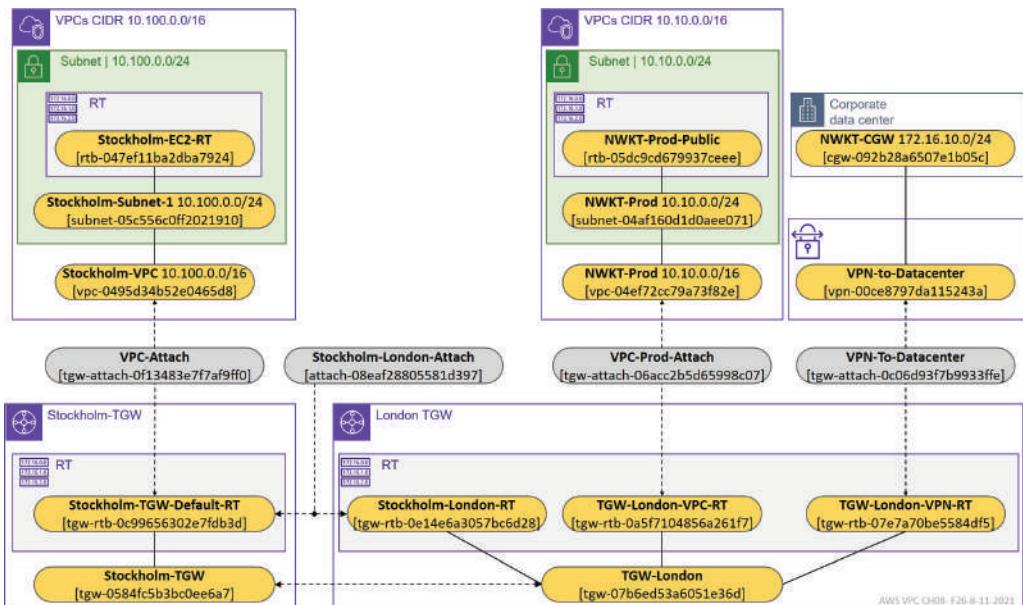


Figure 8-26: Transit Gateway Peering Summary Diagram.

Chapter 9: VPC Peering

Introduction

The VPC Peering is a simple solution that creates an IP connection between the two VPC's CIDR ranges. VPC Peering requires unique VPC CIDR ranges and doesn't support overlapping CIDR ranges. In addition, VPC peering is always a request/accept contract between the two VPCs. You can't use peering VPC as a transit gateway to another VPC. AWS VPC routing policy verifies that either the source IP address or destination IP address is attached to one of the interfaces within VPC, other vice packets are dropped. For example, EC2 instance 10.10.0.218 on VPC NWKT-Prod can't reach EC2 instance 10.11.0.116 on VPC NWKT-Test over VPC Peering connection with NWKT-Dev. That is because neither source nor destination IP address are attached to any local interface in VPC-Dev. When your application traffic connection matrix requires more than three inter-VPC connections, you may consider using Transit Gateway (TGW). The benefit of using VPC peering, however, is that data flows over VPC peering connection within Availability Zone is free. With the TGW, you will be charged 0,05 \$ per VPC connection per hour. For example, if you connect two VPCs via TGW, you will pay 0,1 \$ per hour. You will also pay 0,02 \$ per 1 GB when TGW receives data flow from one of your EC2 instances.

We are going to create three VPC peering in this example. The first one is from VPC NWKT-Prod to VPC NWK-Dev. The second one is between NWKT-Prod and VPC NWK-Test. The last VPC Peering is from VPC NWKT-Dev to VPC NWKT-Test. By doing this, we allow all traffic flows between all three VPCs. However, our security policy defines that in addition intra-VPC communication, data flows from the private subnet 10.10.1.0/24 on VPC NWKT-Prod is allowed to destination instances in subnet 10.12.0.0/24 on VPC NWKT-Test. We implement this policy by adding only allowed destinations into subnet-specific route tables.

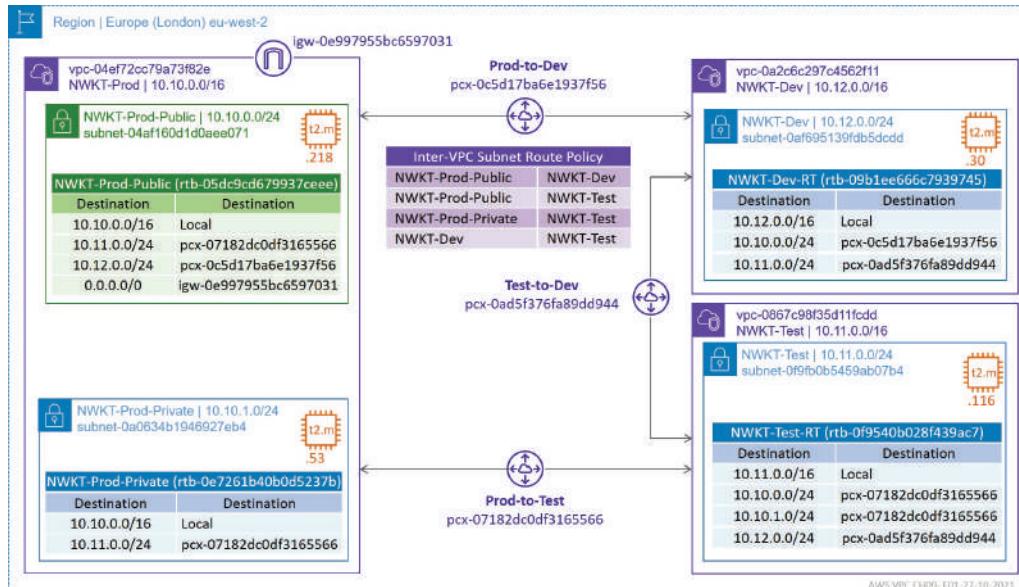


Figure 9-1: VPC Peering Overall Design.

Configure VPC Peering

Figure 9-2 illustrates our VPC Peering Connection. VPC Peering Prod-to-Dev (pcx-0c5d17ba6e1937f56) connects VPCs NWKT-Prod and NWKT-Dev. VPC Peering Prod-to-Test (pcx-07182dc0df3165566), in turn, connects VPCs NWKT-Prod to NWKT-test. VPC Peering Test-to-Dev (pcx-0ad5f376fa89dd944). This section shows how to create the first VPC Peering Prod-to-Dev. The other two VPCs are created using the same configuration processes, and that's why I excluded their configuration steps.

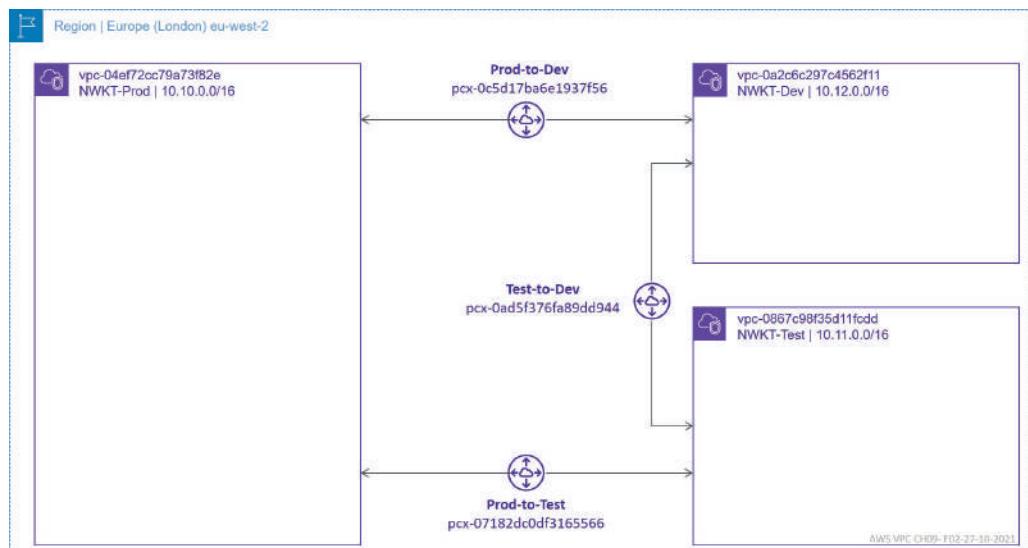


Figure 9-2: VPC Peering.

Navigate to the VPC Dashboard. Select either the *Peering Connection* hyperlink under the *Virtual Private Cloud* section or the VPC Peering Connection hyperlink from the Resources by Region window.

AWS VPC CH09- F03-27-10-2021

New VPC Experience
Tell us what you think

VIRTUAL PRIVATE CLOUD

- Your VPCs
- Subnets
- Route Tables **New**
- Internet Gateways
- Egress Only Internet Gateways
- Carrier Gateways
- DHCP Options Sets
- Elastic IPs
- Managed Prefix Lists
- Endpoints
- Endpoint Services
- NAT Gateways
- Peering Connections **New****

SECURITY

Launch VPC Wizard **Launch EC2 Instances**

Note: Your Instances will launch in the Europe region.

Resources by Region

Refresh Resources

You are using the following Amazon VPC resources

VPCs	Europe 4	NAT Gateways	Europe 0
Subnets	Europe 7	VPC Peering Connections	Europe 0
Route Tables	Europe 9	Network ACLs	Europe 4
Internet Gateways	Europe 2	Security Groups	Europe 6
Egress-only Internet Gateways	Europe 0	Customer Gateways	Europe 0

Figure 9-3: Configuring VPC Peering: Step-1.

Click the *Create peering connection* button in the Peering connection window.

Peering connections **Info**

Create peering connection

Filter peering connections

Name	Peering connection ID	Status	Requester VPC

AWS VPC CH09- F04-27-10-2021

Figure 9-4: Configuring VPC Peering: Step-2.

Give the name to connection (optional). Select the requester VPC NWKT-Prod from the *Select Local to Peer with* drop-down menu. After selecting it, the CIDR range 10.10.0.0/16 associated with VPC NWKT-Prod will appear on the screen.

Because the Acceptor VPC NWKT-Dev is under our account, we check the *My Account* radio button. All our VPC are in AWS London Region (eu-west-2), so the Region option is *This Region (eu-west-2)*. Select the VPC NWKT-Dev from the VPC ID (Acceptor) drop-down menu. You will also see the associated CIDR 10.12.0.0/16 after selecting the VPC.

Create peering connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them privately. [Info](#)

Peering connection settings

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

Prod-to-Dev

Select a local VPC to peer with

VPC ID (Requester)

vhc-04ef72cc79a73f82e (NWKT-Prod)

VPC CIDRs for vhc-04ef72cc79a73f82e (NWKT-Prod)

CIDR	Status	Status reason
10.10.0.0/16	Associated	-

Select another VPC to peer with

Account

My account
 Another account

Region

This Region (eu-west-2)
 Another Region

VPC ID (Acceptor)

vhc-0a2c6c297c4562f11 (NWKT-Dev)

VPC CIDRs for vhc-0a2c6c297c4562f11 (NWKT-Dev)

CIDR	Status	Status reason
10.12.0.0/16	Associated	- AWS VPC CH09- F05-27-10-2021

Figure 9-5: Figure 9-4: Configuring VPC Peering: Step-3.1.

Figure 9-6 is part of the same view as figure 9-5. The Tags field shows the name tag that we use in the name field. You can add a new tag if needed. Next, click Create peering connection button.

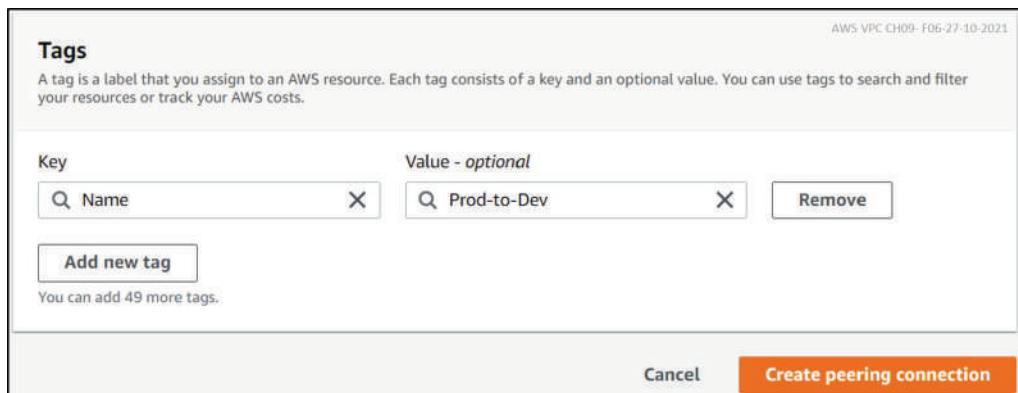


Figure 9-6: Configuring VPC Peering: Step-3.2.

Figure 9-7 shows the notification window. It describes both the Requester's and the Acceptor's Account IDs, VPC names, and AWS Regions. Note that only the Requester CIDR range is shown in the window at this phase. Accept the request by selecting Accept Request from the Action menu.

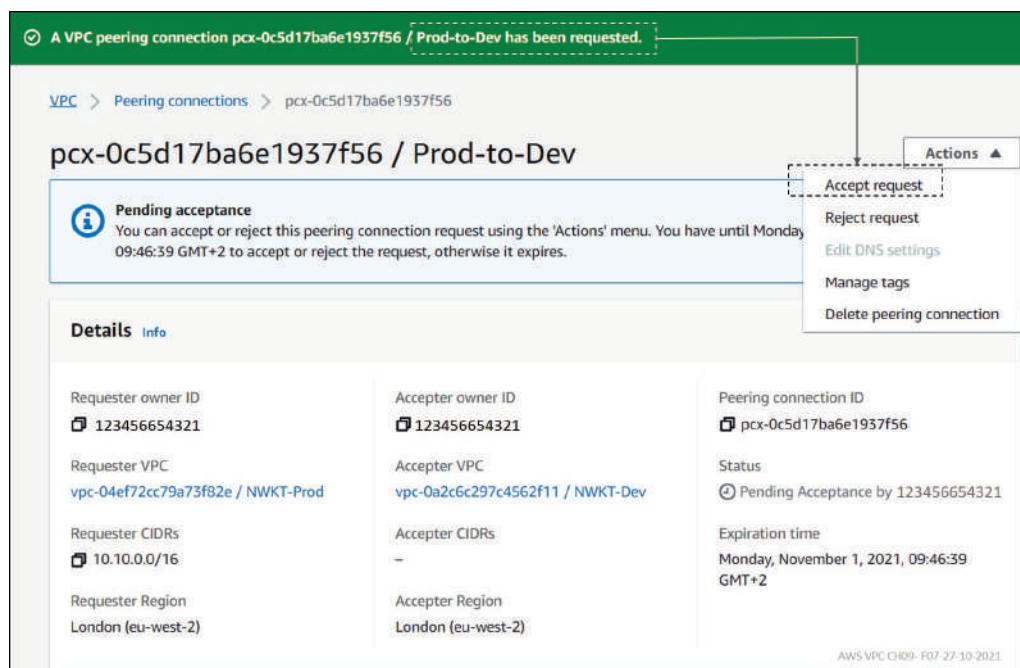


Figure 9-7: Configuring VPC Peering: Step-4.

Click the *Accept request* button to proceed.

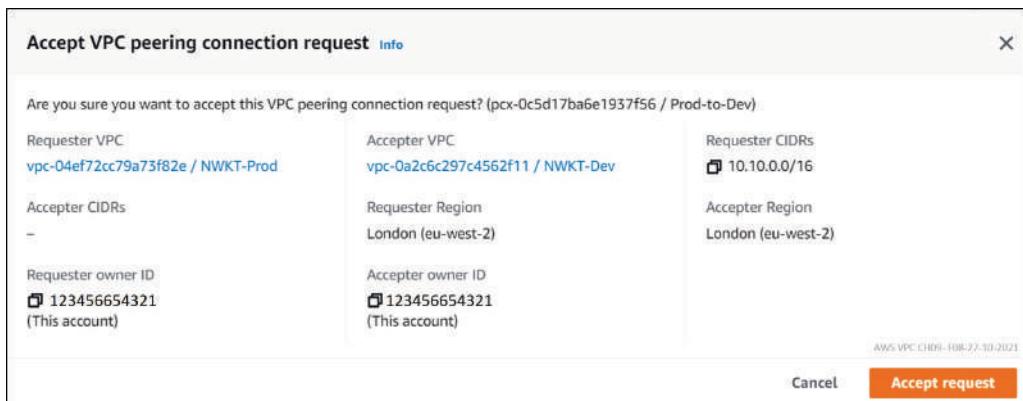


Figure 9-8: Configuring VPC Peering: Step-5.

Figure 9-9 shows the notification about the successful VPC Peering Connection process. VPC Peering itself only connects two VPCs. You still need to route update subnet-specific route tables before packets are forwarded over peering connection.

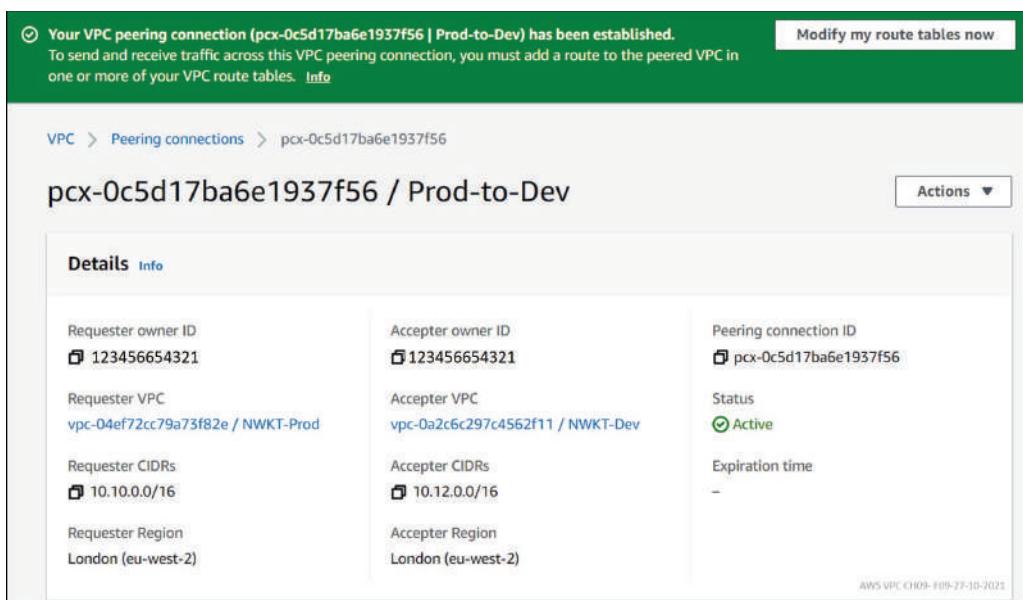


Figure 9-9: Configuring VPC Peering: Step-6.

Figure 9-10 shows all three VPC Peering Connections. You can see that the VPC NWKT-Prod is Requester VPC for peering with NWKT-Dev as well with NWKT-Test. VPC NWKT-Test is the Requester for peering with the VPC NWKT-Dev.

The screenshot shows a table titled "Peering connections (1/3) Info". The columns are: Name, Peering connection ID, Status, Requester VPC, and Acceptor VPC. There are three rows:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC
Test-to-Dev	pcx-0ad5f376fa89dd944	Active	vpc-0867c98f35d11fcdd / NWKT-Test	vpc-0a2c6c297c4
Prod-to-Dev	pcx-0c5d17ba6e1937f56	Active	vpc-04ef72cc79a73f82e / NWKT-Prod	vpc-0a2c6c297c4
Prod-to-Test	pcx-07182dc0df3165566	Active	vpc-04ef72cc79a73f82e / NWKT-Prod	vpc-0867c98f35d

AWS VPC CI401-F10 27.10.2021

Figure 9-10: Configuring VPC Peering: Step-5.

Update Route Tables

Figure 9-11 illustrates route tables of each subnet. The route table of the subnet 10.10.0.0/24 (NWKT-Prod-Public) has two routing entries related to VPC Peering. The next-hop for the subnet 10.11.0.0/24 is the VPC Peering Prod-to-Test (pcx-07182dc0df3165566), and the next-hop for the subnet 10.12.0.0/24 is the VPC Peering connection Prod-to-Dev (pcx-0c5d17ba6e1937f56). We naturally need a route from the subnets 10.11.0.0/24 (NWKT-Test) and 10.12.0.0/24 (NWKT-Dev) back to subnet NWKT-Prod-Public. The routing logic is the same as we used in the route table of subnet 10.10.0.0/24. The next-hop is the VPC Peering Connection. The route table of the subnet NWKT-Prod-Private has a route to the subnet 10.11.0.0/24 (NWKT-Test) with the next of pointing to VPC Peering Prod-to-Test (pcx-07182dc0df3165566). We also add a return route to RT of the subnet NWKT-test. As the last step, we make necessary routing changes to allow traffic flows between NWKT-Test and NWKT-Dev over VPC Peering Test-to-Dev (pcx-0ad5f376fa89dd944).

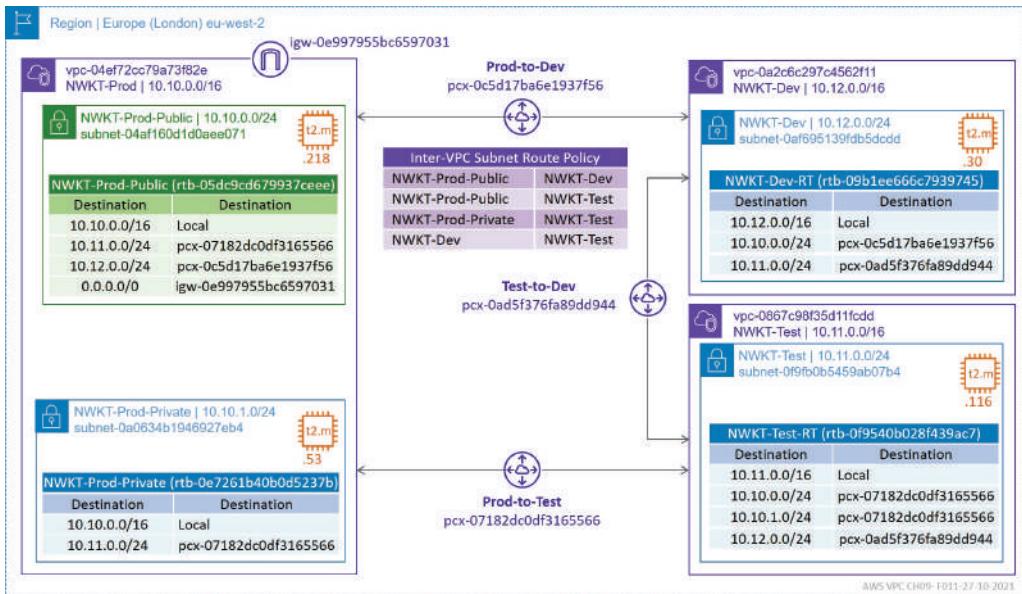


Figure 9-11: Routing Logic.

Figure 9-12 shows our route tables. We edit the route table of NWKT-Prod-Private first. Click the rtb-0e7261b40b0d5237b hyperlink in Route table id column.

Route tables (9) Info				
<input type="checkbox"/>	Name	Route table ID	Actions	Create route table
Filter route tables AWS VPC CH09-F12-27-10-2021				
<input type="checkbox"/>	NWKT-Test-Private	rtb-0f9540b028f439ac7	Edit	Delete
<input type="checkbox"/>	NWKT-PUB-RT	rtb-0fd4639034844c3ea	Edit	Delete
<input type="checkbox"/>	NWKT-Prod-Public	rtb-05dc9cd679937ceee	Edit	Delete
<input type="checkbox"/>	NWKT-Prod-Private	rtb-0e7261b40b0d5237b	Edit	Delete
<input type="checkbox"/>	NWKT-MAIN-RT	rtb-069ac98ac692271fe	Edit	Delete
<input type="checkbox"/>	NWKT-Dev-Private	rtb-09b1ee666c7939745	Edit	Delete

Figure 9-12: Route Tables.

Click the Edit routes button on a Routes tab.

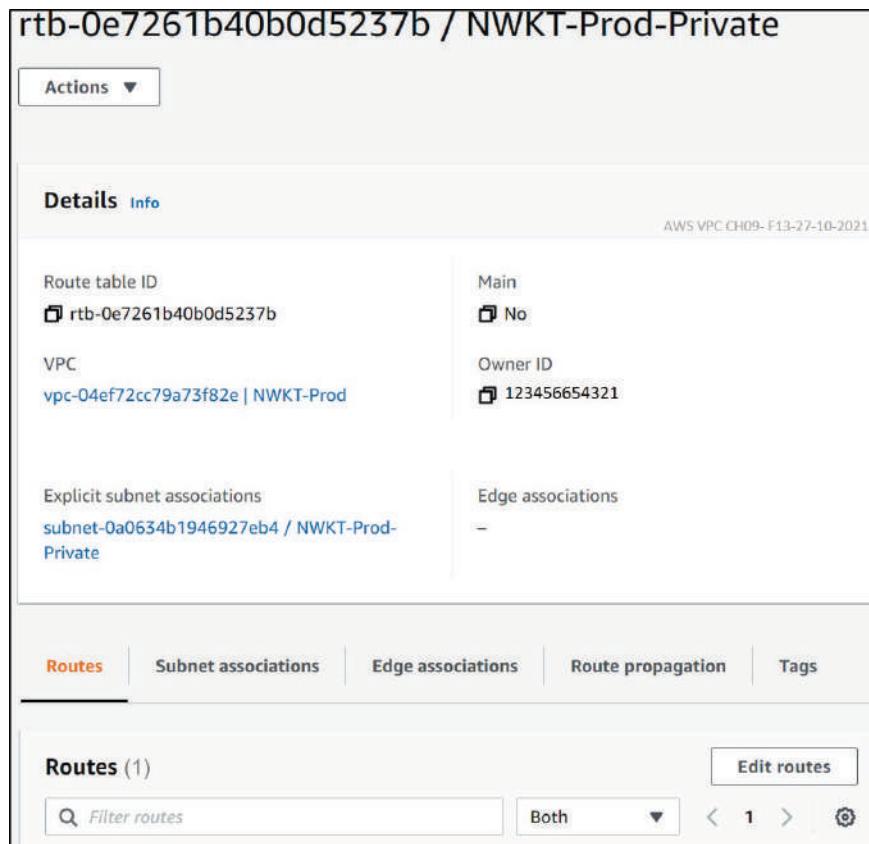


Figure 9-13: Edit Route Table of NWKT-Prod-Private: Step-1.

Add subnet 10.11.0.0/24 to the destination field. Then select Peering Connection from the target drop-down menu and choose the VPC Peering Connection pcx-07182dc0df3165566 | Prod-to-Test. Figure 9-15 shows the partial Details view where our selected VPC Peering is the target (= next-hop). Click the Save changes button (not visible in the screen capture).

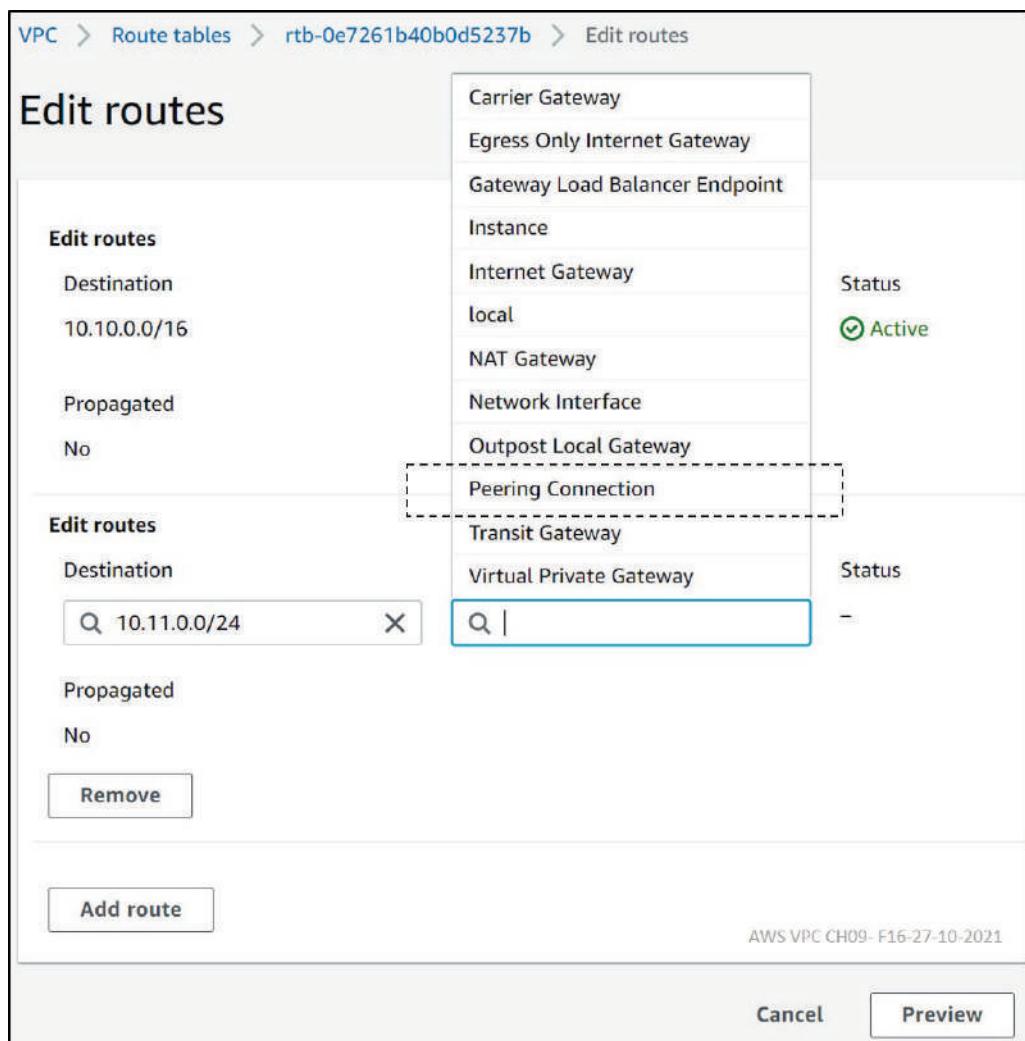


Figure 9-14: Edit Route Table of NWKT-Prod-Private: Step-2.1.



Figure 9-15: Edit Route Table of NWKT-Prod-Private: Step-2.2.

Figure 9-16 shows routing entries on route table NWKT-Prod-Private (the rtb-0e7261b40b0d5237b) after we have added the route.

Routes	Subnet associations	Edge associations	Route propagation	Tags
Routes (2)				
<input type="text" value="Filter routes"/> Both < 1 > ⚙️				
Destination	Target	Status	Propagated	
10.11.0.0/24	pcx-07182dc0df3165566	Active	No	AWS VPC CH09 - F16-27-10-2021
10.10.0.0/16	local	Active	No	

Figure 9-16: Edit Route Table of NWKT-Prod-Private: Step-3.

Figure 9-17 illustrates the route table NWKT-Prod-Public (rtb-05dc9cd679937ceee) after route table edition. This one is associated with the public subnet, and that is why we have the default route pointing to Internet gateway.

Routes	Subnet associations	Edge associations	Route propagation	Tags
AWS VPC CH09 - F17-27-10-2021				
Routes (4)				
<input type="text" value="Filter routes"/> Both < 1 > ⚙️				
Destination	Target	Status	Propagated	
10.11.0.0/24	pcx-07182dc0df3165566	Active	No	
10.12.0.0/24	pcx-0c5d17ba6e1937f56	Active	No	
10.10.0.0/16	local	Active	No	
0.0.0.0/0	igw-0e997955bc6597031	Active	No	

Figure 9-17: Route Table NWKT-Prod-Public.

The same routing logic is used with route tables of NWKT-Test-RT and NWKT-Dev-RT. The target/next-hop for all inter-VPC destinations is VPC Peering we are using between VPCs. Last but not least, remember to update Security Groups and Network ACLs.

Test Connectivity

Examples from 9-1 to 9-6 verifies that our VPC Peering and route policy works as expected.

```
[ec2-user@ip-10-10-0-218 ~]$ ping 10.11.0.116
PING 10.11.0.116 (10.11.0.116) 56(84) bytes of data.
64 bytes from 10.11.0.116: icmp_seq=1 ttl=255 time=0.434 ms
64 bytes from 10.11.0.116: icmp_seq=2 ttl=255 time=0.462 ms
64 bytes from 10.11.0.116: icmp_seq=3 ttl=255 time=0.439 ms
64 bytes from 10.11.0.116: icmp_seq=4 ttl=255 time=0.395 ms
64 bytes from 10.11.0.116: icmp_seq=5 ttl=255 time=0.436 ms
^C
--- 10.11.0.116 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.395/0.433/0.462/0.025 ms
```

Example 9-1: Ping from 10.10.0.218 to 10.11.0.16 (on VPC NWKT-Test).

```
[ec2-user@ip-10-10-0-218 ~]$ ping 10.12.0.30
PING 10.12.0.30 (10.12.0.30) 56(84) bytes of data.
64 bytes from 10.12.0.30: icmp_seq=1 ttl=255 time=0.836 ms
64 bytes from 10.12.0.30: icmp_seq=2 ttl=255 time=0.884 ms
64 bytes from 10.12.0.30: icmp_seq=3 ttl=255 time=0.839 ms
64 bytes from 10.12.0.30: icmp_seq=4 ttl=255 time=0.935 ms
64 bytes from 10.12.0.30: icmp_seq=5 ttl=255 time=0.887 ms
^C
--- 10.12.0.30 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.836/0.876/0.935/0.040 ms
```

Example 9-2: Ping from 10.10.0.218 to 10.12.0.30 (on VPC NWKT-Dev).

```
[ec2-user@ip-10-10-0-218 ~]$ ping 10.10.1.53
PING 10.10.1.53 (10.10.1.53) 56(84) bytes of data.
64 bytes from 10.10.1.53: icmp_seq=1 ttl=255 time=0.655 ms
64 bytes from 10.10.1.53: icmp_seq=2 ttl=255 time=0.449 ms
64 bytes from 10.10.1.53: icmp_seq=3 ttl=255 time=0.436 ms
64 bytes from 10.10.1.53: icmp_seq=4 ttl=255 time=0.387 ms
64 bytes from 10.10.1.53: icmp_seq=5 ttl=255 time=0.437 ms
^C
--- 10.10.1.53 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4074ms
rtt min/avg/max/mdev = 0.387/0.472/0.655/0.097 ms
```

Example 9-3: Ping from 10.10.0.218 to 10.10.1.53 (intra-VPC).

```
[ec2-user@ip-10-11-0-116 ~]$ ping 10.10.1.53
PING 10.10.1.53 (10.10.1.53) 56(84) bytes of data.
64 bytes from 10.10.1.53: icmp_seq=1 ttl=255 time=0.353 ms
64 bytes from 10.10.1.53: icmp_seq=2 ttl=255 time=0.391 ms
64 bytes from 10.10.1.53: icmp_seq=3 ttl=255 time=0.334 ms
64 bytes from 10.10.1.53: icmp_seq=4 ttl=255 time=0.410 ms
64 bytes from 10.10.1.53: icmp_seq=5 ttl=255 time=0.382 ms
^C
--- 10.10.1.53 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4084ms
rtt min/avg/max/mdev = 0.334/0.374/0.410/0.027 ms
```

Example 9-4: Ping from 10.11.0.116 to 10.10.1.53.

```
[ec2-user@ip-10-11-0-116 ~]$ ping 10.12.0.30
PING 10.12.0.30 (10.12.0.30) 56(84) bytes of data.
64 bytes from 10.12.0.30: icmp_seq=1 ttl=255 time=0.788 ms
64 bytes from 10.12.0.30: icmp_seq=2 ttl=255 time=0.832 ms
64 bytes from 10.12.0.30: icmp_seq=3 ttl=255 time=0.831 ms
64 bytes from 10.12.0.30: icmp_seq=4 ttl=255 time=0.847 ms
64 bytes from 10.12.0.30: icmp_seq=5 ttl=255 time=0.834 ms
^C
--- 10.12.0.30 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4104ms
rtt min/avg/max/mdev = 0.788/0.826/0.847/0.032 ms
```

Example 9-5: Ping from 10.11.0.116 to 10.12.0.30.

```
[ec2-user@ip-10-12-0-30 ~]$ ping 10.10.1.53
PING 10.10.1.53 (10.10.1.53) 56(84) bytes of data.
^C
--- 10.10.1.53 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5099ms
```

Example 9-6: Ping from 10.12.0.30 to 10.10.1.53.

Chapter 10: AWS PrivateLink

Introduction

So far, this book has explained how we can implement an Inter-VPC connection by using Transit Gateway or VPC Peering solutions. This chapter introduces the third option, AWS PrivateLink. The first section discusses how we launch a Network Load Balancer (NLB) in a VPC NWKT-Test. NWKT-Test is our *Service Provider* VPC that provides services to *Service Consumers* in its EC2 instance NWKT-EC-01. The Service Consumer can be any VPC under any account (PrivateLink allows cross AWS account VPC connections). During the NLB creation process, we define a *Listener* process, which states the protocol and port, which NLB waits for connection attempts from the consumer side. We are using TCP port 80. Then we create a *Target Group* and associate a set of instances to it. We also specify a protocol and port (TCP: 80) which NLB uses when it opens the connection to one of its TG instances. As a last NLB step, we set up service availability *Health Check* options. We use the same port/protocol (TCP: 80) combination for Health Check than what we are using for actual consumer traffic flows. After creating the NLB, we launch a *VPC Endpoint Service*, which we associate with our NLB. Next, we create an *Interface Endpoint* which, in turn, uses the VPC Endpoint Service for connecting to services behind NLB. Interface Endpoint is seen as Elastic Network Interface (ENI) in your Service Consumer network. In our example, the ENI gets an IP address from the IP pool of the subnet NWKT-Prod-Public (10.10.0.0/24). When we want to connect to service from our Service Consumer VPC, we call the IP address of ENI of Interface VPC Endpoint. It then forwards traffic to NLB, which performs a source NAT before forwarding traffic to the target. In our example, provider registers the service using the name vpce-svc-05ad80b00dd1783f65. You may want to use an Alias record for giving a friendly name for the service, such as vpce.nwktweb.com. When we use DNS names to access the service, our DNS service resolves the name to the IP address assigned to our ENI. As said, it is an IP address that is in the Service Consumer subnet. From our consumer EC2 instance point of view, the service is in the same subnet. It means that we don't have to modify the subnet-specific route table. The same applies to subnet where we are running our services, NLB does the source NAT for ingress traffic using the IP from the service instance subnet. From the service instance NWKT-EC2-1point of view the connection request comes from subnet 10.11.0.0/24.

AWS PrivateLink based Inter-VPC solution differs from VPC Peering (VPC-P) and Transit Gateway (TGW) implementations. Both VPC-P and TGW open bidirectional IP connections between VPCs for the whole VPC IP ranges, which means that these services don't support overlapping IP addresses. AWS PrivateLink, in turn, is an application-specific, host-based connection from the Service Consumer to Service Provider. That said, EC2 instances in Service Provider VPC can't initiate connection to hosts running on a Service Consumer VPC over PrivateLink connection. AWS PrivateLink is a powerful micro-segmentation tool. AWS PrivateLink solution creates Network Load Balancer -based (NLB) VPC Endpoint Service, where NLB uses source NAT for ingress traffic. That makes it possible to use an overlapping IP address in Service Consumer VPC and Service Provider VPC. We can also publish VPC Endpoint Service for several VPC and allow Service Providers to offer shared services. Due to NLB source NAT, also Service Consumers IP ranges can overlap.

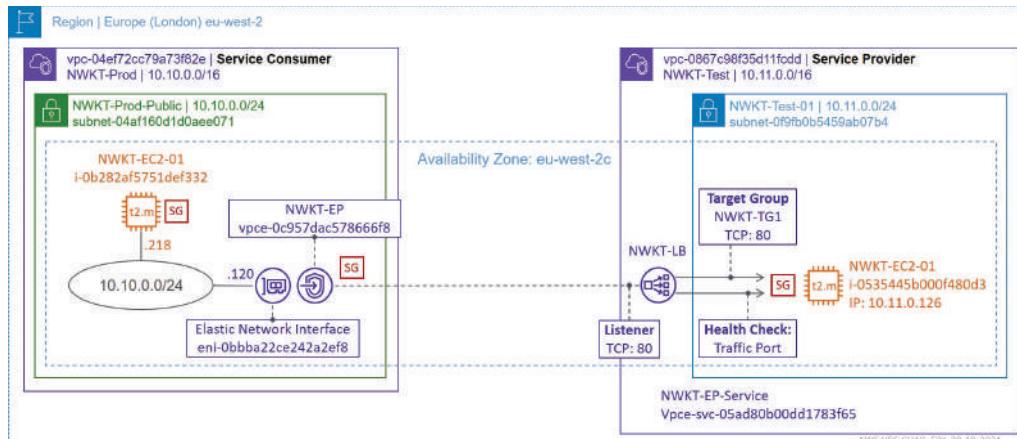


Figure 10-1: AWS Private Link Design.

Create Network Load Balancer

AWS PrivateLink supports only Network Load Balancer (NLB). NLB operates at OSI Layer 4 (Transport Layer). It listens to connection requests for port/protocol we have defined in its Front-End listener process. In our example, we use TCP/80. When NLB receives a connection request, it opens a TCP connection to port 80 with one of its healthy targets included in the Target Group. In our example, we are using the traffic port for monitoring targets health checks.

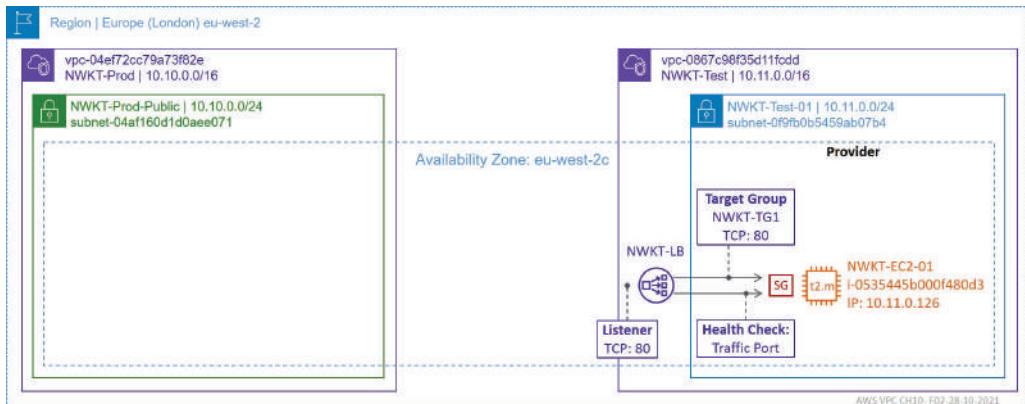


Figure 10-2: Service Provider's Network Load Balancer.

Start the NLB configuration process by navigating to VPC Dashboard and selecting the *Load Balancer* under the Load balancing header. Then, click the *Create Load Balancer* button.

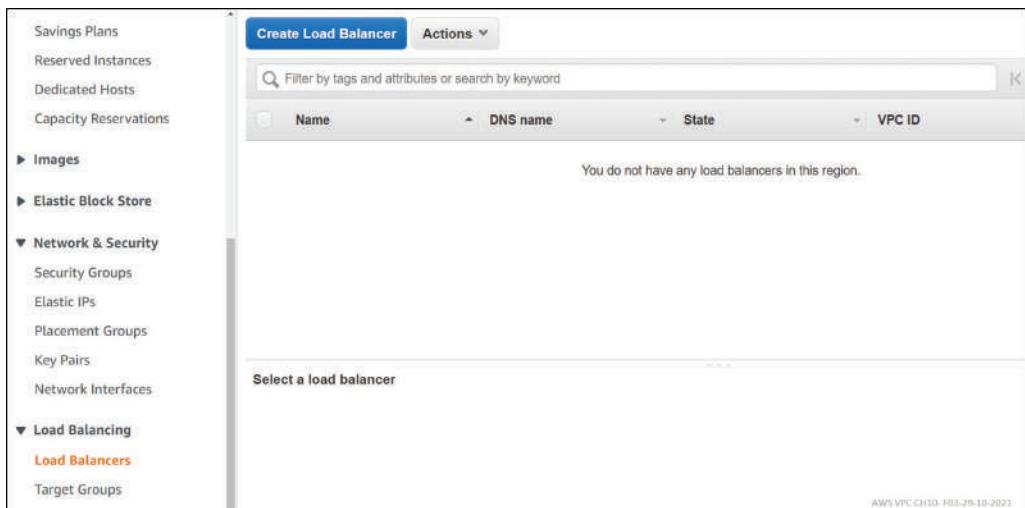


Figure 10-3: Create Network Load Balancer: Step-1, Create NLB.

Next, select the Network Load Balancer option.

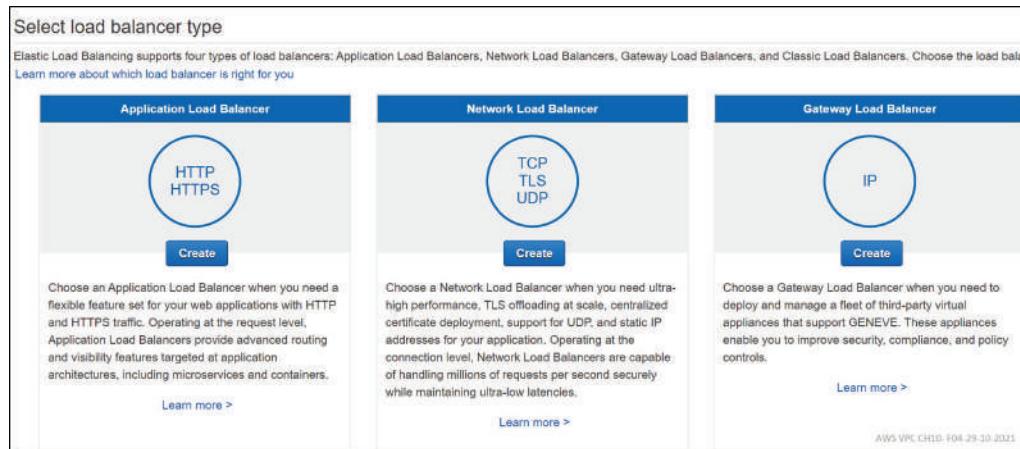


Figure 10-4: Create Network Load Balancer: Step-2, Select LB type.

First, we define the Listener. Select TCP port 80 for Load Balancer Protocol and port for Listener service. By doing that, our NLB listens to connection initiations from the Service Consumer side to TCP port 80. Then we select the NLB location. We will launch it on Availability Zone eu-west-2c on the Service Provider VPC NWKT-Test. Our target EC2 instance is in subnet 10.11.0.0/24 (NWKT-Test), so select it from the drop-down menu. We are not configuring any LB Security for simplicity so we can bypass the second phase.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Routing 4. Register Targets 5. Review

Step 1: Configure Load Balancer

IP address type ⓘ IPv4

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
TCP	80

Add listener

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You may also add one Elastic IP per Availability Zone if you wish to have specific addresses for your load balancer.

VPC ⓘ	vpc-0867c98f35d11fcdd (10.11.0.0/16) NWKT-Test
Availability Zones	<input checked="" type="checkbox"/> eu-west-2c subnet-0f9fb0b5459ab07b4 (NWKT-Test)
IPv4 address ⓘ	Assigned from CIDR 10.11.0.0/24
Private IPv4 address ⓘ	Assigned from CIDR 10.11.0.0/24

① **Temporary limitation**
Choose your Availability Zones and subnets with care. After you create the load balancer, you cannot disable the enabled subnets, but you can enable additional ones.

▼ Tags AWS VPC CH10- F05-29-10-2021

Apply tags to your load balancer to help organize and identify them.

Cancel Next: Configure Security Settings

Figure 10-5: Create Network Load Balancer: Step-3, Listeners and Availability Zones.

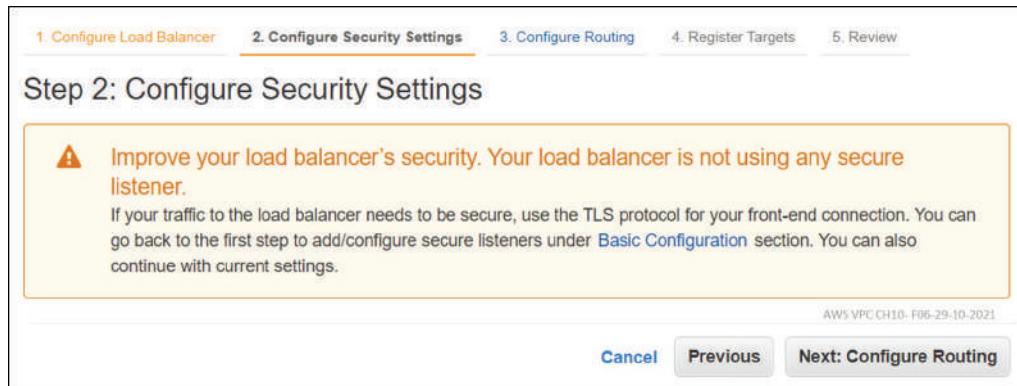


Figure 10-6: Create Network Load Balancer: Step-4, Security Settings.

In step 3, we configure NLB routing policy for the ingress traffic received from the front-end side. First, Create a Target Group. Then name it and define the service port and protocol. Select the Instance options as Target Type. We will observe the service availability using NLB Health Check. In our example, we are using http requests in a 30-second interval. By doing this, we confirm that the actual service running on an EC2 instance is available. We can leave the rest of the fields with their default values. Next, move to the target registration phase.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Routing **4. Register Targets** 5. Review

Step 3: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify here. It also performs health checks on the targets using these settings. The target group you specify in this step will apply to all of the listeners configured on this load balancer. You can edit or add listeners after the load balancer is created.

Target group

Target group: New target group
Name: NWKT-TG1

Target type: Instance

Protocol: TCP
Port: 80

Health checks

Protocol: TCP

Advanced health check settings:

- Port: traffic port (selected)
- override

Healthy threshold: 3
Unhealthy threshold: 3
Timeout: 10 seconds
Interval: 30 seconds

AWS VPC CH10- F07-29-10-2021 [Cancel](#) [Previous](#) [Next: Register Targets](#)

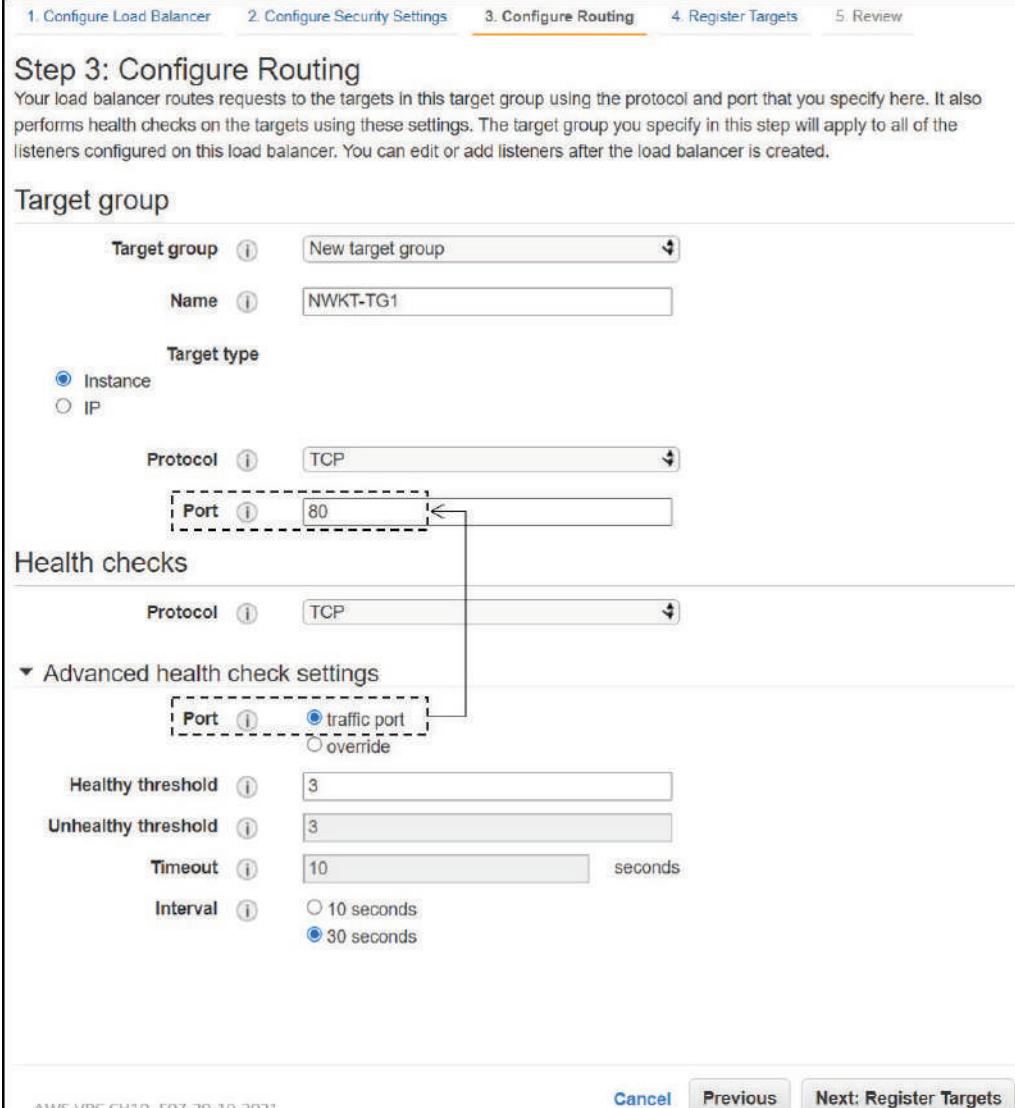


Figure 10-7: Create Network Load Balancer: Step-5, Target Group and Health Checks.

We have NWKT-EC2-01 instance running on the subnet 10.11.0.0/24, which we register as *Target*. We are using port 80 with it, just like we did with the Listener.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Routing 4. Register Targets 5. Review

Step 4: Register Targets

Configure Security Groups
The security groups for your instances must allow traffic from the VPC CIDR on the health check port.

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

<input type="checkbox"/> Instance	Name	Port	State	Security groups	Zone
<input type="checkbox"/> i-0535445b000...	NWKT-EC2...	80	● running	launch-wizard-2	eu-west-2c

Add to registered on port 80

<input type="checkbox"/> Instance	Name	State	Security	Zone	Subnet ID	Subnet CIDR
<input checked="" type="checkbox"/> i-053544...	NWKT-E...	● running	launch-wi...	eu-west-2c	subnet-0f9fb0b5459a...	10.11.0.0/24

AWS VPC CH10- F08-29-10-2021 [Cancel](#) [Previous](#) [Next: Review](#)

Figure 10-8: Create Network Load Balancer: Step-6, Register Targets.

The NLB setup is ready after we have registered our target instance. Now we can review the configuration before we launch the Network Load Balancer. Figure 10-9 shows the review.

The name of our internal NLB is NWKT-LB. There is one front-end listener process that listens to TCP connection initiations to port 80. NLB back-end is associated with the subnet NWKT-Test (10.11.0.0/24) in VPC NWKT-Test. The name of our Target Group is NWKT-TG1, and NLB routes ingress HTTP requests traffic towards target instances that are registered to Target Group. The Health Check protocol and port are the same that we use for traffic flows. Next, click the *Create* button to launch the NLB.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Routing 4. Register Targets 5. Review

Step 5: Review

Please review the load balancer details before continuing

▼ Load balancer Edit

Name: NWKT-LB
Scheme: internal
Listeners: Port:80 - Protocol:TCP
IP address type: ipv4
VPC: vpc-0867c98f35d11fcdd (NWKT-Test)
Subnets: subnet-0f9fb0b5459ab07b4 (NWKT-Test)
Tags:

▼ Routing Edit

Target group: New target group
Target group name: NWKT-TG1
Port: 80
Target type: Instance
Protocol: TCP
Health check protocol: TCP
Health check port: traffic port
Healthy threshold: 3
Unhealthy threshold: 3
Interval: 30

▼ Targets Edit

Instances: i-0535445b000f480d3 (NWKT-EC2-01):80

AWS VPC CH10- F09-29-10-2021 Cancel Previous Create

Figure 10-9: Create Network Load Balancer: Step-7, Launch NLB.

You will get a notification message after the NLB is created.

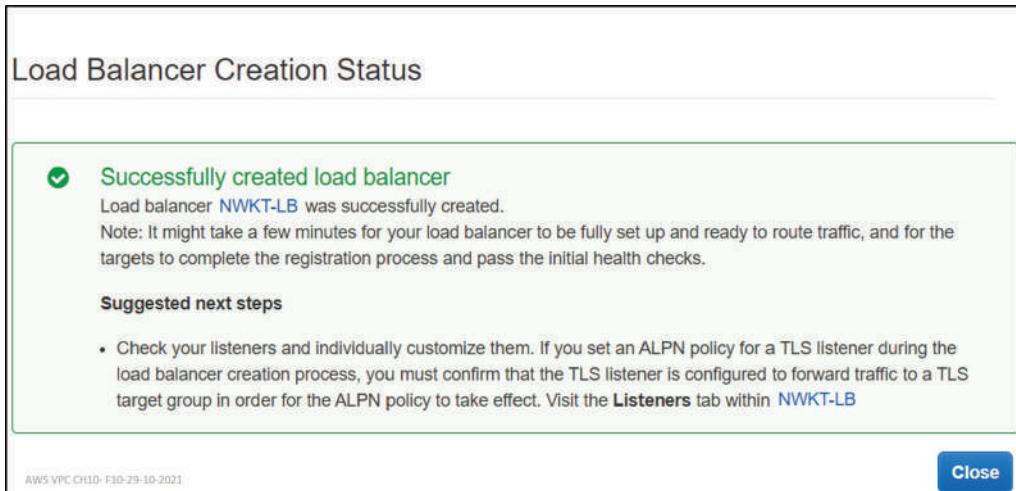


Figure 10-10: Create Network Load Balancer: Step-9, Verification.

Figures 10-11 shows that our NLB is now listed in the Load Balancers view. The Description tab includes the basic NLB information.

Name	NWKT-LB
ARN	arn:aws:elasticloadbalancing:eu-west-2:017857243309:loadbalancer/net/NWKT-LB/960d5399db99479a
DNS name	NWKT-LB-960d5399db99479a.elb.eu-west-2.amazonaws.com (A Record)
State	Active
Type	network
Scheme	internal
IP address type	ipv4
VPC	vpc-0867c98f35d11fcdd
Availability Zones	subnet-0f9fb0b5459ab07b4 - eu-west-2c IPv4 address: Assigned from CIDR 10.11.0.0/24 Private IPv4 address: Assigned from CIDR 10.11.0.0/24
Edit subnets	
Hosted zone	ZD4D7Y8KGAS4G
Creation time	October 29, 2021 at 2:08:31 PM UTC+3
Attributes	
Deletion protection	Disabled
Cross-zone load balancing	Disabled

AWS VPC CH10- F11-29-10-2021

Figure 10-11: Network Load Balancer: Description Tab.

The Listeners tab in figure 10-12 shows the front-end listener settings. It also shows that the NLB forwards traffic initiated to this listener to one of the targets attached to target group NWKT-TG1.

Load balancer: NWKT-LB

AWS VPC CH10- F12-29-10-2021

Description **Listeners** Monitoring Integrated services Tags

Listeners listen for connection requests using their protocol and port. You can add, remove, or update listeners and lists.

To view and edit listener attributes, select the listener and choose Edit.

Add listener Edit Delete

Listener ID	Security policy	SSL Certificate	ALPN policy	Default action
TCP : 80 arn...e89bf7ff782b3d8f~	N/A	N/A	N/A	Forward to NWKT-TG1

Figure 10-12: Network Load Balancer: Listeners Tab.

You can start creating the *Endpoint Service* by clicking the Create Endpoint Service in the Integrated Service tab.

Load balancer: NWKT-LB

Description Listeners Monitoring **Integrated services** Tags

You can integrate the following AWS services with your load balancer. Integration status and details are displayed below. You set up and configure integration through these services.

Global Accelerator Create an accelerator to get static IP addresses that act ... [Learn more](#)

This load balancer is not associated with an accelerator.

[Create accelerator](#)

Config Records configuration changes to your load balancer and provides vi... [Learn more](#)

This load balancer is not currently monitored in Config.

[AWS Config settings](#)

VPC Endpoint Services (AWS PrivateLink) Establishes a private connectio... [Learn more](#)

This load balancer is not configured as an endpoint service.

[Create Endpoint Service](#)

AWS VPC CH10- F13-29-10-2021

Figure 10-13: Network Load Balancer: Integrated Services Tab.

Create Endpoint Service

After creating a Network Load Balancer, we create an Endpoint Service.

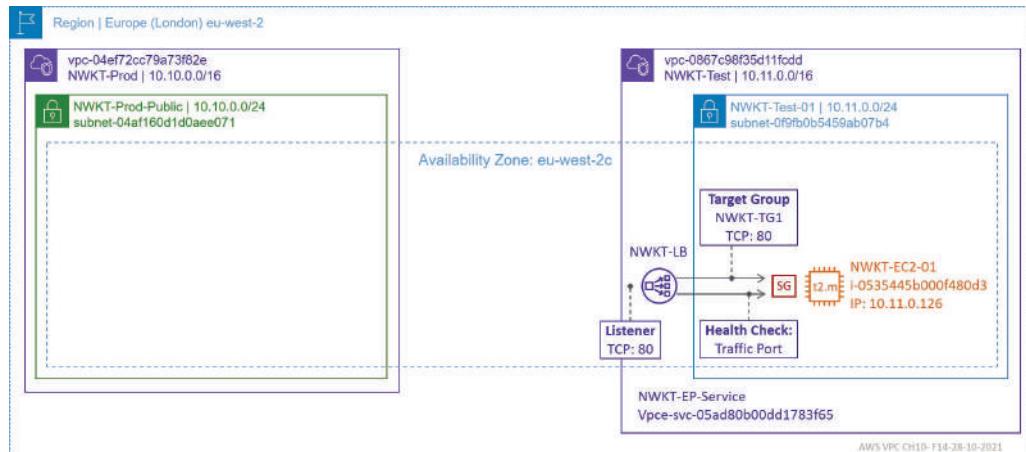


Figure 10-14: End Point Service.

Navigate to VPC Dashboard and select the *Endpoint Service* under the Virtual Private Cloud header. Click the *Create endpoint service* button.

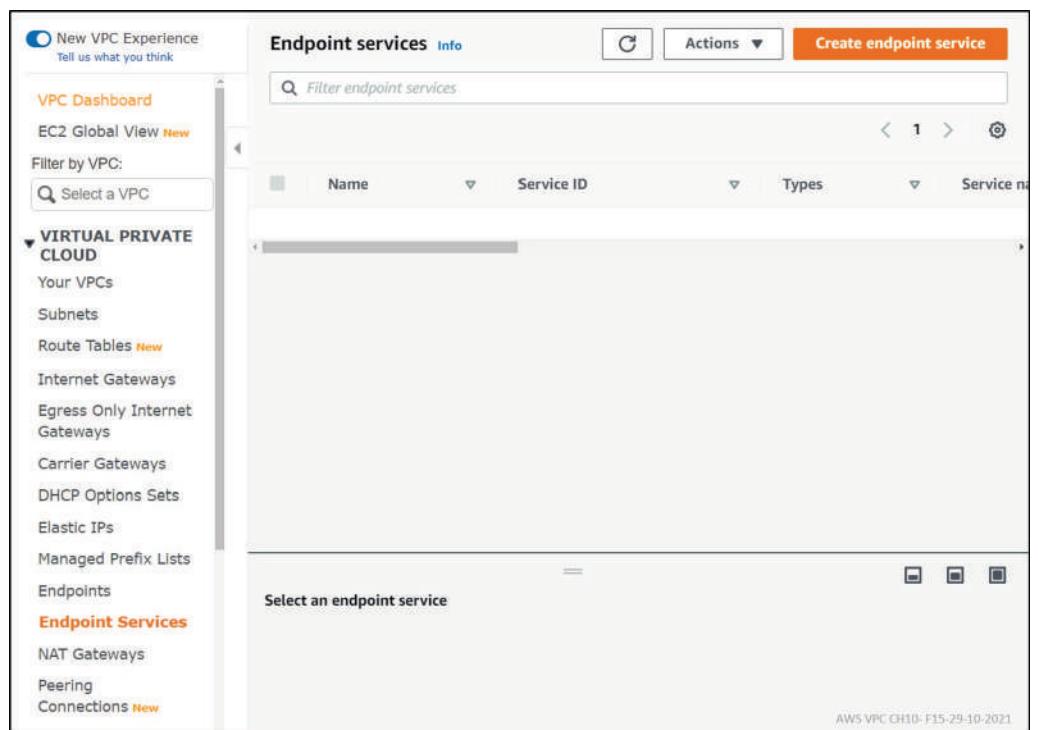


Figure 10-15: Create Endpoint Service.

Fill in the name field and set the Load balancer type to Network. Select our NLB from the NLB list (we only have one NLB).

The screenshot shows the 'Create endpoint service' wizard in the AWS VPC console. The current step is 'Endpoint service settings'. In the 'Name - optional' field, 'NWKT-EP-Service' is entered. Under 'Load balancer type', 'Network' is selected. The 'Available load balancers (1/1)' section shows a single entry: 'NWKT-LB' (eu-west-2c). A 'Details of selected load balancers' section below lists the selected load balancer and its included availability zones.

VPC > Endpoint services > Create endpoint service

Create endpoint service Info

Endpoint service settings

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

NWKT-EP-Service

Load balancer type

Network
 Gateway

Available load balancers (1/1)

Select the load balancers to send traffic from service consumers to your application or service.

Filter load balancers	Load balancer name	Availability Zones
	<input checked="" type="checkbox"/> NWKT-LB	eu-west-2c

Details of selected load balancers

Load balancers
The load balancers in which your service will be available.

NWKT-LB X
network

Included Availability Zones
The Availability Zones in which your service will be available.

- eu-west-2c (euw2-az1)

AWS VPC CH10- F16-29-10-2021

Figure 10-16: Create Endpoint Service: Select NLB.

Check the *Acceptance required* option. By doing this, we need to accept consumer requests before they can use the Endpoint Service. The Tag field shows the name tag you gave to Endpoint Service. To launch the Endpoint Service

Additional settings

Require acceptance for endpoint [Info](#)
Specify whether requests from service consumers to connect to your service through an endpoint must be accepted.

Acceptance required

Enable private DNS name
This option allows users of endpoints to use the specified private DNS name for access the service from their VPCs.

Associate a private DNS name with the service

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.

Key	Value - optional
<input type="text" value="Name"/> X	<input type="text" value="NWKT-EP-Service"/> X Remove

[Add new tag](#)

You can add 49 more tags.

AWS VPC CH10- F17-29-10-2021

[Cancel](#) [Create](#)

Figure 10-17: Create Endpoint Service: Select NLB.

Figure 10-18 shows the notification and summary window after the Endpoint Service is ready. When you later configure the Endpoint that will use this service, you need the service name. You can copy it from the *Endpoint Service* view. The name of our Endpoint Service is com.amazonaws.vpce.eu-west-2.vpc-svc-05ad80b0dd1783f65.

✓ You have successfully created endpoint service configuration for vpce-svc-05ad80b0dd1783f65 / NWKT-EP-Service.

vpce-svc-05ad80b0dd1783f65 / NWKT-EP-Service

[Actions ▾](#)

Details	
Service ID vpce-svc-05ad80b0dd1783f65	Types Interface
Network Load Balancers ARNs arn:aws:elasticloadbalancing:eu-west-2:017857243309:loadbalancer/net/NWKT-LB/960d5399db99479a	Gateway Load Balancers ARNs -
DNS names vpce-svc-05ad80b0dd1783f65.eu-west-2.vpce.amazonaws.com	Private DNS name -
Domain verification type Info -	Domain verification value Info -
	Availability Zones eu-west-2c (euw2-az1)
	Domain verification status Info -
	State Available
	Acceptance required Yes:
	Domain verification name Info -

[Copied](#)

AWS VPC CH10- F18-29-10-2021

Figure 10-18: Create Endpoint Service: Verification.

Create Endpoint

After creating an Endpoint Service, we create an *Interface Endpoint* to Service Consumer VPC NWKT-Prod and attach it to subnet 10.10.0.0/24 (NWKT-Prod-Public) with its *Elastic Network Interface (ENI)*. ENI gets an IP address from the IP range of the subnet 10.10.0.0/24. Next, we request permission to use Service Provider Endpoint Service, which we then, as an account and service owner, accept on the Service Consumer side. Like EC2 instances, Endpoints are protected by the Security Group (SG). In our example, we need to permit traffic using TCP port 80.

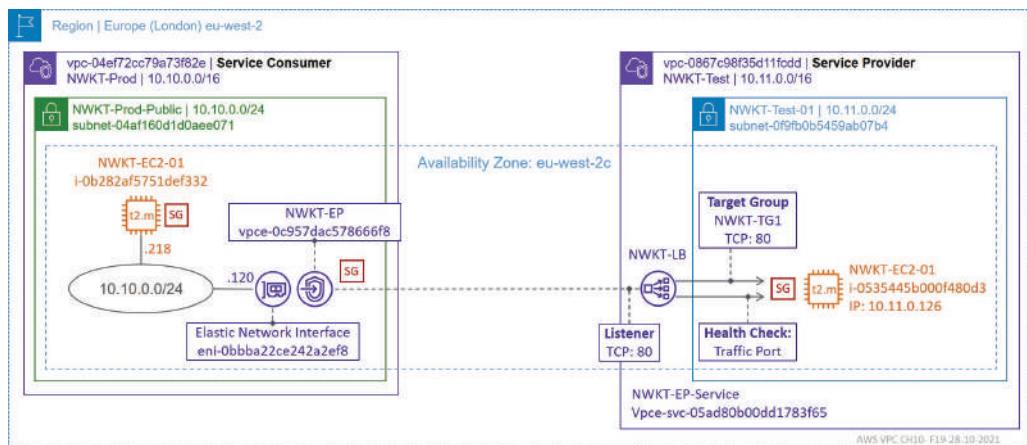


Figure 10-19: Create Endpoint.

Start by navigating to the VPC dashboard and select *Endpoints*. Click the *Create Endpoint* button.

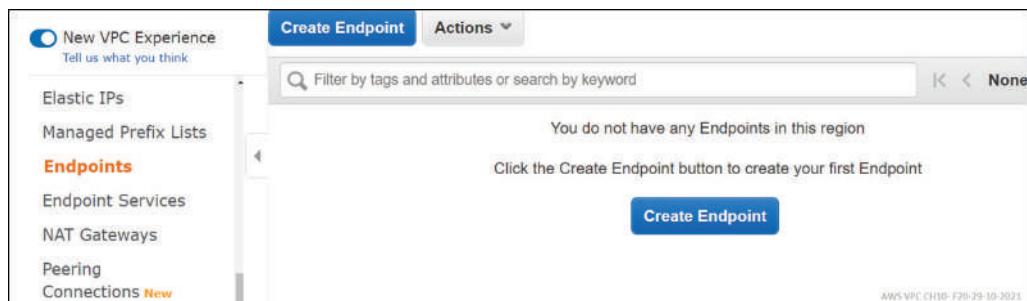


Figure 10-20: Create Endpoint.

Select *Find service by name* from the *Service category*. Then paste the service name that you have previously copied (in figure 10-18). Select the VPC where you launch the Endpoint (NWKT-Prod | vpc-04ef72cc79a73f82e). Then, select the subnet within the VPC (NWKT-Prod-Public | subnet-04af160d1d0aeee071) where the ENI attached to Endpoint will be associated. The NWKT-Prod-Public subnet is available at Availability Zone eu-west-2c (euw2-az1).

The screenshot shows the 'Create Endpoint' wizard in the AWS Management Console. The top navigation bar says 'Endpoints > Create Endpoint'. The main title is 'Create Endpoint'. A descriptive text explains that a VPC endpoint enables secure connectivity between your VPC and another service. It notes three types: Interface endpoints, Gateway Load Balancer endpoints, and gateway endpoints. Interface and Gateway Load Balancer endpoints are powered by AWS PrivateLink and use an elastic network interface (ENI) as an entry point for traffic destined to the service. Interface endpoints are typically accessed via public or private DNS names, while gateway endpoints serve as targets for routes in your route table.

Service category:

- AWS services
- Find service by name
- Your AWS Marketplace services

Service Name: Enter private service name and verify. i

com.amazonaws.vpce.eu-west-2.vpce-svc-05ad80bc

Service name found.

Verify

VPC*: vpc-04ef72cc79a73f82e C i

Subnets: subnet-04af160d1d0aeee071 i

Availability Zone	Subnet ID
<input type="checkbox"/> eu-west-2a (euw2-az2)	Service not supported in this Availability Zone
<input type="checkbox"/> eu-west-2b (euw2-az3)	Service not supported in this Availability Zone
<input checked="" type="checkbox"/> eu-west-2c (euw2-az1)	subnet-04af160d1d0aeee071 (NWKT-Prod-Public) ▼

AWS VPC CH10-F21-29-10-2021

Figure 10-21: Create Endpoint – Step1: Define Service, VPC, and the Subnet.

Note that I have divided the *Create Endpoint* window into two figures (figure 10-21 and 10-22). We have pre-configure Security Group NWKT-EP-SG, which we use with Endpoint. After filling in all the information, click the Create endpoint.

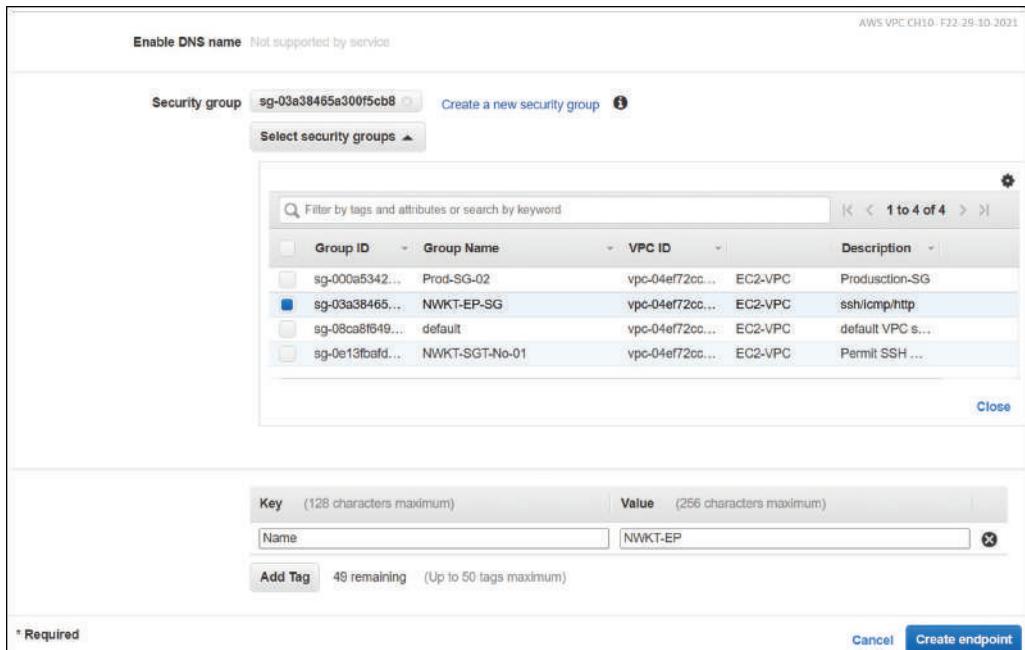


Figure 10-22: Create Endpoint – Step2: Security Group.

Figure 10-23 verifies that our Endpoint is created. The Endpoint Id is vpce-0c957dacf578666f8.

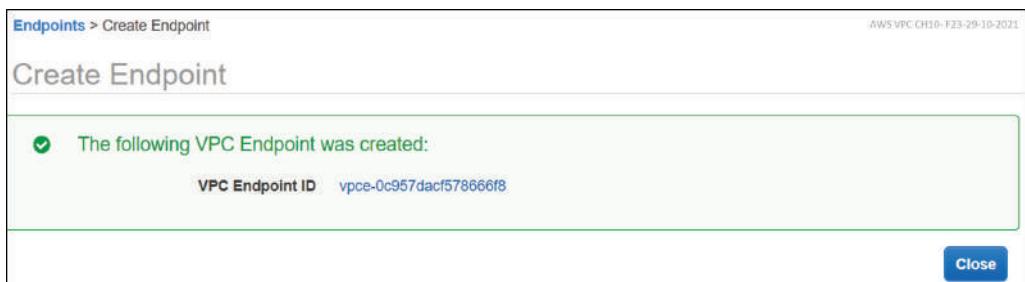


Figure 10-23: Create Endpoint – Step3: Verification.

Figure 10-24 show the details of our new Endpoint NWKT-EP (vpce-0c957dacf578666f8). We haven't accepted the connection request, and that is why the state is Pending Acceptance for the Endpoint Service com.amazonaws.vpce.eu-west-2.vpce-svc-05ad80b0dd1783f65. Details tab also shows the type of the endpoint, which in our example is Interface (there is also Gateway Endpoints in AWS). We can also see that the endpoint is running on a VPC NWKT-Prod (vpc-04ef72cc79a73f82e). In addition, the Details tab shows the Service name and DNS names.

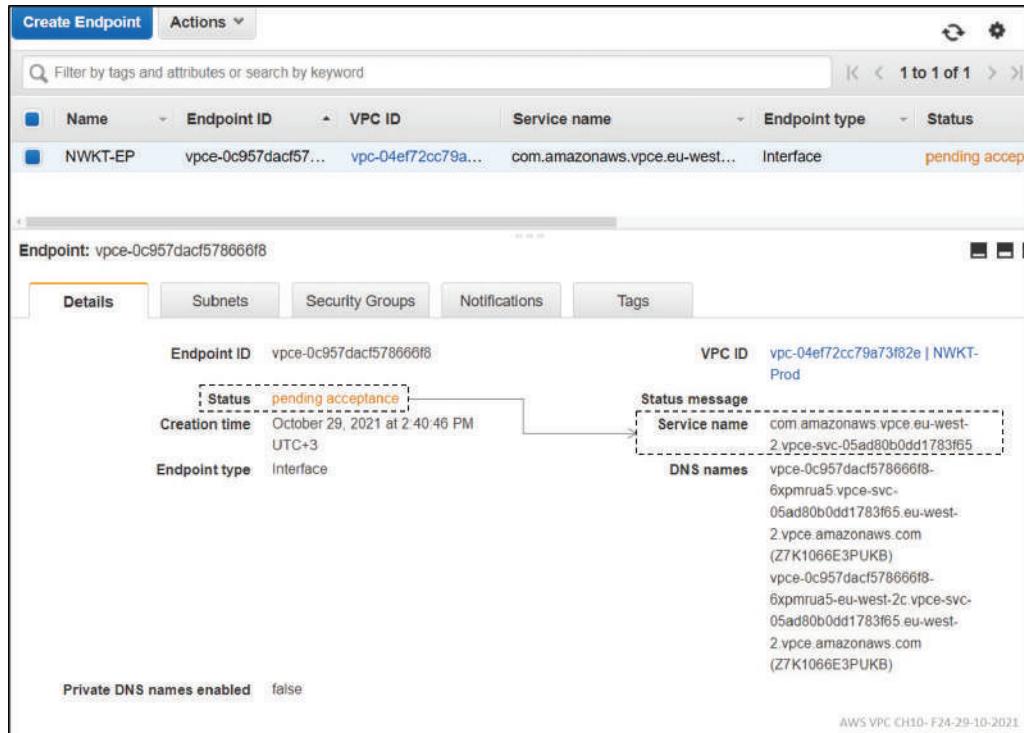


Figure 10-24: Endpoint Details

To accept the endpoint connection request, go to the Endpoint service view and select the NWKT-EP-Service. Open the Endpoint connection tab. Then, choose the *Accept endpoint connection request* option from the Action drop-down menu.

The screenshot shows the AWS VPC Endpoint Services console. At the top, there's a header bar with 'Endpoint services (1/1)' and a 'Create endpoint service' button. Below the header is a search bar labeled 'Filter endpoint services'. The main table has columns: Name, Service ID, Types, and Service name. One row is selected, showing 'NWKT-EP-Service', 'vpce-svc-05ad80b0dd1783f65', 'Interface', and 'com.amazonaws.vpce.eu-west-2.vpc'. At the bottom of the table, it says 'AWS VPC CH10-F25-29-10-2021'. Below the table, the endpoint ID 'vpce-svc-05ad80b0dd1783f65 / NWKT-EP-Service' is displayed. Underneath, there are tabs: Details, Load balancers, Allow principals, Endpoint connections (which is highlighted in orange), Notifications, and Tags. The 'Endpoint connections' tab shows a sub-table with columns: Endpoint ID, Owner, and State. One entry is listed: 'vpce-0c957dacf578666f8', '123456654321', and 'Pending acceptance'. There are two buttons above this table: 'Accept endpoint connection request' (highlighted with a dashed box) and 'Reject endpoint connection request'.

Figure 10-25: Accept Endpoint Connection Request – Phase 1.

You have to confirm the acceptance before clicking the *Accept* button.

The screenshot shows a confirmation dialog titled 'Accept endpoint connection request'. It asks 'Are you sure you want to accept this endpoint connection request?'. Below this is a 'Summary' section with four fields: 'Endpoint ID' (vpce-0c957dacf578666f8), 'Endpoint service ID' (vpce-svc-05ad80b0dd1783f65), 'Endpoint service name tag' (NWKT-EP-Service), and 'Endpoint service name' (com.amazonaws.vpce.eu-west-2.vpce-svc-05ad80b0dd1783f65). At the bottom, it says 'To confirm accept, type accept in the field:' followed by a text input field containing 'accept'. There are 'Cancel' and 'Accept' buttons at the bottom right. A timestamp 'AWS VPC CH10-F26-29-10-2021' is visible at the very bottom left.

Figure 10-26: Accept Endpoint Connection Request – Phase 2.

The Endpoint connection tab in figure 10-27 verifies that the state of the Endpoint connection from the Endpoint `vpce-0c957dacf578666f8` to Endpoint Service is *Available*.

The screenshot shows the AWS VPC console interface. At the top, there's a header for "Endpoint services (1/1) Info". Below it is a search bar with the placeholder "Filter endpoint services". To the right of the search bar are buttons for "Actions" and "Create endpoint service". A table below lists one endpoint service: "NWKT-EP-Service" with "Service ID" `vpce-svc-05ad80b0dd1783f65`, "Types" `Interface`, and "Service name" `com.amazonaws.vpce.eu-west-2.vpce-...`. The table has columns for "Name", "Service ID", "Types", and "Service name". The "Service name" column shows a truncated version of the full service name.

At the bottom of the main pane, there's a timestamp: "AWS VPC CH10- F27-29-10-2021".

Below the main header, the title "vpce-svc-05ad80b0dd1783f65 / NWKT-EP-Service" is displayed. Underneath, there are tabs: "Details", "Load balancers", "Allow principals", "Endpoint connections" (which is highlighted in orange), "Notifications", and "Tags".

In the "Endpoint connections" section, there's a sub-header "Endpoint connections (1) Info" with a search bar. The table lists one connection: "Endpoint ID" `vpce-0c957dacf578666f8`, "Owner" `123456654321`, and "State" `Available` (indicated by a green circle icon).

Figure 10-27: Endpoint Connection from the Endpoint Service Perspective.

Figure 10-28 shows that the Status of the Endpoint connection is also available from the Endpoint perspective.

The screenshot shows the AWS VPC console interface. At the top, there's a header for "Endpoint: vpce-0c957dacf578666f8". Below it is a navigation bar with tabs: "Details" (which is highlighted in orange), "Subnets", "Security Groups", "Notifications", and "Tags". To the right of the tabs, there's a timestamp: "AWS VPC CH10- F28-29-10-2021".

The main content area displays endpoint details. On the left, there's a table with columns: "Name", "Endpoint ID", "VPC ID", "Service name", and "Endpoint type". One row is shown: "NWKT-EP" with "Endpoint ID" `vpce-0c957dacf578666f8`, "VPC ID" `vpc-04ef72cc79a73f82e | NWKT-Prod`, "Service name" `com.amazonaws.vpce.eu-west...`, and "Endpoint type" `Interface`.

On the right, there's a detailed view of the endpoint. It shows the "Status" as `available` (highlighted with a dashed box). Below it, the "Creation time" is listed as "October 29, 2021 at 2:40:46 PM UTC+3". The "Endpoint type" is listed as "Interface".

Further down, there's a "Status message" box containing "Service name" `com.amazonaws.vpce.eu-west-2.vpce-svc-05ad80b0dd1783f65` and "DNS names" `vpce-0c957dacf578666f8-`.

Figure 10-28: Endpoint Connection from the Endpoint Perspective.

The subnet tab shows that we have attached our endpoint to subnet 10.10.0.0/24 (NWKT-Prod-Public | subnet-04af160d1d0aee071), which is in AZ eu-west-2c. The IP address of the endpoint is 10.10.0.120, and the Elastic Network Interface ID for an endpoint is eni-0bbba22ce242a2ef8.

The screenshot shows the AWS VPC console with the 'Subnets' tab selected. A table displays subnet information for the endpoint. One row is highlighted, showing the Network Interface ID as eni-0bbba22ce242a2ef8, which is enclosed in a dashed red box. The table has columns: Subnet ID, Availability Zone, IPv4 Addresses, IPv6 Addresses, and Network Interface ID.

Subnet ID	Availability Zone	IPv4 Addresses	IPv6 Addresses	Network Interface ID
subnet-04af160d1d0aee071	eu-west-2c (euw2-az1)	10.10.0.120	-	eni-0bbba22ce242a2ef8

Figure 10-29: Subnet Information Related to Endpoint.

You can click the Network Interface ID eni-0bbba22ce242a2ef8 to see the details of the ENI. Figures 10-30 and 10-31 on the next page show the information related to ENI. Note that once again, I have divided the view into two figures.

The screenshot shows the AWS VPC Network Interfaces page. At the top, there is a search bar with the placeholder "Filter network interfaces" and a "Create network interface" button. Below the search bar, a table lists one network interface:

Name	Network interface ID	Subnet ID	VPC ID	Available
-	eni-0bbba22ce242a2ef8	subnet-04af160d1d0aee071	vpc-04ef72cc79a73f82e	eu-west-2c

Below the table, a section titled "Network interface: eni-0bbba22ce242a2ef8" is expanded. It contains three tabs: "Details" (selected), "Flow logs", and "Tags".

The "Details" tab displays the following information:

Network interface details		
Network interface ID eni-0bbba22ce242a2ef8	Name -	Description VPC Endpoint Interface vpc-0c957dacf578666fb
Network interface status In-use	Interface type vpc_endpoint	Security groups sg-03a38465a300f5cb8 (NWKT-EP-SG)
VPC ID vpc-04ef72cc79a73f82e	Subnet ID subnet-04af160d1d0aee071	Availability Zone eu-west-2c
Owner 123456654321	Requester ID 123456654321	Requester-managed True
Source/dest. check: True	AWS VPC CH10- F30-29-10-2021	

Figure 10-30: Information Related to Endpoint Elastic Network Interface (ENI).

▼ IP addresses		
Private IPv4 address 10.10.0.120	Private IPv4 DNS ip-10-10-0-120.eu-west-2.compute.internal	Elastic Fabric Adapter False
Public IPv4 address	Public IPv4 DNS	IPv6 addresses
Secondary private IPv4 addresses	Association ID	Elastic IP address owner
MAC address 02:c0:69:bc:b7:1c	IPv4 Prefix Delegation	IPv6 Prefix Delegation
▼ Instance details		
Instance ID	Instance owner amazon-aws	Device index 1
Allocation ID		
▼ Network interface attachment		
Attachment status Attached	Attachment ID ela-attach-0475f98d5206f3267	Attachment time
Delete on termination False		AWS-VPC-CH10-F31-29-10-2021

Figure 10-31: Information Related to Endpoint Elastic Network Interface (ENI) continues.

Connection Verification

Our example service instance NWKT-EC2-01 is a free tier Linux machine, and we are not running any HTTP server on it. That is why it doesn't listen to TCP session initiation to port 80. However, it listens to TCP port 22, so we can test the connection by using SSH.

[ec2-user@ip-10-11-0-126 ~]\$ netstat -lntu					
Active Internet connections (only servers)					
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::111	:::*	LISTEN
udp	0	0	127.0.0.1:323	0.0.0.0:*	
udp	0	0	0.0.0.0:962	0.0.0.0:*	
udp	0	0	0.0.0.0:68	0.0.0.0:*	
udp	0	0	0.0.0.0:111	0.0.0.0:*	
udp6	0	0	:::1:323	:::*	
udp6	0	0	:::962	:::*	
udp6	0	0	fe80::d3:c6ff:fe7d::546	:::*	
udp6	0	0	:::111	:::*	

Example 10-1: Information Related to Endpoint Elastic Network Interface (ENI) continues.

We used the TCP port 80 in our example configuration in Network Load Balancer Target Group (NWKT-TG1), Health Check, and Listener process. Figure 10-32 shows the new Target Group (NWKT-TG-2), which I have created for ssh testing. It uses TCP port 22.

search : arn:aws:elasticloadbalancing:eu-west-2:... Add filter

Name	Port	Protocol	Target type	Load Balancer	VPC
NWKT-TG-2	22	TCP	instance	NWKT-LB	vpc-0867c98f35d11fcdd

Target group: NWKT-TG-2

Description Targets Health checks Monitoring Tags

Basic Configuration

Name	NWKT-TG-2
ARN	arn:aws:elasticloadbalancing:eu-west-2:017857243309:targetgroup/NWKT-TG-2/44c9564e714275de
Protocol	TCP
Port	22
Target type	instance
VPC	vpc-0867c98f35d11fcdd
Load balancer	NWKT-LB

AWS VPC CH10-F32-29-10-2021

Figure 10-32: New Target Group for SSH Testing.

Figure 10-33 verifies that Health Check automatically starts using TCP port 22 for service availability monitoring.

Name	Port	Protocol	Target type	Load Balancer	VPC ID
NWKT-TG-2	22	TCP	instance	NWKT-LB	vpc-0867c98f35d11fcdd

Target group: NWKT-TG-2

Description Targets **Health checks** Monitoring Tags

Protocol: TCP
Port: 22
Healthy threshold: 3
Unhealthy threshold: 3
Timeout: 10
Interval: 30

Edit health check

AWS VPC CH10-F33-29-10-2021

Figure 10-33: Health Check.

The Targets tab shows that the instance NWKT-EC2-01 is healthy.

Target group: NWKT-TG-2																	
Edit																	
Description Targets Health checks Monitoring Tags																	
The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.																	
Registered targets <table border="1"> <thead> <tr> <th>Instance ID</th> <th>Name</th> <th>Port</th> <th>Availability Zone</th> <th>Status</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>I-0535445b000f480d3</td> <td>NWKT-EC2-01</td> <td>22</td> <td>eu-west-2c</td> <td>healthy</td> <td>This target is currently passing target group's health checks.</td> </tr> </tbody> </table>						Instance ID	Name	Port	Availability Zone	Status	Description	I-0535445b000f480d3	NWKT-EC2-01	22	eu-west-2c	healthy	This target is currently passing target group's health checks.
Instance ID	Name	Port	Availability Zone	Status	Description												
I-0535445b000f480d3	NWKT-EC2-01	22	eu-west-2c	healthy	This target is currently passing target group's health checks.												
Availability Zones <table border="1"> <thead> <tr> <th>Availability Zone</th> <th>Target count</th> <th>Healthy?</th> </tr> </thead> <tbody> <tr> <td>eu-west-2c</td> <td>1</td> <td>Yes</td> </tr> </tbody> </table>						Availability Zone	Target count	Healthy?	eu-west-2c	1	Yes						
Availability Zone	Target count	Healthy?															
eu-west-2c	1	Yes															

Figure 10-34: Target of NWKT-TG-2 - Availability.

Monitoring Tab verifies that after changing the Target Group TCP port from 80 (http) to 22 (ssh), our EC2 is available and healthy.

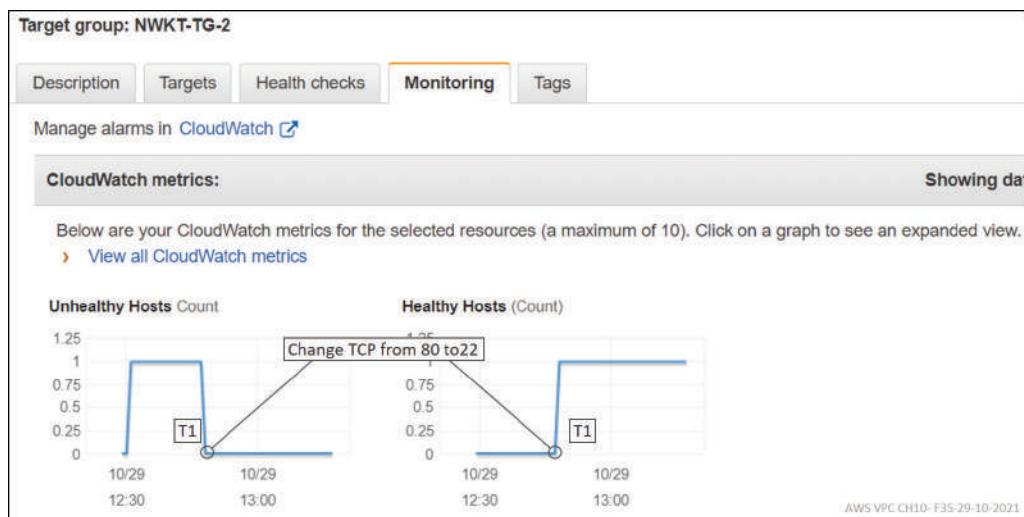


Figure 10-35: Monitoring Tab.

We have one thing to do before we can test the ssh from the Consumer VPC NWKT-Prod to Service Provider VPC NWKT-Test. We need to change the Listener process to listen to TCP session initiation to port 22. Figure 10-36 shows our modified Listener values where NLB forwards ssh session initiation to one of the instances (we only have one) belonging to Target Group NWKT-TG-2.

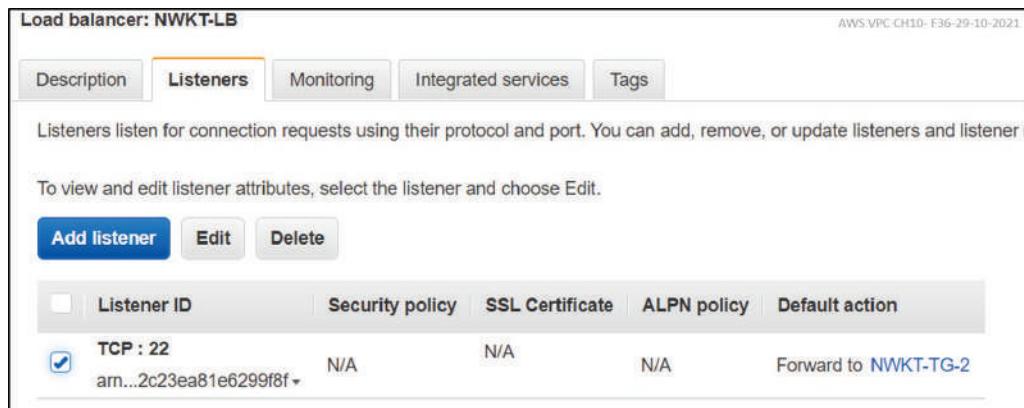


Figure 10-36: Listener Process

Now we are ready to test our PrivateLink. Example 10-2 shows what happens when we open ssh session from the EC2 instance 10.10.0.218 (Service Consumer VPC NWKT-Prod/subnet NWKT-Prod-Public) to Endpoint Elastic ENI IP 10.10.0.120 (also in the same VPC/Subnet).

Endpoint forwards our ssh session initiation to Load Balancer, which does the source NAT and forwards session initiation to instance NWKT-EC2-01 10.11.0.126. That verifies that our VPC PrivateLink is working.

```
[ec2-user@ip-10-10-0-218 ~]$ ssh -i "NWKT-KEY-EC2.pem" ec2-user@10.10.0.120
Last login: Fri Oct 29 12:51:25 2021 from 10.11.0.104
[ec2-user@ip-10-11-0-126 ~]$
```

_ _|_)
_ | (_ / Amazon Linux 2 AMI
_ | _ | _ |

<https://aws.amazon.com/amazon-linux-2/>

Example 10-2: SSH from 10.10.0.218 to 10.11.0.126.

Billing

Figure 10-37 shows the cost components related to PrivateLink.

AWS Service Charges		\$18.90
▶ Data Transfer		\$0.00
▶ Elastic Compute Cloud		\$7.72
▼ Elastic Load Balancing		\$0.08
▶ EU (London)		\$0.08
Elastic Load Balancing - Network		\$0.08
\$0.0060 per used Network load balancer capacity unit-hour (or partial hour)	0.000053 LCU-Hrs	\$0.00
\$0.02646 per Network LoadBalancer-hour (or partial hour)	3.000 Hrs	\$0.08
▶ Key Management Service		\$0.00
▶ Virtual Private Cloud		\$7.53
Taxes		
VAT to be collected		\$3.57

Figure 10-37: PrivateLink and NLB Pricing.

Chapter 11: Dedicated Direct Connect & Transit VIF

Introduction

AWS Direct Connect (DX) is such a broad topic that I decided to break it into five chapters. This chapter introduces the *Dedicated Direct Connect (DDX)* with Transit Gateway association using *Transit Virtual Interface (TVIF)*. Chapter 12 introduces a *Hosted Direct Connect (HDX)*. The focus of chapter 13 is on BGP routing over Direct Connect. Chapter 14 explains how we can use *Direct Connect Gateway (DXGW)* as a site-to-site hub using a *Direct Connect SiteLink*. Chapter 15 explains *Public Virtual Interface (PVIF)*, which we use to create a connection to Amazon public service bypassing the Internet.

Dedicated Direct Connect Connection

Figure 11-1 illustrates our example solution, where we have a customer's edge router CE-1 in Equinix LD5, Slough, UK Datacenter. Equinix LD5 Datacenter is one of the four *AWS Direct Connection Locations (AWS-DCL)* in the AWS eu-west-2 (London) region in which Amazon has prepended its backbone network with its edge router (EqLD5-xyz in figure 11-1). Equinix, in turn, is one of the *AWS Direct Connect Partners (AWS-DCP)*, which all are members of the *AWS Partner Network (APN)*. On the AWS side, we have our VPC NWKT (CIDR Range 10.10.0.0/24) attached to Transit Gateway (TGW). We associate our TGW to *Direct Connect Gateway (DXGW)*, a global AWS service, to which we can attach TGWs from any AWS Region. Our customer router CE-1 is connected to AWS physical device EqLD5-xyz using fiber optic cable (1310 nm SM/1000Base-LX). The glue between the router CE-1 and DXGW is a Virtual Interface (VIF). VIF defines the VLAN Identifier, which we use to extend the broadcast domain from the AWS Logical Device (ALD) TCSH-1a2b3c4d, running on AWS physical device EqLD5-xyz, to customer router CE-1. We need the broadcast domain (=subnet) for eBGP peering between CE-1 and TCSH-1a2b3c4d. VIF also defines the BGP Autonomous System Number (ASN) of customer router CE-1 and the IP addressing scheme for eBGP peering. Note that the Amazon side ASN is defined in DXGW configuration. On the Amazon side, we attach VIF to DXGW, and that's basically is it. We have a dedicated connection, bypassing the Internet, from our CE-1 router to VPC.

Figure 11-1 illustrates the configuration process. The following sections introduce each of these steps in detail.

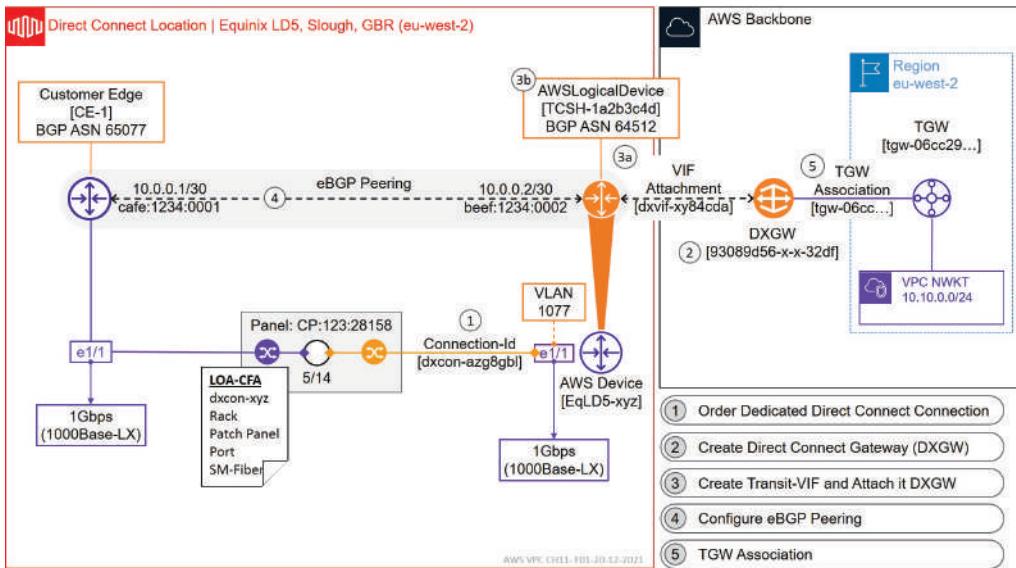


Figure 11-1: AWS Dedicated Direct Connect Connection & Transit Virtual Interface.

Direct Connect Ordering Process

Figure 11-2 illustrates the example Dedicated Direct Connect connection. Our edge device CE-1 is in the Direct Connect Location DC and we don't have to use AWS Direct Connect Partner infrastructure for AWS device cross-connect. The 1 Gbps dedicated connection, which we are ordering, requires a 1000Base-LX SFP optical transceiver and 1310nm Single-Mode optical fiber. The 10 Gbps dedicated connection, in turn, requires a 10Gbase-LR SFP+ optical transceiver and 1310nm Single-Mode optical fiber. First, we order a Dedicated DX connection from Amazon. In this phase, we define the AWS-DCL (Equinix LD5 Datacenter) and cross-connection port speed (1Gbps). Note that available cross-connect port speed options are 1Gbps, 10Gbps, and 100Gbps in this DCL. When Amazon has validated our order (it may take up to three days), we can download the *Letter of Authorization – Connection Facility Assignment (LOA-CFA)* document. After downloading the LOA-CFA document, we send it to the Direct Connect Partner. The LOA-CFA document authorizes DCP to do the cross-connection. Besides, it gives instructions to where the cross-connect cable should be connected (Rack, Patch-Panel, Port) by their staff.

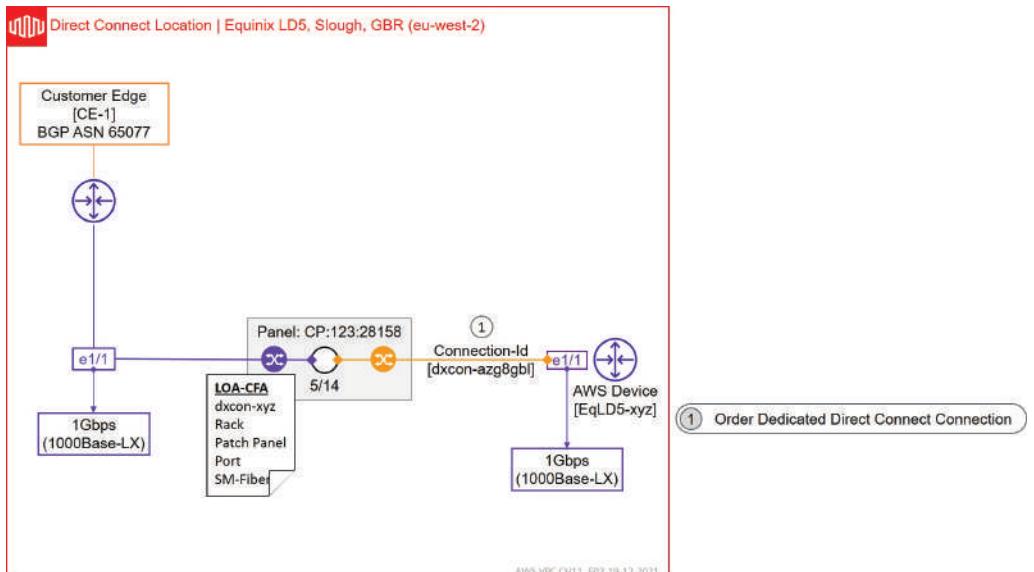


Figure 11-2: Create Dedicated Direct Connect.

As the first step, we need to order the Dedicated Direct Connect connection from Amazon. Start the process by selecting the *Networking & Content Delivery* option from the *Services* menu. Then, select the *Direct Connect* hyperlink from the right window.

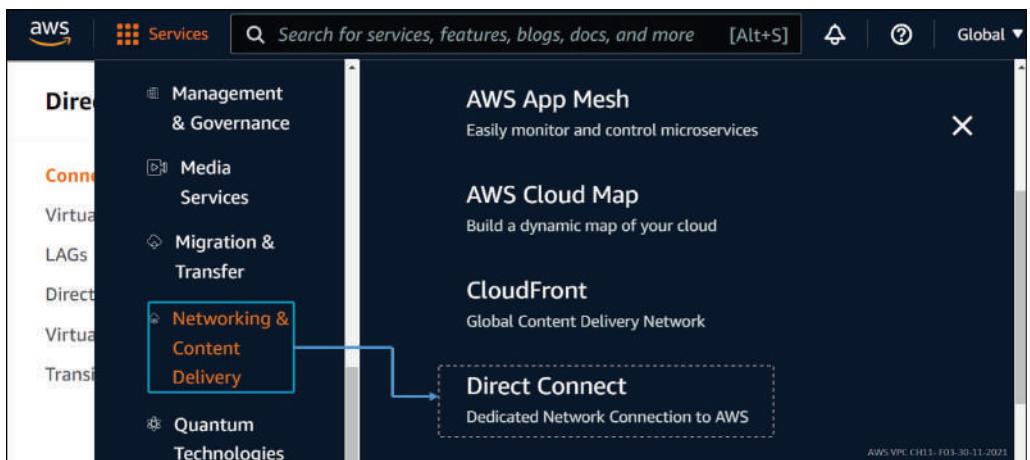


Figure 11-3: AWS Direct Connect: Direct Connection Ordering – Step#1.

Click the *Create connection* button in *Get Started* section.



Figure 11-4: AWS Direct Connect: Direct Connection Ordering – Step#2.

First, name the connection, then select the Direct Connect Location. Write the AWS Region name into the *Location* drop-down menu to see all available AWS DCLs in the selected region. As figure 11-5 shows, there are four AWS-DCL in the AWS region eu-west-2 (London). The only one where you can establish 100Gbps cross-connection to AWS Router is the Equinix LD5 Datacenter. We are using that one even though we are setting up the 1Gbps cross-connect.

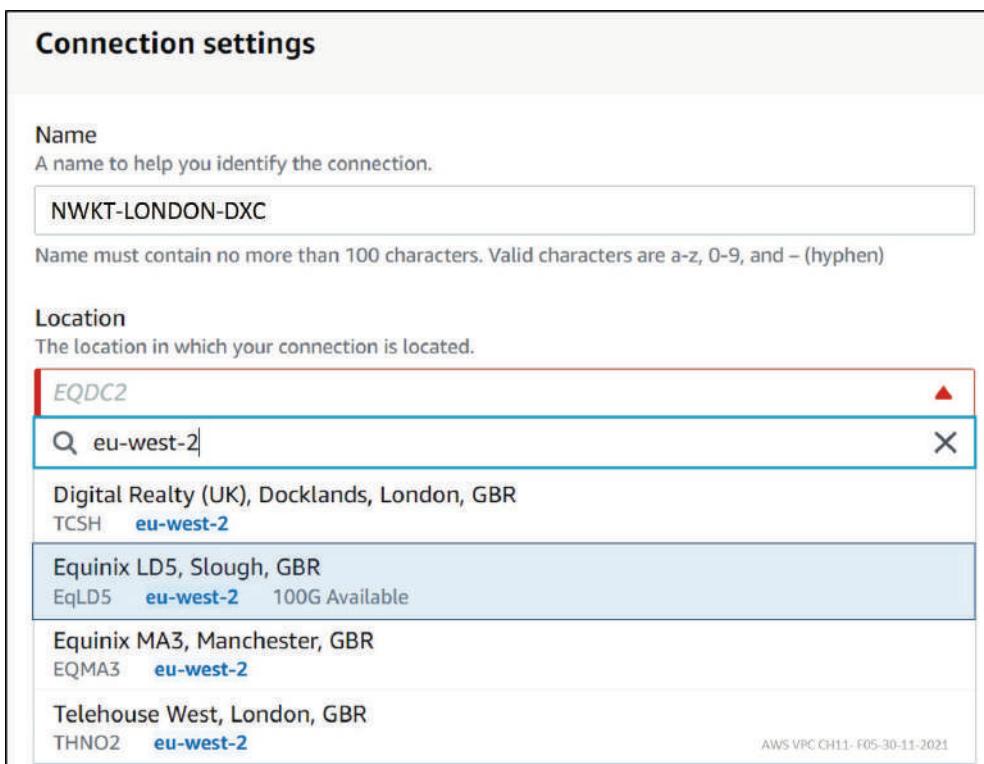


Figure 11-5: AWS Direct Connect: Direct Connection Ordering – Step#3.

After selecting the AWS-DCL, select the port speed. In this example, we are ordering a straight cross-connection between the customer device and AWS router without AWS-DCP in between them. Uncheck the *Connect through an AWS Direct Connect Partner* under the *on-Premises* option. You can configure an LACP (Ethernet Channel) and MACsec (link-level encryption) under the Additional Settings section. We are using neither of them. Now everything is in place, and we may continue the order process by clicking the *Create connection* button. Note that I only explain how to create Direct Connect without actually doing it.

Connection settings

Name
A name to help you identify the connection.

Name must contain no more than 100 characters. Valid characters are a-z, 0-9, and – (hyphen)

Location
The location in which your connection is located.

Port speed
Desired bandwidth for the new connection.
 1Gbps
 10Gbps
 100Gbps

On-premises
 Connect through an AWS Direct Connect partner.

Additional settings

AWS VPC CH11-F06-30-11-2021

Cancel **Create connection**

Figure 11-6: AWS Direct Connect: Direct Connection Ordering – Step#4.

Example 11-1 shows an example response to the AWS CLI command *aws direct connect describe-connection*. You can see the name of the AWS device, connection speed, AWS-DCL location, and the AWS region among the other information from the output. Note that the state of the connection is down at this moment because there is no cross-connection in place.

```
aws directconnect describe-connections

{
  "connections": [
    {
      "awsDevice": "EqLD5-xyz",
      "ownerAccount": "123456654321",
      "connectionId": "dxcon-azg8gb1",
      "lagId": "dxlag-xyz",
      "connectionState": "down",
      "bandwidth": "1Gbps",
      "location": "EqDC5",
      "connectionName": "NWKT-LONDON-DXC",
      "loaIssueTime": 1491528158.0,
      "region": "eu-west-2"
    }
  ]
}
```

Example 11-1: Direct Connect Verification using the AWS CLI.

You can download the LOA-CFA document after AWS has provisioned the interface for your connection in their AWS. To download the LOA-CFA document, navigate to *Direct Connect > Connections* window. Then select the Direct Connect connection which LOA-CFA you want to download. Note that we don't have existing connections in this example. Next, click the *View Details* button (it activates after you have selected the connection from the list) and then Select the *Download LOA-CFA* option. Note that the LOA-CFA document is valid for 90 days.

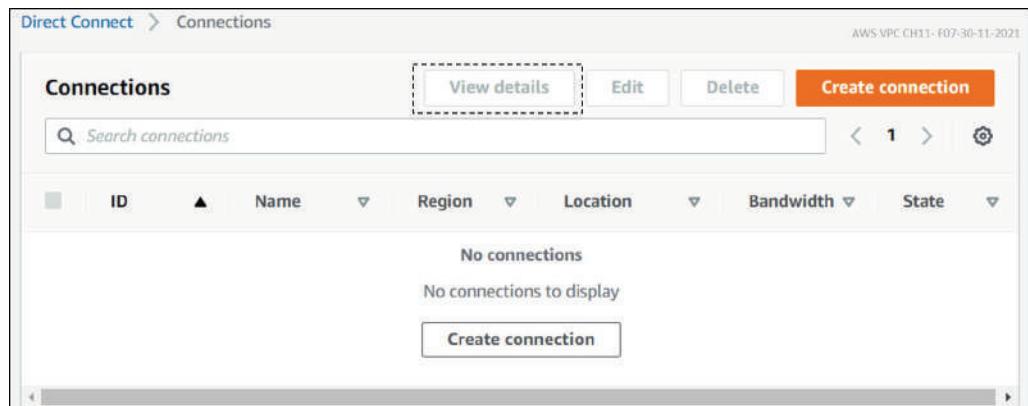


Figure 11-7: AWS Direct Connect – Direct Connection.

You can download the LOA-CFA document also by using the AWS CLI. The LOA-CFA content is base64-encoded. Example 11-2 shows the Windows command which extracts the loa.Content into the file NWKT-LD5_LOA-CFA.base64.

```
aws directconnect describe-loa --connection-id dxcon-azg8gb1 --output text --query loa.loaContent > NWKT-LD5_LOA-CFA.base64
```

Example 11-2: Download LOA-CFA with AWS CLI.

After downloading and extracting the file, you can create a pdf file from the loaContent using the Windows Certutil tool.

```
certutil -decode NWKT-LD5_LOA-CFA.base64 NWKT-LD5_LOA-CFA.pdf
```

Example 11-3: Create PDF File.

Figure 11-8 illustrates an example LOA-CFA document.

Letter of Authorization and Connection Facility Assignment	
Issue Date	Requested By
November 28, 2021	NWKT, TP
Issued By	Issued To
Xxxx, corp	IBX –Equinix LD5, Slough
Facility – Cage Number	AWS Direct Connect Id
Equinix LD5, Slough, GBR (eu-west-2)	dxcon-azg8gbl
Rack, Patch Panel, Port Number	Cable Type
Rack: 123 Patch Panel: CP:123:28158 Strands: 5/14	Single Mode Fiber
Access Ticket Number	
026745981123	

AWS VPC CH11- F08-30-11-2023

Figure 11-8: AWS Direct Connect – Direct Connection.

When you have got the LOA-CFA document, you need to send it to AWS-DCP. AWS maintains contact information concerning its Direct Connect Partners. You can find the Equinix London DCL contact information by navigating the AWS Direct connect service page and clicking the Cross connects-hyperlink (figure 11-9).

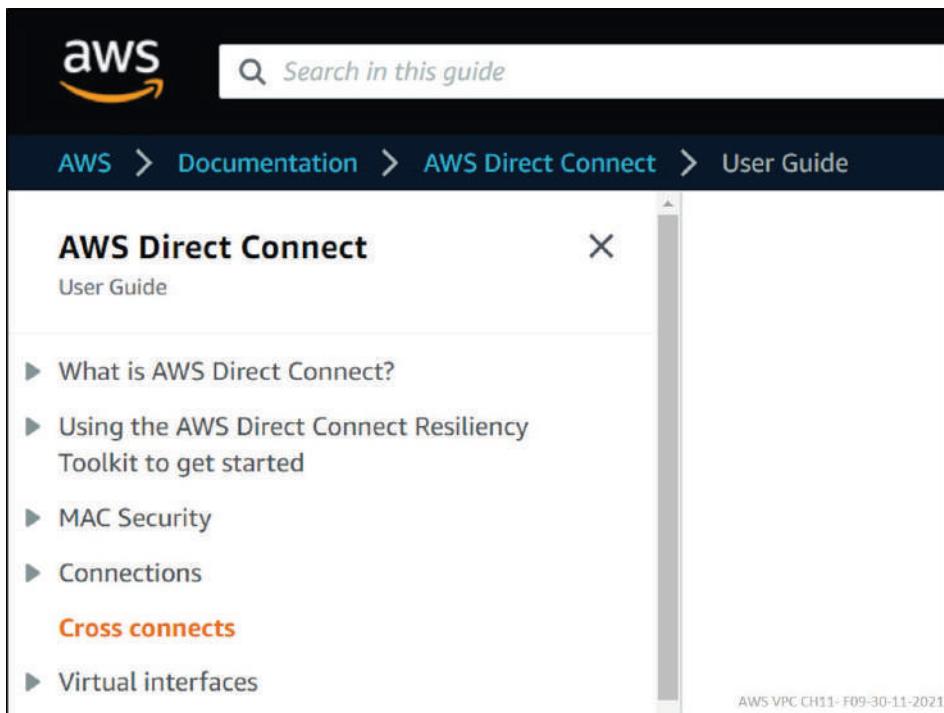


Figure 11-9: AWS Direct Connect – DCP Contact Information.

You will be forwarded to *Request cross-connects at the AWS Direct Connect locations* window. Select Europe (London) from the Contents section, and you find the contact email address of Equinix LD5, London Datacenter.

The screenshot shows the "Requesting cross connects at AWS Direct Connect locations" page. The left sidebar contains a "Contents" section with links to Africa (Cape Town) and Asia Pacific (Mumbai). The main content area is titled "Europe (London)". It contains a table with two columns: "Location" and "How to request a connection". The table rows are:

Location	How to request a connection
Digital Realty (UK), Docklands	Contact Digital Realty (UK) at amazon.orders@digitalrealty.com .
Equinix LD5, London (Slough)	Contact Equinix at awsdealreg@equinix.com .
Equinix MA3, Manchester	Contact Equinix at awsdealreg@equinix.com .
Telehouse West, London	Contact Telehouse UK at sales.support@uk.telehouse.net .

Figure 11-10: AWS Direct Connect – DCP Contact Information.

Create Direct Connect Gateway

As the next step, after we have ordered the Dedicated Direct Connect connection from Amazon and sent the LOA-CFA to AWS Direct Connect Partner, we create AWS Direct Connect Gateway (DXGW).

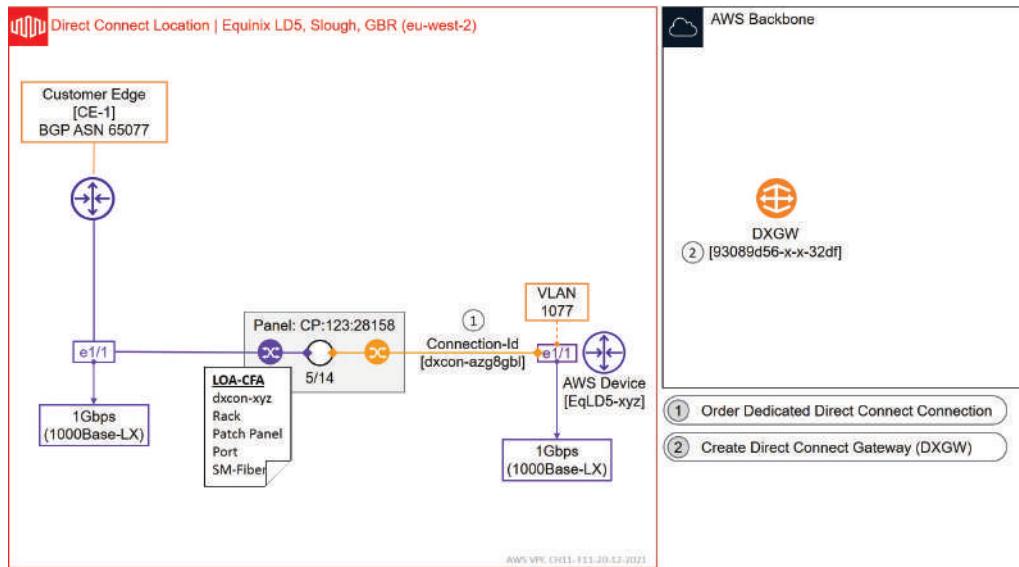


Figure 11-11: AWS Direct Connect – Create Direct Connect Gateway.

Go to *Direct Connect* service view and select *Direct Connect gateways*. Then click the *Create Direct Connect gateway* button.

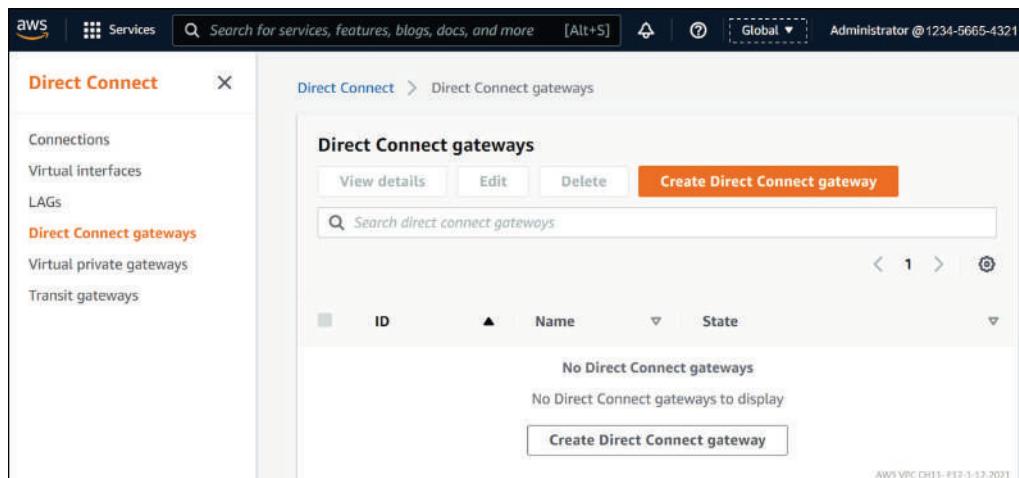


Figure 11-12: AWS Direct Connect – Create Direct Connect Gateway: Step1.

Name the connection and select the Amazon side BGP ASN. We are using the AWS default ASN 64512. Then, click the *Create Direct connect gateway* button.

Direct Connect > Direct Connect gateways > Create

Create Direct Connect gateway

A Direct Connect gateway allows you to use your Direct Connect connections to access your VPCs in remote AWS regions. [Learn more](#)

Direct Connect gateway settings

Name
A name to help you identify the new Direct Connect gateway.

Name must contain no more than 100 characters. Valid characters are a-z, 0-9, and – (hyphen)

Amazon side ASN
The Autonomous System Number for the new Direct Connect gateway.

Valid ranges are 64512 - 65534 and 4200000000 - 4294967294.

AWS VPC CH11- F13-1-12-2021

Cancel **Create Direct Connect gateway**

Figure 11-13: AWS Direct Connect – Create Direct Connect Gateway: Step2.

You will be forwarded to the Direct Connect gateway window, where our DXGW is now listed.

Direct Connect > Direct Connect gateways

Direct Connect gateways (1)

<input type="checkbox"/>	ID	Name	State
<input type="checkbox"/>	93089d56-4213-469f-bf9f-153f9f1532df	NWKT-DirectConnectGW	available

AWS VPC CH11- F14-1-12-2021

Figure 11-14: AWS Direct Connect – Create Direct Connect Gateway: Verification.

Example 11-4 shows the AWS CLI response to command `aws directconnect describe-direct-connect-gateways`. You can get the same information also from the Direct Connect gateway window (figure 11-14) by selecting the DXGW and then clicking the View details button.

```
aws directconnect describe-direct-connect-gateways
{
    "directConnectGateways": [
        {
            "directConnectGatewayId": "93089d56-4213-469f-bf9f-153f9f1532df",
            "directConnectGatewayName": "NWKT-DirectConnectGW",
            "amazonSideAsn": 64512,
            "ownerAccount": "123456654321",
            "directConnectGatewayState": "available"
        }
    ]
}
```

Example 11-4: AWS CLI: DXGW Verification.

Create Transit Virtual Interface

Next, we create a Transit Virtual Interface (VIF). The process launches AWS Logical Device (ALD) TCSH-1a2b3c4d on AWS router EqLD5-xyz. It is the virtual router to which we are using eBGP peering with the customer router CE-1. We extend the subnet 10.0.0.0/30 (Broadcast Domain) from TCSH-1a2b3c4d to CE-1 using the VLAN-Id 1077 over dxcon-azg8gbl. Then bind TSCH-1a2b3c4d to DXGW using dxvif-xy84cda.

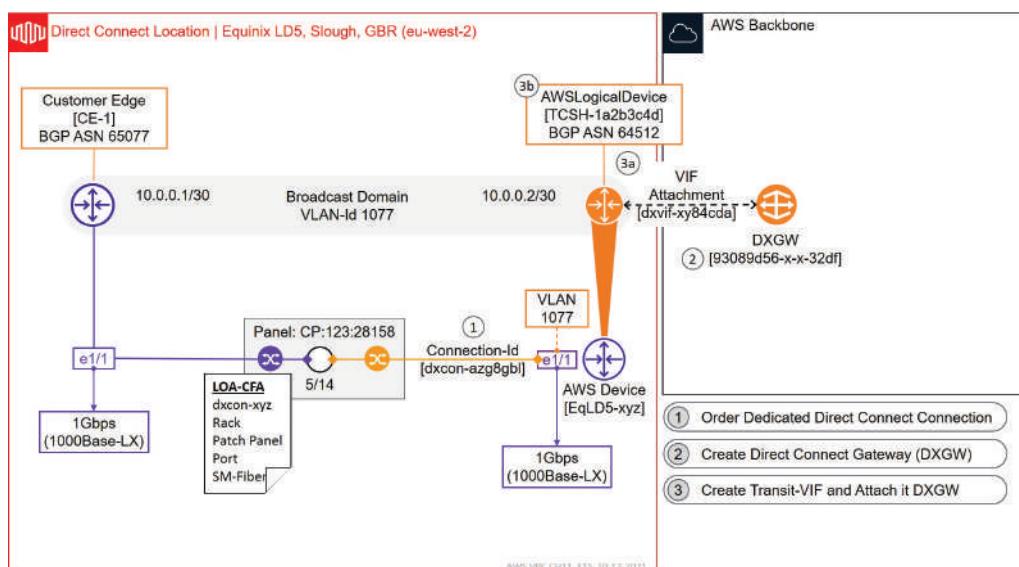


Figure 11-15: AWS Direct Connect – Attach DXC-GW to DXC Using Transit VIF.

Start by selecting the *Virtual interface* option from the Direct Connect section. Next, click the *Create virtual interface* button.

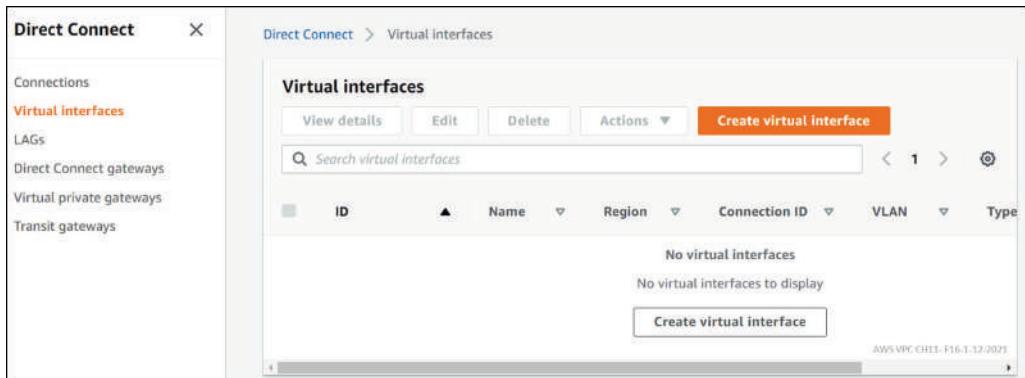


Figure 11-16: AWS Direct Connect – Create Transit VIF: Step-1.

Select the *Transit* option from the *Create virtual interface* window. Note, figures from 11-17 to 11-20 are part of the same view.

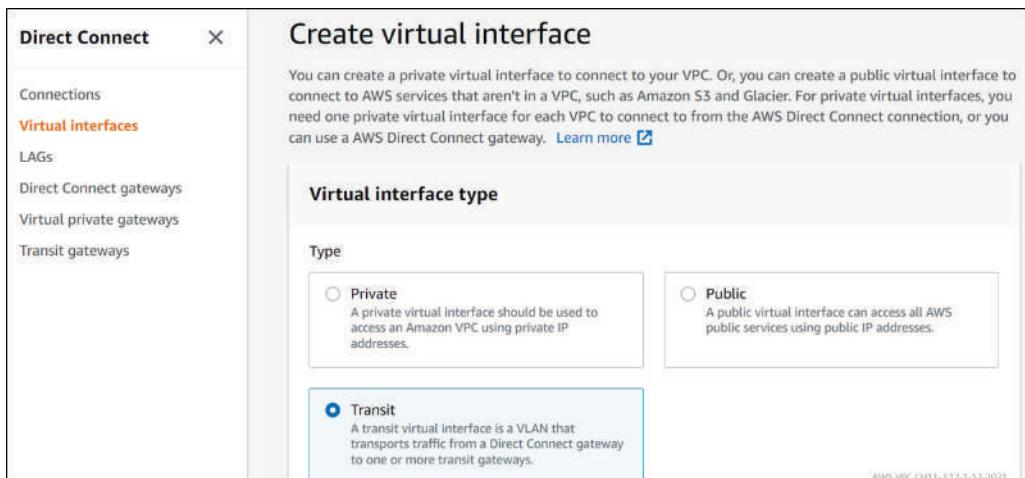


Figure 11-17: AWS Direct Connect – Create Transit VIF: Step-2.

Scroll down to *Transit virtual interface settings* section. Fill in the *Virtual interface name* field. We haven't created a Dedicated Direct Connect connection, so the *Connection* drop-down menu is empty. We own the VIF, so we choose the *My AWS account* option from the *Virtual interface owner* section.

Transit virtual interface settings

Virtual interface name
A name to help you identify the new virtual interface.

NWKT-Transit-VIF

Name must contain no more than 100 characters. Valid characters are a-z, 0-9, and – (hyphen)

Connection
The physical connection on which the new virtual interface will be provisioned.

(dropdown menu)

⚠ Required

Virtual interface owner
The account that will own the virtual interface.

My AWS account
 Another AWS account

AWS VPC CH11- F18-1-12-2021

Figure 11-18: AWS Direct Connect – Create Transit VIF: Step-3.

Select the *Direct Connect Gateway* option from the *Gateway type* section. Open the Direct Connect gateway drop-down menu and select our DXGW NWKT-DirectConnectGW. Then, choose the VLAN Id and customer side BGP ASN. I have used VLAN Id 1077 and BGP ASN 65077 in this example.

Gateway type
Gateway type for this virtual interface.

Direct Connect Gateway - recommended
Allows connections to multiple VPCs and regions

Virtual Private Gateway
Allows connections to a single VPC in the same region

Direct Connect gateway
The Direct Connect gateway to which the new virtual interface will be attached.

NWKT-DirectConnectGW

VLAN
The Virtual Local Area Network number for the new virtual interface

1077

Valid ranges are 1 - 4094

BGP ASN
The Border Gateway Protocol Autonomous System Number of your on-premises router for the new virtual interface.

65077

Valid ranges are 1 - 2147483647.

AWS VPC CH11- F19-1-12-2021

Figure 11-19: AWS Direct Connect – Create Transit VIF: Step-4.

You can set the BGP peering addresses in the *Additional settings* section. We are using the default subnet 10.0.0.0/30. We are attaching the IP address 10.0.0.1 to the customer router and the IP address 10.0.0.2 to the AWS router. We are using these addresses for eBGP peering. You can use authentication for BGP messages if your security policy requires that. Besides, you can enable Jumbo MTU (Maximum Transfer Units in Bytes). To proceed, click the *Create virtual interface* button.

▼ Additional settings

Address family - optional
Determines whether the virtual interface is created with an IPV4 or IPV6 peering.

IPV4
 IPV6

Your router peer ip - optional
The BGP peer IP configured on your endpoint
10.0.0.1/30

Amazon router peer ip - optional
The BGP peer IP configured on the AWS endpoint.
10.0.0.2/30

BGP authentication key - optional
The password that will be used to authenticate the BGP session.
my-secret-password

Jumbo MTU (MTU size 9001) - optional
Allow MTU size of 9001 on virtual interface.
 Enabled

Tags
Specified tags to help identify a AWS Direct Connect resource.
No tags associated with the resource
Add tag

AWS VPC CH11- F20-1-12-2021

Cancel **Create virtual interface**

Figure 11-20: AWS Direct Connect – Create Transit VIF: Step-4.

We couldn't finalize the Transit VIF configuration because we haven't created a real Dedicated Direct Connect connection. However, figure 11-21 shows how you can download AWS auto-generated customer device configuration file. First, select the Virtual interface listed in the Virtual interface view (we don't have any). The *View Details* button is activated after selection. Click it in order to move forward. Select the *Download router configuration* option. Select customer router's vendor, device model, and software version. Then choose the *Download* option.

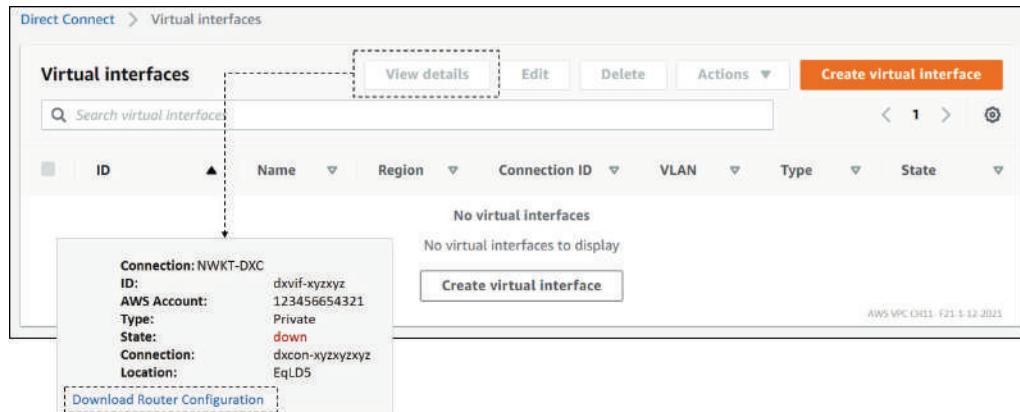


Figure 11-21: AWS Direct Connect – Create Transit VIF: Step-5.

Example 11-5 shows how we can verify the Transit Virtual Interface attachment from the AWS CLI. It shows the owner account id, DXGW Id, AWS region, attachment state (still attaching), and the VIF Id. The prefix of VIF is dxvif-.

```
aws directconnect describe-direct-connect-gateway-attachments --direct-connect-gateway-id 93089d56-4213-469f-bf9f-153f9f1532df

{
  "directConnectGatewayAttachments": [
    {
      "virtualInterfaceOwnerAccount": "123456654321",
      "directConnectGatewayId": "93089d56-4213-469f-bf9f-153f9f1532df",
      "virtualInterfaceRegion": "eu-west-2",
      "attachmentState": "attaching",
      "virtualInterfaceId": "dxvif-xy84cda"
    }
  ],
  "nextToken": "token snippet for brevity"
}
```

Example 11-5: AWS CLI: DXC-GW VIF Attachment Verification.

Configure BGP Peering Between Routers

When we have downloaded the router configuration file, we can create an eBGP peering configuration into CE-1.

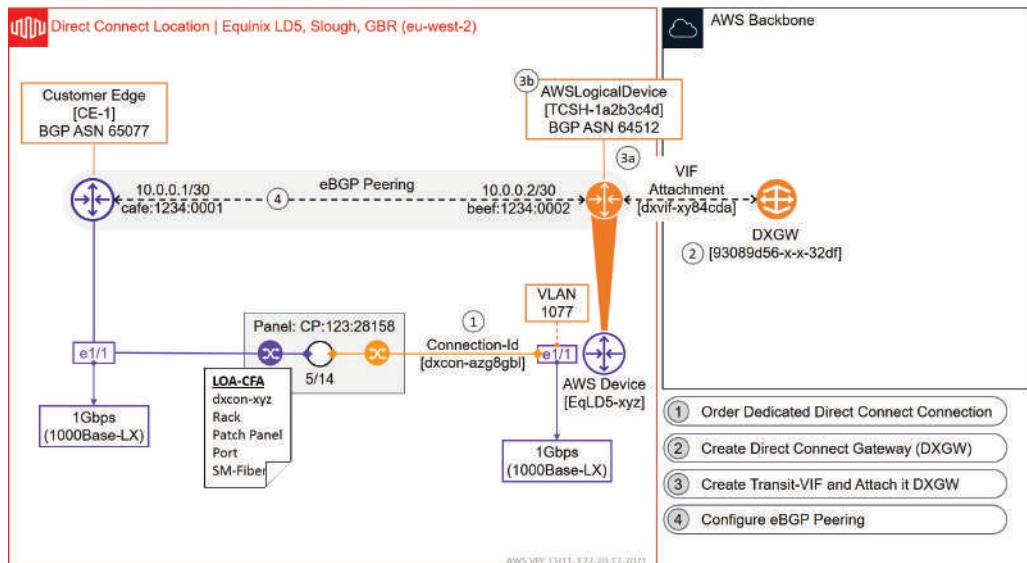


Figure 11-22: AWS Direct Connect – Configure eBGP Peering.

The BGP configuration depends on a device model and operating system. Example 11-6 shows an example BGP configuration.

```
router bgp 65077
neighbor 10.0.0.2 remote-as 64512
neighbor 10.0.0.2 activate
neighbor 10.0.0.2 timers 10 30 30
address-family ipv4 unicast
    neighbor 10.0.0.2 remote-as 64512
    neighbor 10.0.0.2 timers 10 30 30
    neighbor 10.0.0.2 default-originate
    neighbor 10.0.0.2 activate
    neighbor 10.0.0.2 soft-reconfiguration inbound
    network 172.16.0.0 mask 255.255.255.0
```

Example 11-6: CE-1 eBGP Configuration Example.

Associate TGW with Direct Connect GW

As the last step, we create a Transit Gateway association with DXGW. Besides, we define what subnets DXGW propagates to ALD TCSH-1a2b3c4d over dxvif-xy84cda, which in turn sends BGP Update message to CE-1. In our example, we advertise the subnet 10.10.0.0/24.

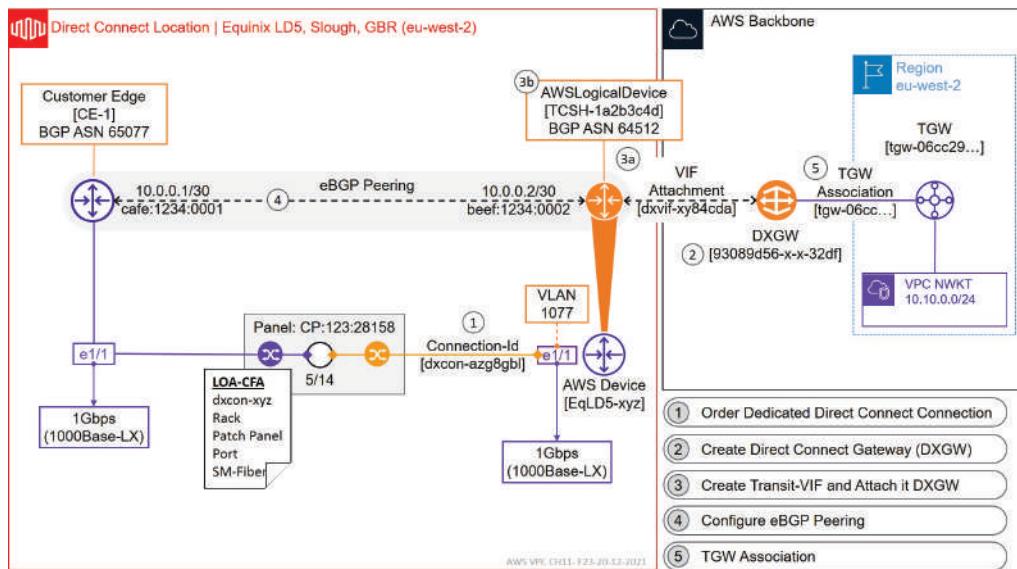


Figure 11-23: AWS Direct Connect – Association TGW with DXC-GW.

Navigate to *Direct Connect gateways* view. Then select the DXGW to which you want to associate the TGW. We only have NWKT-DirectConnectGW on the list, so we choose it. Next, click the *View details* button.

ID	Name	State
93089d56-4213-469f-bf9f-153f9f1532df	NWKT-DirectConnectGW	available

Figure 11-24: Association TGW with DXC-GW: Step-1.

Figure 11-25 shows the information related to our DXGW. The Virtual interface attachment shows that we actually don't have any Virtual Interface attachment because we can't create one without a real Direct Connect connection. However, we can configure the Transit Gateway association. Choose the Gateway association tab.

The screenshot displays the AWS Direct Connect Gateway (DXGW) configuration page. At the top, there is a header with the ID '93089D56-4213-469F-BF9F-153F9F1532DF' and two buttons: 'Edit' and 'Delete'. Below the header, there is a section titled 'General configuration' containing the following details:

ID	AWS account	Amazon side ASN
93089d56-4213-469f-bf9f-153f9f1532df	123456654321	64512
Name	State	
NWKT-DirectConnectGW	<input checked="" type="checkbox"/> available	

Below the general configuration, there are two tabs: 'Virtual interface attachments' (selected) and 'Gateway associations'. The 'Virtual interface attachments' tab shows a table with the following columns: ID, Region, AWS account, and State. The table currently displays the message 'No attached virtual interfaces'. At the bottom of this section is a 'Create virtual interface' button. The 'Gateway associations' tab is also present but not selected.

Figure 11-25: Association TGW with DXC-GW: Step-2.

Click the *Associate gateway* button on the *Gateway association* tab.

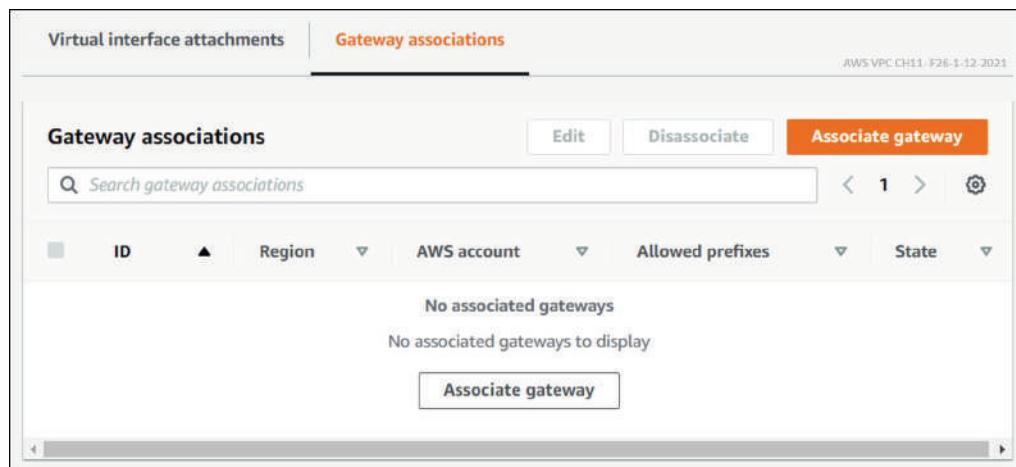


Figure 11-26: Association TGW with DXC-GW: Step-3.

Select the TGW from the Gateway section's drop-down menu, which you associate with the DXGW. Our example TGW is NWKT-TGW-LONDON. You can also define the allowed prefixes which DXGW propagates to VIF. Click the Associate gateway button to implement the configuration.

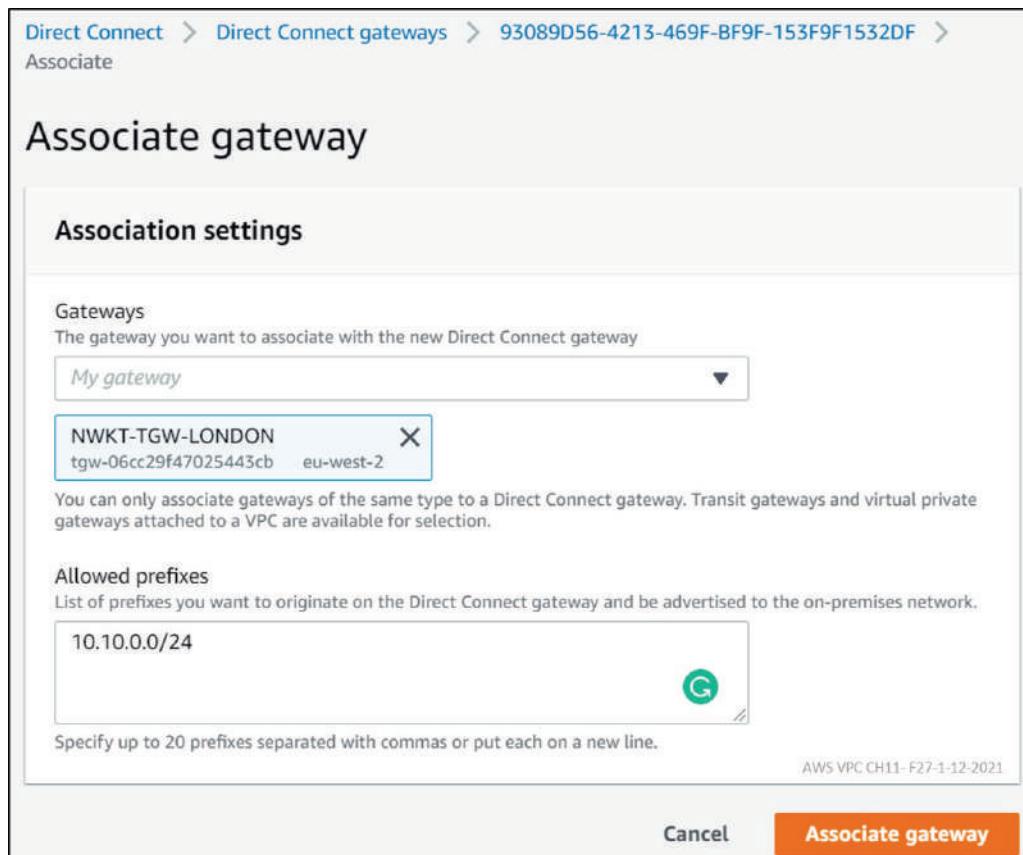


Figure 11-27: Association TGW with DXC-GW: Step-4.

Figure 11-28 verifies the successful association. Note that you can associate three TGWs with DXGW.

Virtual interface attachments		Gateway associations				AWS VPC CH11-F27-1-12-2021	
		Gateway associations (1)					
		<input type="button" value="Edit"/> <input type="button" value="Disassociate"/> <input type="button" value="Associate gateway"/>					
<input type="text"/> Search gateway associations							
	<input checked="" type="checkbox"/>	ID	Region	AWS account	Allowed prefixes	State	
	<input checked="" type="checkbox"/>	tgw-06cc29f47025443cb	eu-west-2	123456654321	10.10.0.0/24	<input checked="" type="checkbox"/> associated	

Figure 11-28: Association TGW with DXC-GW – Verification.

Example below shows the synthetic output of command `aws directconnect describe-virtual-interfaces --connection-id dxcon-xyz85cda` after we have done all the configurations.

```
{
  "virtualInterfaces": [
    {
      "addressFamily": "IPv4",
      "amazonAddress": "10.0.0.2",
      "amazonSideAsn": 64512,
      "asn": 65077,
      "authKey": " ",
      "awsDeviceV2": " ",
      "awsLogicalDeviceId": " ",
      "bgpPeers": [
        {
          "addressFamily": " IPv4",
          "amazonAddress": "10.0.0.2/30",
          "asn": 64512,
          "authKey": " ",
          "awsDeviceV2": " ",
          "awsLogicalDeviceId": "TCSH-1a2b3c4d",
          "bgpPeerId": "10.0.0.1",
          "bgpPeerState": " ",
          "bgpStatus": " ",
          "customerAddress": "10.0.0.0/1"
        }
      ],
      "connectionId": "dxcon-xyz85cda",
      "customerAddress": "10.0.0.1/30",
      "customerRouterConfig": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<logical_connection id=\"dxvif-xy84cda\"\>\n  <vlan>1077</vlan>\n  <customer_address>10.0.0.1/30</customer_address>\n  <amazon_address>10.0.0.2/30</amazon_address>\n  <bgp_asn>65077</bgp_asn>\n  <bgp_auth_key> </bgp_auth_key>\n  <amazon_bgp_asn>64512</amazon_bgp_asn>\n<connection_type>tranist</connection_type>\n</logical_connection>\n",
      "directConnectGatewayId": " 93089d56...",
      "jumboFrameCapable": boolean,
      "location": "",
      "mtu": ,
      "ownerAccount": "123456654321",
      "region": "eu-west-2",
      "routeFilterPrefixes": [
        {
          "cidr": "10.10.0.0/24"
        }
      ],
      "siteLinkEnabled": ,
      "tags": [
        {
          "key": " ",
          "value": " "
        }
      ]
    }
  ]
}
```

```

        ],
        "virtualGatewayId": "",
        "virtualInterfaceId": " dxvif-xy84cda ",
        "virtualInterfaceName": "NWKT-Tranist-VIF",
        "virtualInterfaceState": "",
        "virtualInterfaceType": "Transit",
        "vlan": 1077
    }
]
}

```

Example 11-: Example Output of `Describe-Virtual-Interface`.

Direct Connect Gateway – Traffic Flow

Direct Connect Gateway doesn't route traffic between gateways associated with the same DXGW. This rule applies to both Transit Gateway (TGW) and Virtual Private Gateway (VGW). In figure 11-29, we have one TGW in the London region and the other one in the Stockholm region. If we want to allow traffic between VPCs on different AWS regions, we either have to use TGW Peering or Inter-Region VPC peering. IP traffic between VPCs that are attached to the same TGW is allowed. Besides, traffic between customer CE-1 to all three VPCs is permitted.

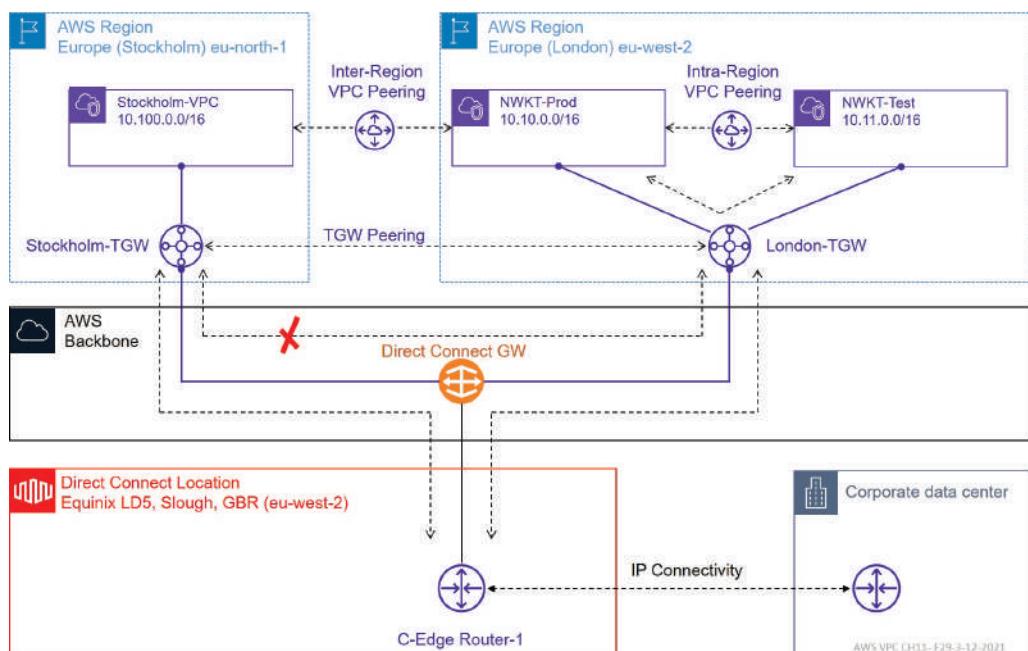


Figure 11-29: AWS Dedicated Direct Connect Traffic Policy Model.

Chapter 12: Hosted Direct Connect

Introduction

Customers can order *AWS Hosted Direct Connection (H-DXC)* from AWS Direct Connect Partner (AWS-DCP) that has a presence in AWS Direct Connect Location (AWS-DCL). The AWS-DCPs, which offer H-DXC service, have existing cross-connections from their Cloud Edge Fabric to AWS router. These connections are either 1Gbps or 10Gbps. Some AWS-DCPs also have a 100Gbps connection. H-DXC is useful for customers who do not have network devices in the AWS-DCL. They can terminate their Layer 2 WAN circuit (e.g. Dark Fiber, Metro Ethernet, MPLS VPLS, EoMPLS) from on-prem Datacenter to Direct Connect Partner's Network/Cloud Edge Fabric. In this model, customers can order an AWS router connection straight from the AWS-DCP, using their AWS cross-connection. The benefit of using Hosted Direct Connect is that it supports bandwidths from 50Mbps to 100Gbps. At the time of writing, supported data transfer rates are 50 Mbps, 100 Mbps, 200 Mbps, 300 Mbps, 400 Mbps, 500 Mbps, 1 Gbps, 2 Gbps, 5 Gbps, and 10 Gbps. Customers whose device is in AWS-DCL can also order an H-DXC connection from the AWS-DCP if they need other transfer speed than standard AWS Dedicated Direct Connect connections (1Gbps, 10Gbps, or 100Gbps).

The focus of this chapter is to explain how we can extend Layer 2 segment over AWS-DCP's Cloud Edge fabric by using BGP EVPN Control Plane with VXLAN Data Plane. AWS-DCPs do not publish detailed technical documentation about their Cloud Edge solutions. However, our example AWS-DCP Equinix has published video series, *Packet Flow Part 1 & 2: Network Edge*. In these videos, Equinix's Senior Product Manager Brad Gregory explains the Control Plane and Data Plane operation in their Network Edge Fabric. Based on these videos, we know that they use EVPN/VXLAN technologies for extending Layer 2 segments over Network Edge Fabric.

If the VXLAN is not relevant for you, you can skip this chapter and move to the next chapter.

Network Edge

Figure 12-1 illustrates our example Hosted Direct Connect solution. We are using Equinix's LD5 Datacenter as our Direct Connect Location. Equinix is also our Hosted Direct Connect Partner. Customer's edge device CE-1 is in on-prem Datacenter from where they have Layer 2 circuit to Equinix LD5 Datacenter. The connection is terminated to Edge-1's interface e1/1. The cross-connection between the Edge-2 and AWS router is 10Gbps over single-mode fiber. In real life, the Equinix Network Edge Fabric is connected to the Equinix ECX device, which is connected to AWS Router. Network Edge fabric uses BGP EVPN for MAC address advertisement with ECX devices. To keep things simple, I excluded ECX from figure 12-1. The purpose of the Network Edge, in our example, is to extend the Layer 2 broadcast domain between the Customer router and AWS Logical Device TCSH-1a2b3c4d running on AWS router.

BGP EVPN Control Plane Operation

Our purpose is to establish eBGP peering between the customer router CE-1 and TCSH-1a2b3c4d using shared network segment 10.0.0.0/30. BGP peering session uses TCP as a transport protocol. CE-1 starts the BGP peering process by initiating a TCP connection with ALD TCSH-1a2b3c4d. Because they both are in the same Layer 2 segment, CE-1 first has to resolve the MAC address associated with the IP 10.0.0.2 (ALD TCSH-1a2b3c4d eBGP peering IP). It sends an ARP request (L2 Broadcast), where it asks who owns the IP address 10.0.0.2. Edge-1 receives the ARP message from the interface e1/1. The ingress ARP message triggers the BGP EVPN Control Plane process where Edge-1 advertises CE-1 MAC/IP binding information to Edge-2. Figure 12-1 illustrates the overall process.

Step 1: *MAC/IP learning process*

Edge-1 receives the ARP Request message from the interface e1/1. It learns MAC and IP addresses of CE-1 (cafe:1234:0001/10.0.0.1) from the ingress ARP message. It installs the MAC address into VLAN 1077 MAC address table and IP/MAC binding information into the ARP table.

Step 2: *Layer 2 Routing Information Base (L2RIB)*

Edge-1 programs the MAC and IP addresses to the Layer 2 Routing Information Table (L2RIB) associated with Virtual Network 10000.

Step 3: Exporting MAC/IP binding information to the BGP process

Edge-1 exports the MAC/IP binding information to the BGP process and installs information into the local BGP table.

Step 4: BGP Update Message from Edge-1 to Edge-2

Edge-1 sends a BGP Update message to Edge-2. This message describes a Network Layer Reachability Information (NLRI) MAC/IP addresses of CE-1.

Step 5: Importing MAC/IP Binding from BGP Table into L2RIB

Edge-2 imports MAC/IP address information from the BGP table to L2RIB with associated Next-Hop IP address (Edge-1).

Step 6: Updating MAC Address Table and ARP Cache

Next, Edge-2 installs the MAC address information into VLAN 1077 MAC address table. Besides, it programs the MAC/IP Binding information into the local ARP table. This way, Edge-2 can reply to ARP requests from ALD TCSH-1a2b3c4d without forwarding ARP messages to Edge-1.

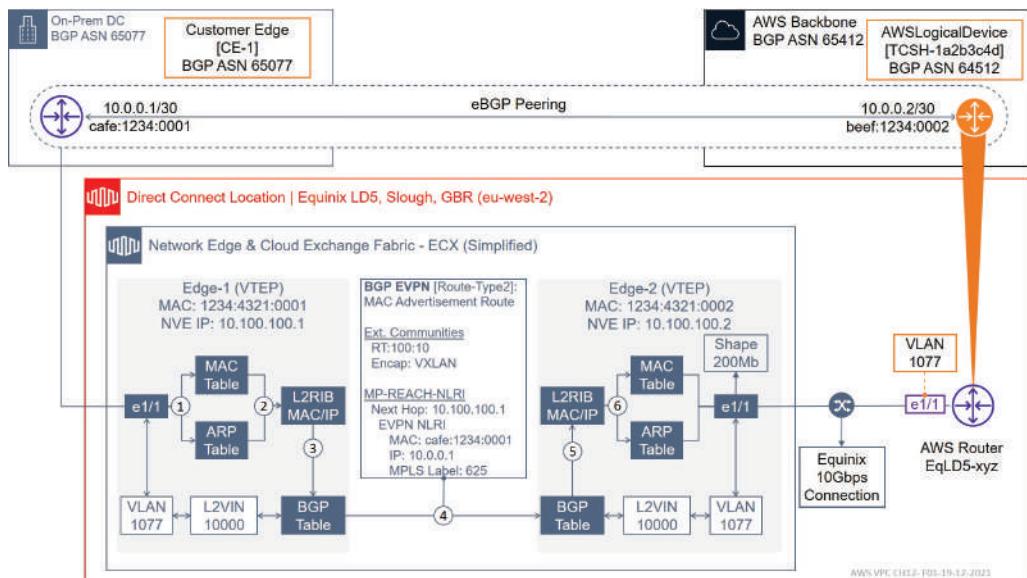


Figure 12-1: AWS Hosted Direct Connect Connection – EVPN Control Plane.

The MAC address table of Edge-1 in figure 12-2 shows that MAC address cafe:1234:0001 belongs to VLAN 1077 and that it has been learned dynamically (Data Plane learning) from the interface e1/1 24 seconds ago. VLAN 1077 is associated with Virtual Network (VN) 10000. ARP table includes MAC/IP binding information and the Layer 3 interface (Routing Interface) of VLAN 1077. MAC

address from the MAC table and IP address from the ARP table are programmed into the L2RIB. In addition to MAC/IP addresses mapping information, L2RIB describes the VN-Id (10000), how the information is produced into it (locally), and the Next-Hop information (Local). L2RIB information is then exported to the BGP process. Edge-1 installs the MAC/IP address NLRI with associated BGP Path Attributes into the local BGP table (not shown in figure). Then it generates a BGP Update message. BGP Update message includes the set of Path Attributes. Edge-1 attaches the Route Target 100:10 to all BGP update messages, which carries NLRI related to MAC/IP mapping of hosts in VLAN 10. The other Extended Community, Encapsulation Type, is used to inform the receiving side that Ethernet frames towards the advertised MAC address have to send within VXLAN encapsulation (Outer Ethernet/IP headers, UDP header, and VXLAN header). The BGP Path Attribute MP_REACH_NLRI carries the EVPN NLRI (MAC/IP information) and related Next-Hop address. BGP EVPN has several Route Types. MAC/IP address information is advertised as EVPN Route Type 2 (MAC Advertisement Route). The Route Distinguisher (RD) value is used as a prefix for MAC Advertisement addresses, and it is a part of the address. In our example, RD identifies the originating device using its IP address. The value 32777 after IP address 10.100.100.1 is derived from the VLAN 10. The base value of this field is 32767. Subtracting the base value 32767 from 32777, we get $10 = \text{VLAN Id}$. As said, each MAC/IP address NLRI about hosts attached locally to VLAN 10 will get the same RD prefix. This way, we can use overlapping MAC and IP addresses in BGP EVPN Fabric. In addition to MAC and IP address information, our example EVPN NLRI has MPLS Label Stack 1: 625 that identifies the VN 10000. To understand how 625 is turned to 10000, we need to convert the decimal number 625 to HEX. The result is 271 (12 bits). If we capture a BGP Update message from the wire, we can see that the length of the MPLS Label value carried in EVPN NLRI is actually 24 bits, and 625 is encoded as 00 27 10 (there we have our missing 12 bit). When we convert this HEX 00 27 10 back to the decimal format, we get 10000. This math is out of the scope of AWS Network Fundamentals. However, I added it because I tend to forget the math behind MPLS Label = VNI.

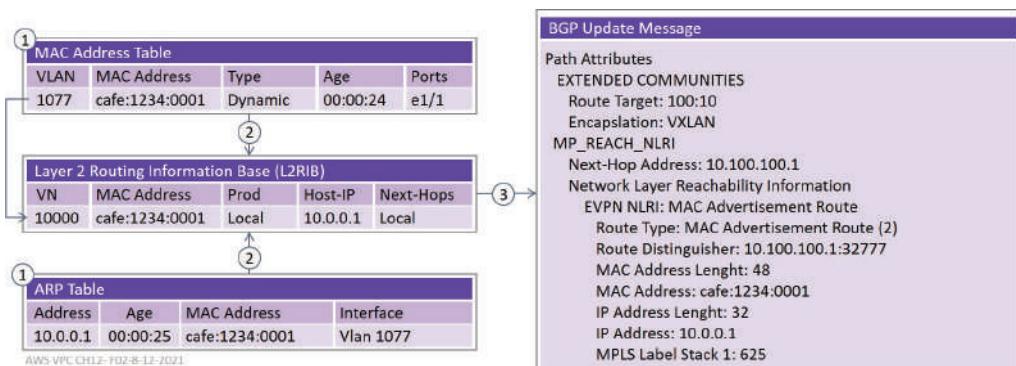
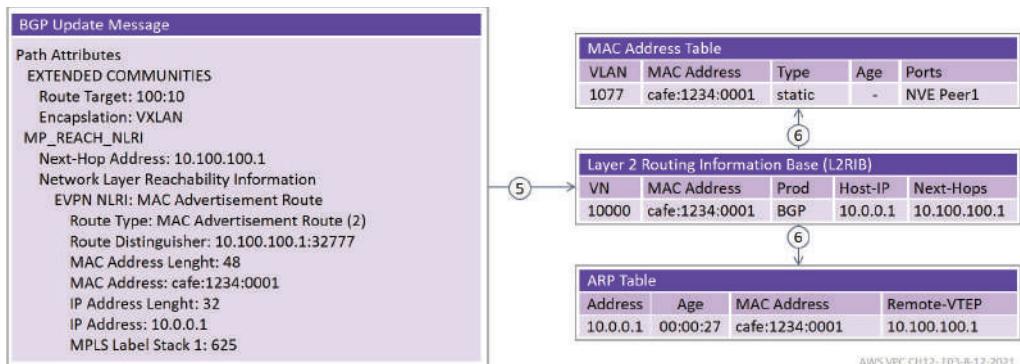


Figure 12-2: Control Plane Processes of Edge-1.

When Edge-2 receives the BGP Update message originated by Edge-1, it first confirms that it has a route to Next-Hop address 10.100.100.1. The Next-Hop IP address is Edge-1's NVE (Network Virtualization Edge) interface address. Edge-2 uses 10.100.100.1 as a destination IP address in the tunnel IP header for data packets to hosts connected to Edge-1. Next, Edge-2 imports NLRI information to its local BGP table. The import process is based on the Route Target Extended Community 100:10, which is associated with the VN 10000. BGP Update message and the local BGP table hold almost identical information. That's why the BGP table is not shown in figure 12-3. When Edge-2 has updated the local BGP table, it installs the routing information into L2RIB. Then it updates VLAN 10 MAC address table and the ARP Table.



AWS VPC CH12 - F03-B-12-2021

Figure 12-3: Control Plane Processes of Edge-1.

When the MAC Address table and ARP table are updated, Edge-2 is able forward Ethernet frames from ALD TCSH-1a2b3c4d to CE-1. It can also reply to the ARP Request message on behalf of CE-1. It means that ARP messages are not forwarded across the fabric. The same BGP EVPN process happens in the opposite direction when Edge-2 receives the first IP packet from the ALD TCSH-1a2b3c4d.

VXLAN Data Plane

Figure 12-4 illustrates the Data Plane operation. In this example, we assume that both CE-1 and ALD TCSH-1a2b3c4d have sent Gratuitous ARP (GARP) messages after booting up. GARP process has triggered the BGP EVPN Control Plane process where Edge devices have sent EVPN MAC Route advertisements about their locally connected device. That said, both Edge devices have an up-to-date MAC address table, ARP table, and L2RIB.

Before CE-1 can start the three-way handshake process with ALD TCSH-1a2b3c4d, it resolves the MAC address associated with the IP address 10.0.0.2. It sends an ARP Request message to figure out who has 10.0.0.2. When Edge-1 receives the ARP Request message, it consults its ARP table. It has requested information in its ARP table, and it sends an ARP Reply with the MAC/IP binding information to CE-1. The ARP Request/Reply process is not shown in figure 12-4.

Step 1: *Data Packet from C-Edge Router-1 to Edge-1*

After resolving the MAC address, CE-1 sends a TCP SYN message with the destination port 179 (BGP) to 10.0.0.1. It sets the destination MAC address to beef:1234:0002 (MAC associated with IP 10.0.0.1) and forwards the frame to Edge-1.

Step 2: *Data Packet from Edge-1 to Spine-11*

Edge-1 receives the message. It checks the destination MAC address. Because the MAC address does not belong to Edge-1, it does a switching lookup from the MAC Address table. The destination is NVE peer 1, which is Edge-2. Remember that Edge-2 has advertised the MAC Address as EVPN MAC Advertisement route, where the Next-Hop IP address is its NVE interface IP address (VXLAN Tunnel interface aka Network Virtual Edge interface-NVE). Edge-1 uses VXLAN encapsulation when sending data out to the NVE Tunnel interface. Edge-1 sets the VN identifier 10000 into the VXLAN header. The destination UDP port is 4789, which tells to receiving device that the following header is the VXLAN header. The destination IP address in the outer IP header is the NVE interface address of Edge-2. The destination MAC address in the outer Ethernet header is always the MAC of the next router. In our example, it belongs to Spine-11.

Step 3: *Data Packet from Spine-11 to Edge-2*

When Spine-11 receives the packet, it notices that the destination MAC address is one of its MAC addresses. Next, it checks the destination IP address from the outer IP header. Because it is not one of its IP addresses, it does a routing lookup from its L3 RIB. Before forwarding the packet to Edge-2, Spine-11 re-writes the Ethernet header's MAC addresses and calculates a new FCS value (Frame Check Sequence for error detection). After that, it forwards the packet to Edge-2.

Step 4: Data Packet from Edge-2 to Direct Connect Gateway

Edge-2 notices that the destination MAC address is its MAC address, and it removes the Ethernet header. The destination IP address in the outer IP header also belongs to Edge-2, so it also removes the outer IP header. Edge-2 knows that the following header is the VXLAN header based on the destination UDP port 4789. Then it removes the UDP header and checks the VNI from the VXLAN header. VNI 10000 is associated with VLAN 1077. Edge-2 removes the VXLAN header and makes a switching decision based on VLAN 1077 MAC Address table.

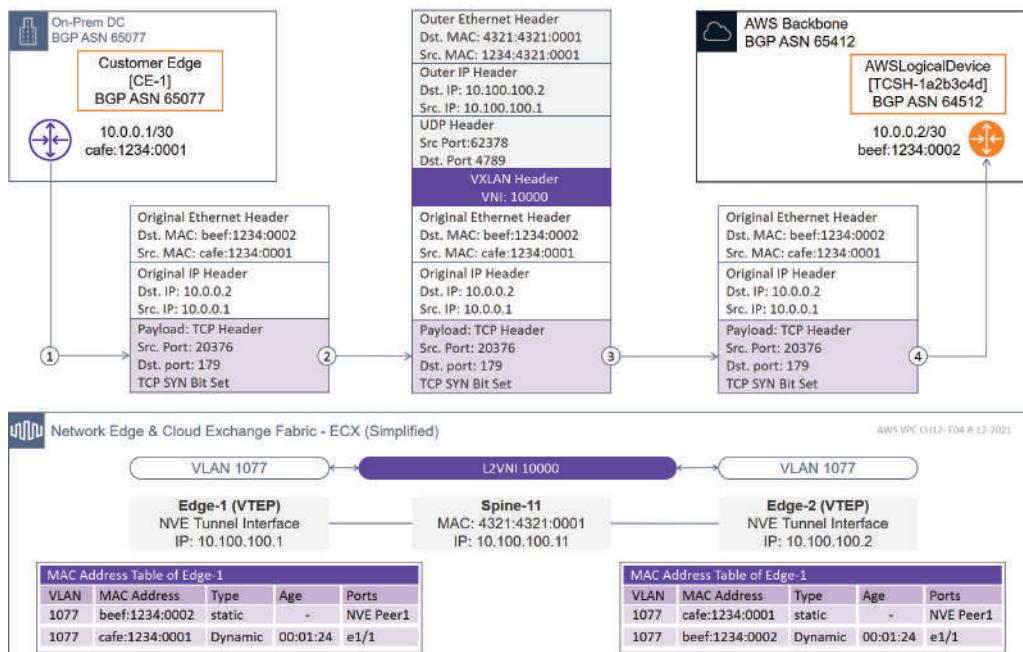


Figure 12-4: AWS Hosted Direct Connect Connection – VXLAN Data Plane.

We just scratched the surface of the BGP EVPN/VXLAN fabric. If you want to learn more about the VXLAN solution, you may want to read my Virtual Extensible LAN – VXLAN book.

Chapter 13: Direct Connect BGP Policy

Introduction

This section explains how we can affect to egress path selection process on AWS Direct Connect Gateway (DXGW). DXGW are AWS-managed services, so we can't change its configuration. However, we can use BGP built-in tools (BGP route aggregation, BGP AS-Path Prepending, and BGP Communities) for affecting to egress path selection process. Figure 13-1 shows our example setup where we have Customer Edge routers CE1 and CE2 in the on-prem Datacenter. CE-1 has eBGP peering with ALD-1 and CE2 with ALD-2. I haven't included AWS physical routers on either DCL in figure 13-1 for simplicity.

BGP Route Selection Process

Figure 13-1 illustrates a default BGP behavior and route selection processes. We have subnets 172.16.0.0/24 and 172.16.1.0/24 on the on-prem Datacenter. We haven't done any BGP policy configuration at this phase, just a basic BGP configuration (peer IP and AS + network advertisements). CE1 and CE2 advertise these subnets without any BGP Path-Attribute modification. Because subnets 172.16.0.0/24 and 172.16.1.0/24 are local subnets, the BGP AS-Path path attribute list has only the local ASN 65077 in all four BGP Update messages. CE routers set the next-hop IP address to their eBGP peering interface's IP.

When the AWS Logical Devices (ALD) receive BGP Update message from their eBGP peer, they first verify that they have a route to the next-hop IP address carried within the message. After Next-Hop verification, they install the subnet and its path-attribute information into the BGP Adj-RIB-In table. Next, the BGP process imports the NLRI through the BGP Inbound Policy Engine into the Local BGP table (Loc-RIB). After installing NLRI into the BGP Loc-RIB, ALDs start the best path selection process.

When there are two or more entries about the same subnet, the BGP process decides which one is the best in the following order. (1) Longest subnet mask (2) Highest Weight value, (3) Highest Local-Preference value, (4) prefer locally originated prefixes, (5) Shortest AS_Path attribute list, (6) prefer IGP < EGP < Incomplete, (7) lowest MED. In the case of each Path-Attributes are equal, the decision process prefers (8) eBGP over iBGP, (9) the lowest IGP metric to Next-Hop, (10) When both paths are external, select the oldest one, (11), and as the last step prefer the path through the neighbor that has the lowest BGP Route Id (RID). DXGW has learned the subnets 172.16.0.0/24 and 172.16.1.0/24 via VIF attachments dxvif-1234 and dxvif-5678. Let's say that CE-1 has sent BGP Updates first to ALD-1. That is why DXGW installs routes learned via VIF attachment dxvif-1234 into Route Table based (BGP Best Path selection criteria 10).

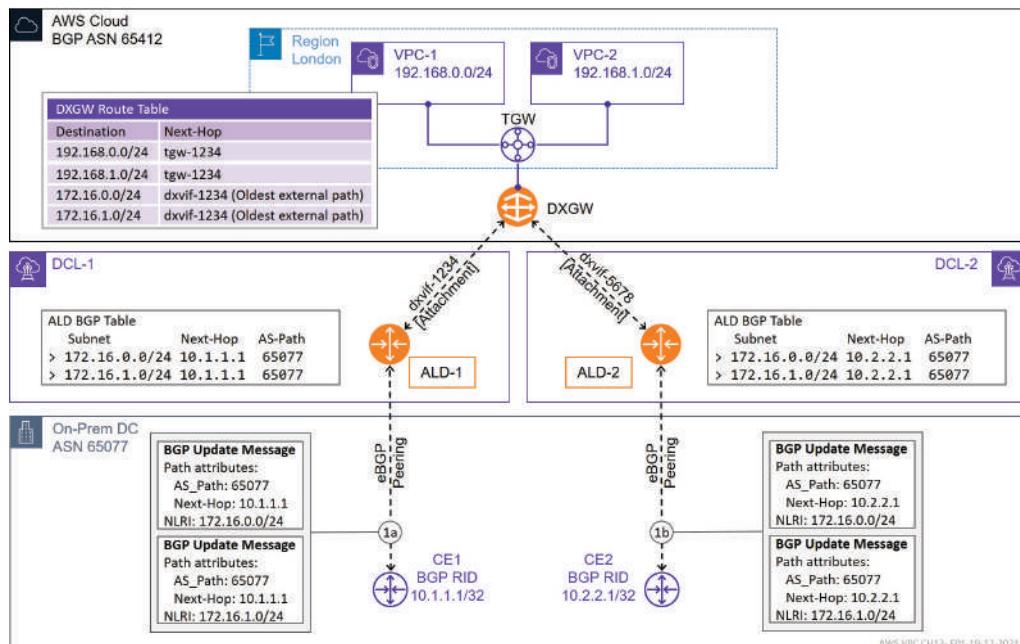


Figure 13-1: DXGW's BGP Egress Policy – Default BGP Process.

DXGW Egress Policy - BGP Summary Route

In this example, we use a BGP Summary Route to ensure that DXGW forwards all traffic from VPCs over the attachment dxvif-1234 to CE1. This policy does not require any BGP configuration changes on CE1, but we create an aggregate route in CE2. Example 13-1 shows how we configure an aggregate on the Cisco IOS device. Note, CE2 advertises both route aggregate and original subnets if we do not use the *summary-only* option.

```
Router bgp 65077
neighbor 10.10.0.2 remote-as 64577
aggregate-address 172.16.0.0 255.255.254.0 summary-only
```

Example 13-1: CE2 BGP Policy – BGP Aggregate Address.

When there are two or more entries about the same subnet, the BGP Figure 13-2 shows that ALD-1 learns both subnets 172.16.0.0/24 and 172.16.1.0/24 from CE1. After route aggregation on CE2, ALD-2 receives the BGP Update about summary-address 172.16.0.0/23 from CE2. DXGW learns routes over VIF attachments. It installs all three subnets into the route table. The summary-address has a shorter subnet mask (/23) than C-class subnets (/24). For this reason, DXGW routes traffic towards subnets 172.16.0.0/24 or 172.16.1.0/24 over VIF attachment dxvif-1234.

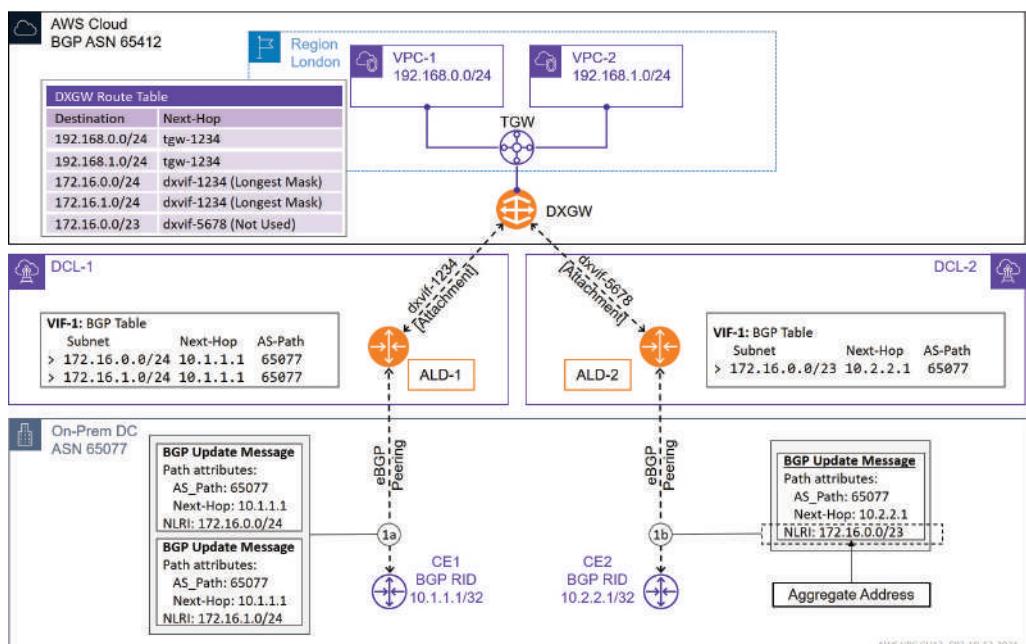


Figure 13-2: DXGW’s BGP Egress Policy – Aggregate Address.

DXGW Egress Policy – BGP AS-Path Prepend

The 5th step in the BGP Path-Selection process prefers the NLRI with the shortest AS-Path. In figure 13-3, we advertise a summary route 17.16.0.0/23 on both CE devices. Consider route aggregation because the AWS Direct Connect BGP device accepts no more than 100 routes from the customer BGP peer. We intend to use the ALD-1 over dxvif-1234 as our primary traffic path from DXGW to on-prem DC. In this example, we prepend the AS-Path list in BGP Update advertised by CE2.

The example below illustrates how we can prepend the AS-Path list in the Cisco IOS device. First, we create an IP prefix-list *ON_PREM_AGGREGATE* that permits the aggregate route 172.16.0.0/23. Next, we configure a route-map *ASW-DCX-Egress-Policy*. This route-map prepends the BGP AS-Path attribute list with one local BGP ASN for the advertised subnet, which matches to aggregate address defined in IP prefix-list *ON_PREM_AGGREGATE*. As the last step, we add the route-map *ASW-DCX-Egress-Policy* under CE2 BGP configuration as outbound policy.

```
ip prefix-list ON_PREM_AGGREGATE seq 10 permit 172.16.0.0/23
!
route-map ASW-DCX-Egress-Policy permit 10
  match ip address prefix-list ON_PREM_AGGREGATE
  set as-path prepend 65077
!
route-map ASW-DCX-Egress-Policy permit 100
!
Router bgp 65077
  neighbor 10.2.2.2 remote-as 64512
  neighbor 10.2.2.2 route-map ASW-DCX-Egress-Policy out
  aggregate-address 172.16.0.0 255.255.254.0 summary-only
```

Example 13-2: Customer Edge BGP Policy – AS-Path Prepending.

The BGP Update sent by CE1 carries NLRI about subnet 172.16.0.0/23 with the shorter AS-Path list. For this reason, DXGW forwards data received from TGW to on-prem DC over VIF attachment dxvif-1234.

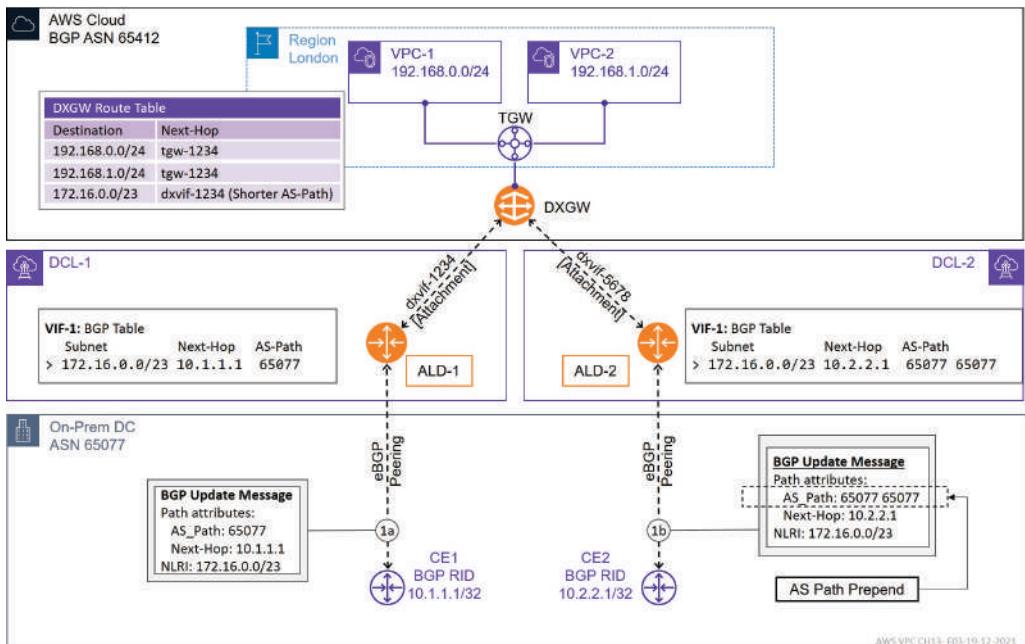


Figure 13-3: DXGW's BGP Egress Policy – AS-Path Prepending.

DXGW Egress Policy - BGP Communities

AWS allows us to use BGP communities for DXGW egress traffic engineering. DXGW uses the following formula for setting BGP community-based Local-Preference (LP). The BGP community 7224:7300 gives the "High" Local Preference, the BGP community 7224:7200 gives the "Medium" Local Preference, and the BGP community 7224:7100 gives "Low" Local Preference. Note that the AWS documentation uses the definition of High/Medium/Low Preference, but in reality, the value of LP is a number, and the higher number is better.

Figure 13-4 shows how CE1 and CE2 set an additional BGP Community Path-Attribute to their BGP Update messages. CE1 uses the community value 7224:7300 (AWS High Preference), and CE2 uses the community value 7224:7100 (AWS Low Preference). ALD-1 and ALD-2 propagate routing information with attached communities to DXGW.

Example 13-3 shows how we can add a BGP community Path-Attribute into the outgoing BGP Update messages about NLRI 172.16.0.0/23. We create an IP prefix-list and route-maps, just like what we did with the AS-Path Prepending example. The difference is that now the route-map action adds BGP Community Path Attribute to outgoing BGP Update messages.

```
ip prefix-list ON_PREM_AGGREGATE seq 10 permit 172.16.0.0/23
!
route-map ASW-DCX-Egress-Policy permit 10
  match ip address prefix-list ON_PREM_AGGREGATE
  set community 7224:7300
!
route-map ASW-DCX-Egress-Policy permit 100
!
Router bgp 65077
  neighbor 10.1.1.2 remote-as 64577
  neighbor 10.1.1.2 send-community
  neighbor 10.1.1.2 route-map ASW-DCX-Egress-Policy out
!
ip bgp-community new-format
```

Example 13-3: Customer Edge BGP Policy – Communities.

Figure 13-4 shows the BGP Inbound Policy of DXGW. It checks the community values from all received routing updates. It gives the High Preference to routes that has a Community value of 7224:7300. In our example, NLRI about the subnet 172.16.0.0/23 advertised by CE1 gets “High” Local Preference while the NLRI originated by CE2 gets Low Preference. DXGW installs the route with the High Preference to the routing table.

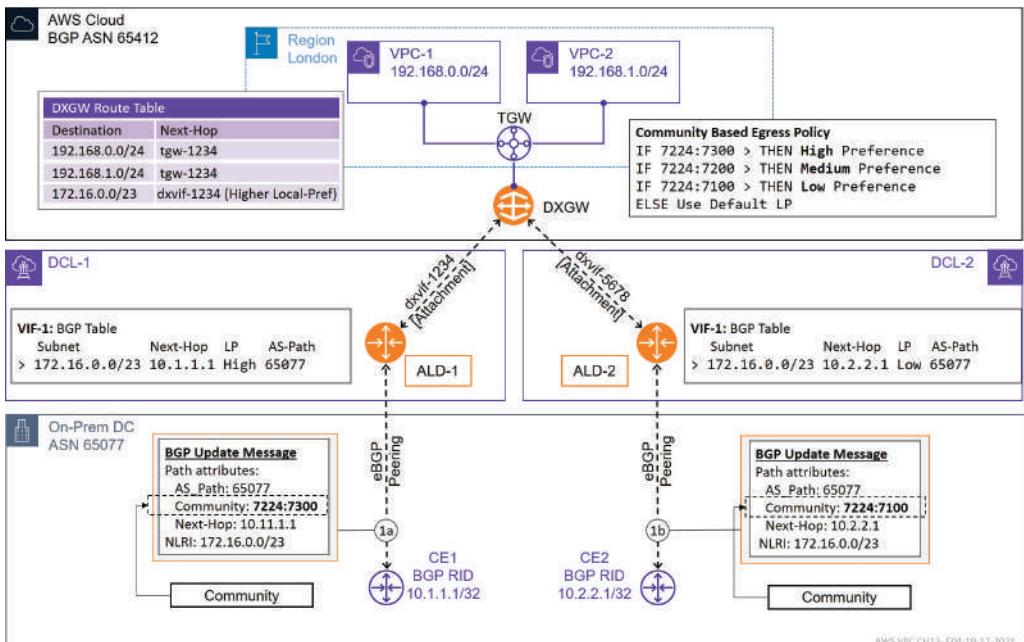


Figure 13-4: DXGW’s BGP Egress Policy – Community Based Local-Preference.

On-Prem DC Egress Policy

In order to avoid asymmetric traffic flows, we also have to create the BGP egress policy in our CE routers in the on-prem Datacenter. There are many ways to do that, but we use BGP Local-Preference Path-Attribute. The example below shows the CE1 BGP configuration. It has an inbound policy *AWS-Ingress-Policy*, which points to route-map *ASW-DCX-Ingress-Policy*. This route-map sets the BGP Local-Preference value 300 for all routes that match the VPC CIDRs 192.168.0.0/24 and 192.168.1.0/24 defined in IP Prefix-List *AWS-London-CIDRs*. CE1 and CE2 are iBGP peers. It means that CE1 advertises the NLRI to CE2 with the new BGP Local Preference value of 300.

```
ip prefix-list AWS-London-CIDRs seq 10 permit 192.168.0.0/24
ip prefix-list AWS-London-CIDRs seq 10 permit 192.168.1.0/24
!
route-map ASW-DCX-Ingress-Policy permit 10
  match ip address prefix-list AWS-London-CIDRs
  set local-preference 300
!
route-map ASW-DCX-Ingress-Policy permit 100
!
Router bgp 65077
  neighbor 10.1.1.1 remote-as 64512
  neighbor 10.1.1.1 route-map ASW-DCX-Ingress-Policy in
  neighbor 10.11.11.2 remote-as 65077
  neighbor 10.11.11.2 next-hop-self
```

Example 13-4: CE1 BGP Ingress Policy – Set Local-Preference to 300.

Example 13-5 shows the CE2 configuration. The only difference among the BGP peer addresses is that now we change the Local-Preference value to 50.

```
ip prefix-list AWS-London-CIDRs seq 10 permit 192.168.0.0/24
ip prefix-list AWS-London-CIDRs seq 10 permit 192.168.1.0/24
!
route-map ASW-DCX-Ingress-Policy permit 10
  match ip address prefix-list AWS-London-CIDRs
  set local-preference 50
!
route-map ASW-DCX-Ingress-Policy permit 100
!
Router bgp 65077
  neighbor 10.2.2.2 remote-as 64512
  neighbor 10.2.2.2 route-map ASW-DCX-Ingress-Policy in
  neighbor 10.11.11.1 remote-as 65077
  neighbor 10.11.11.1 next-hop-self
```

Example 13-5: CE2 BGP Ingress Policy – Set Local-Preference to 300.

When CE2 receives the BGP Update message from CE1, it installs the route to the BGP Adj-RIB-In table and, from where it imports the information into the BGP Loc-RIB. At this phase, it has two BGP entries about subnet 192.168.0.0/24. Next, the BGP path selection process selects the best route. The subnet mask for both subnets is the same /24. Also, the BGP Weight attribute is the same (default value 32768, not shown in the figure). The Local-Preference is now the tiebreaker. Because NLRI advertised by CE1 has a higher Local-Preference value, CE2 selects it as the best route, and CE2 installs it to the routing table. CE2 does not send BGP Update to CE1 because the best route is the route received from CE1. Note that we have configured both CE routers to change the original Next-Hop Path-Attribute (pointing to DXGW) to their iBGP peering addresses. The reason for this is that the Next-Hop IP address carried with BGP Update has to be reachable, otherwise the route is invalid.

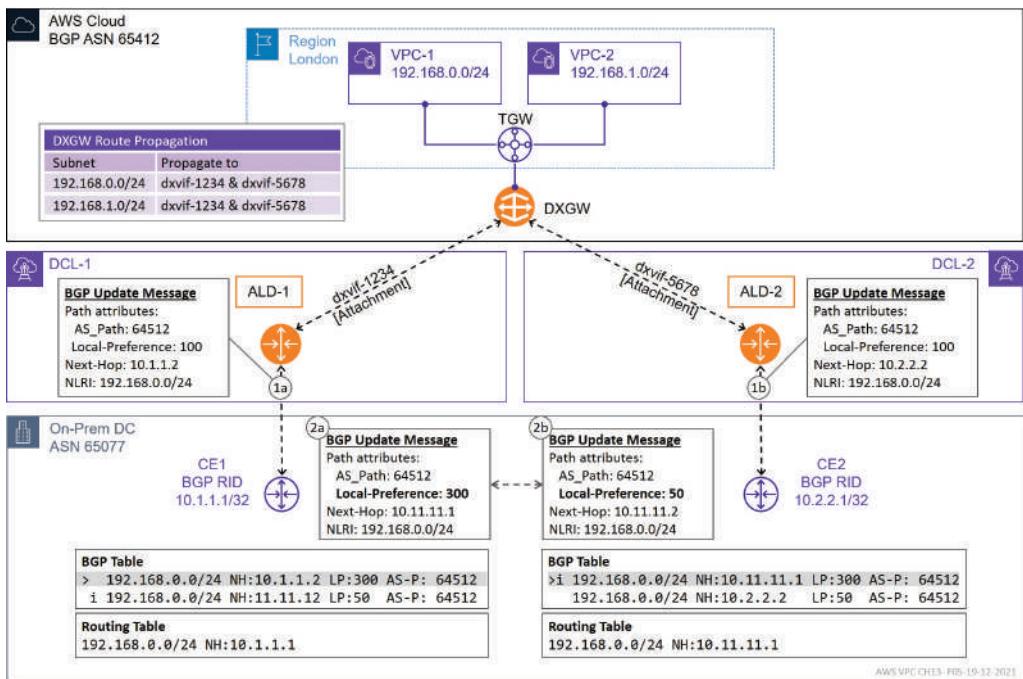


Figure 13-5: AWS Direct Connect – Customer Device in On-Prem DC

Chapter 14: Direct Connect SiteLink

Introduction

Amazon introduced an *AWS Direct Connect SiteLink (DX SiteLink)* solution in AWS re:Invent at the end of 2021. DX SiteLink is a VIF-specific feature, which enables data paths between customers' remote sites over AWS Direct Connect terminated to the same Direct Connect Gateway (DXGW).

SiteLink Disabled on VIFs

Figure 14-1 illustrates the routing scheme when we haven't enabled DX SiteLink on dxvif-1234 and dxvif-5678. Customer edge routers CE-1 on DC-1 and CE-2 on DC-2 have learned the customer VPC-1 CIDR range 192.168.0.0/24 from their eBGP peering ALDs. DXGW propagates subnet 172.16.1.0/24 (DC-1's local subnet) over tgw-1234 association to Transit Gateway (TGW) but not over dxvif-5678 attachment to CE-2. The same routing policy applies to DC-2's subnet 172.16.2.0/24. DXGW only propagates it over the tgw-1234 association to TGW.

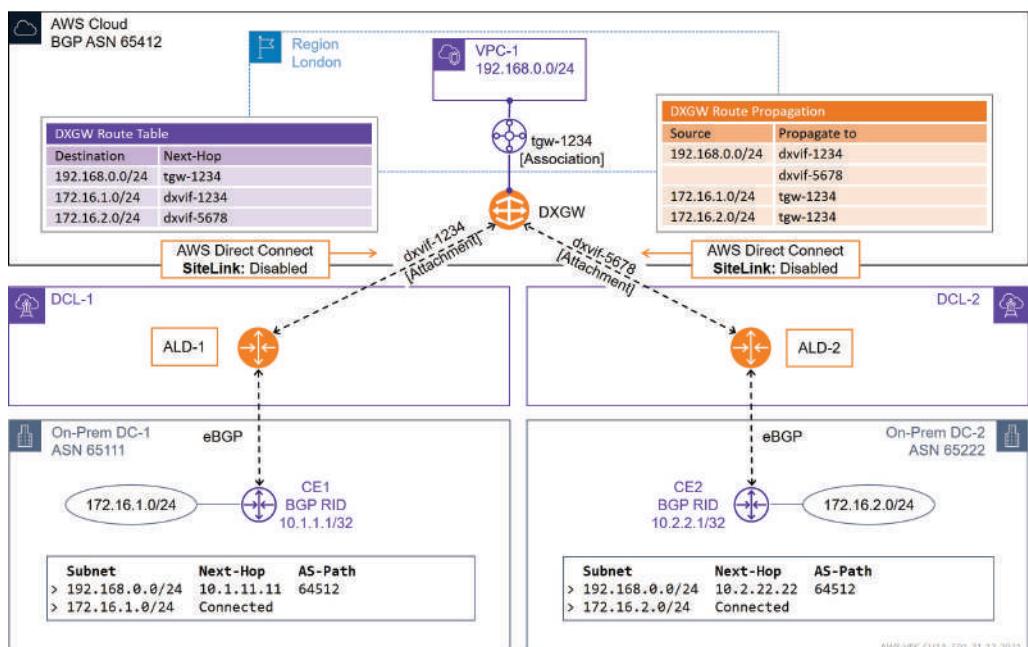


Figure 14-1: AWS DX SiteLink Disabled.

SiteLink Enabled on VIFs

Figure 14-2 shows the example of the routing scheme after we have enabled DX SiteLink on dxvif-1234 and dxvif-5678. Now DXGW propagates routing information about subnet 172.16.1.0/24 received over dxvif-1234 to TGW and dxvif-5678. Besides, it propagates routing information about subnet 172.16.2.0/24 received from dxvif-5678 to TGW and dxvif-1234. DXGW prepends the BGP AS-Path list as many times as there are Amazon Logical Devices (ALDs). In our example, we have two ALDs. For example, The CE-2 receives BGP Update about subnet 172.16.1.0/24 from ALD-2 with BGP AS numbers 64512 64512 65111 listed in the AS-Path attribute list. After routing updates, the traffic path between sites will be tunneled over AWS BackBone, bypassing AWS region, using the shortest path. Note that all traffic over AWS BackBone is encrypted. Note that DXGW advertises the subnets we are using for eBGP peering between ALDs and CEs.

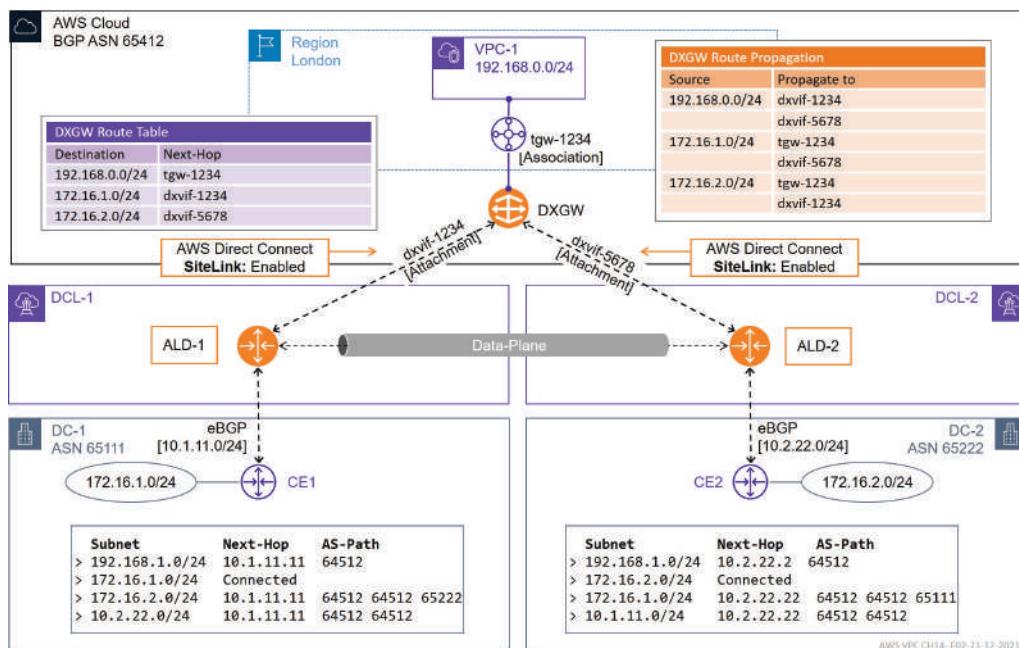


Figure 14-2: AWS DX SiteLink Enabled.

Enabling SiteLink

You can enable the Virtual Interface SiteLink feature by selecting the Virtual Interfaces option under the Direct Connect service. Then, select the VIF Settings and then scroll down to Additional settings. To enable SiteLink, mark the Enable SiteLink check box.

Transit virtual interface settings

▼ **Additional settings**

Address family - optional
Determines whether the virtual interface is created with an IPV4 or IPV6 peering.

IPV4
 IPV6

Your router peer ip - optional
The BGP peer IP configured on your endpoint
10.1.11.1/24

Amazon router peer ip - optional
The BGP peer IP configured on the AWS endpoint.
10.1.11.11/24

BGP authentication key - optional
The password that will be used to authenticate the BGP session.
my-secret-password

Jumbo MTU (MTU size 8500) - optional
Allow MTU size of 8500 on virtual interface.
 Enabled

Enable SiteLink - optional
Enable direct connectivity between Direct Connect points of presence.
 Enabled

AWS VPC CH14- F03-14-12-2021

Figure 14-3: AWS DX SiteLink Configuration

SiteLink Implementation

In figure 14-4, customer sites DC-1 and DC-2 have IP connectivity over the corporate BackBone network. Both sites have a Dedicated Direct Connect connection with the same DXGW. At this phase, we haven't yet enabled SiteLink in neither VIFs. Our intent is to start using the AWS BackBone Network as a primary site-to-site WAN between DC-1 and DC-2 and the Corporate BackBone as a backup WAN connection. To do that, we will use BGP standard communities to signal which egress path CE-2 should use when forwarding data packets toward subnet 172.16.1.0/24. In the first phase, we create a BGP ingress policy on CE-2 that sets the Local-Preference value to 50 for all received NLRIIs with BGP community 65111:50, and the Local-Preference value to 500 for all NLRIIs with BGP community 65111:500. Next, we configure BGP Egress policies on CE-1. We attach the BGP community 65111:50 to NLRIIs, which we advertise to CE-2 over the Corporate BackBone network. Besides, we add the BGP community 65111:500 to NLRIIs, which we advertise to ALD-1. Remember to enable neighbor-specific send-community option under BGP configuration, otherwise communities are not attached to outgoing BGP Update messages. Next, we turn on the SiteLink feature on dxvif-1234 and dxvif-5678. Because we have changed the BGP egress policy on CE-1, we have to re-send BGP Update messages to BGP peers. Cisco IOS command *clear ip bgp * soft out* sends BGP routes from the local routing table through the BGP Egress Policy Engine and attaches communities to BGP Update messages. After BGP soft reset, CE-2 receives new BGP Updates about subnet 172.16.1.0/24, now with BGP community 65111:500 from ALD-2 and the BGP community 65111:50 from BBR-2. Note that AWS Control-Plane protocol retains customer BGP standards communities during the route propagation process. CE-2 imports the NLRIIs from the neighbor-specific Adj-RIB-In table into the BGP Loc-RIB table. During the import process, the BGP Ingress policy sets the Local-Preference value 50 to route received via corporate BackBone and the Local-Preference value 500 for routes received from the AWS side. Then CE-2 runs the BGP Best Path selection algorithm. CE-2 installs the route received from ALD-2 into the routing table because of the higher Local-Preference value.

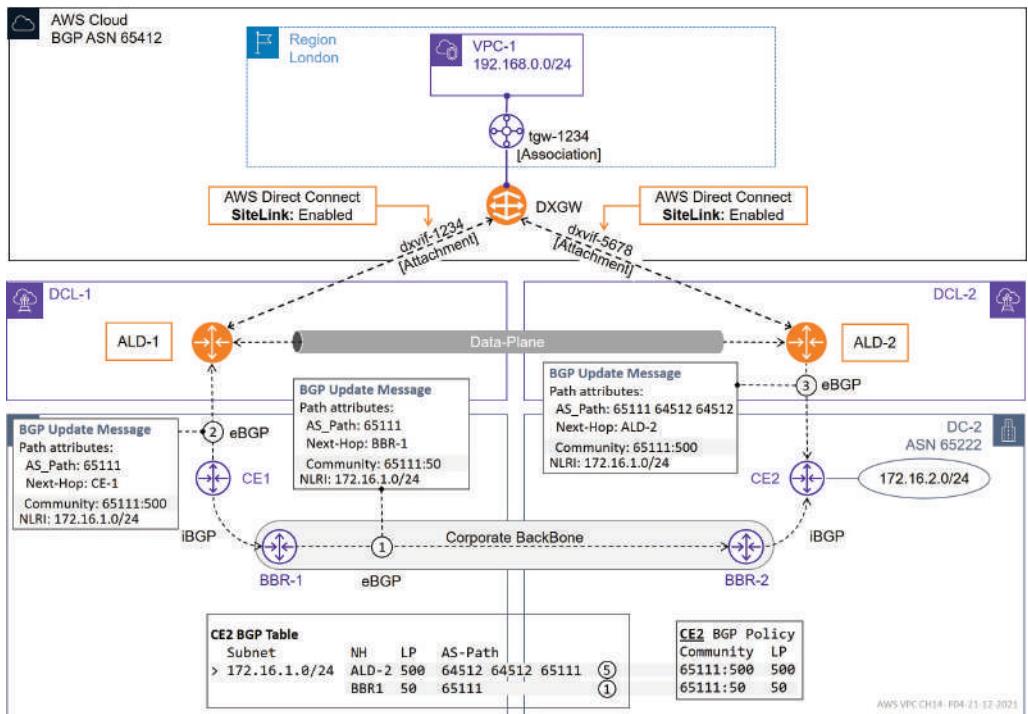


Figure 14-4: Implementing DX SiteLink

References

- [AWS-DXSL] Introducing AWS Direct Connect SiteLink Abdul Rahim & Sidhartha Chauhan - December 1, 2021:
<https://aws.amazon.com/blogs/networking-and-content-delivery/introducing-aws-direct-connect-sitelink/>
- [NVO3] An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3) D. Black et al. <https://datatracker.ietf.org/doc/html/rfc8014>
- [AWS-DX-API] Describe VirtualInterfaces
https://docs.aws.amazon.com/directconnect/latest/APIReference/API_DescribeVirtualInterfaces.html

Chapter 15: Direct Connect - Public Virtual Interface

Introduction

This chapter introduces how we can use an AWS Direct Connect connection (DX) and a Public Virtual Interface (P-VIF) attached with it to access AWS Public Service like Amazon Simple Storage Service (S3) and AWS DynamoDB.

BGP Updates from on-Prem to AWS

The upper part of figure 15-1 illustrates how we can adjust BGP Update messages propagation using BGP communities. The eBGP peering between AWS and Customer Devices is established over DX connection using public IP addresses. I haven't registered any public IP address, therefore I am using the "subnet" x.y.z.0/30. Our example company, Company-X, uses Private IP subnets on the on-prem location. For this reason, we have to hide the real source IP addresses of data packets towards AWS Public services. We use Port Address Translations (PAT) for changing the source IP address to IP address x.y.z.1, which is the Public IP address that the Customer Device uses for eBGP peering. That's why we advertise the "subnet" z.x.y.0/30 to AWS.

The geographical scope of the BGP Update message without BGP community value, or with BGP community 7224:9300, is global. In our example, the DX Location (DCL) is on AWS Region eu-north-1 (Stockholm). When the AWS device in our example receives BGP updates with BGP community 7224:9300 (2) or without BGP community (1), it propagates the route within AWS region eu-north-1, from where it is advertised globally (EU Intra and Inter-Regional). However, AWS does not advertise subnet carried within the BGP Update message to other customers over their DX or to the Internet. That said, routes we are advertising to AWS stay within the AWS Backbone. The subnets we advertise to AWS are reachable over our DX connection for all customers' EC2 instances running on the VPCs on the AWS cloud. Therefore, we have to use the Data-Plane ingress policy filter at your side. In our example, we are using a FireWall for filtering traffic.

We use a BGP community value 7224:9200 on the third BGP Update example. The scope of this BGP Update message is Intra-Continental AWS regions. In our example, the AWS device peering with our CE device is in the Stockholm region (eu-north-1), so the BGP Update message stays with Europe.

In the fourth BGP Update example, we use BGP community value 7224:9100. The scope of this community value is intra-Region. AWS Backbone devices do not advertise the subnet to any other region. In our example, the NLRI carried within the fourth BGP Update message stays within the AWS region eu-north-1.

AWS uses Unicast Reverse Path Forwarding (uRPF) checks against IP address spoofing. When the AWS DX Device receives traffic from the Customer Device, it verifies that it has a route to the source IP address. AWS does not describe how they have implemented the uRPF. However, I assume that the route doesn't have to point back to the interface where the packet is received because it excluded the asymmetric routing and eBGP Equal-Cost Multipath (ECMP) solutions. After these validations, AWS DX Device forwards routes packet.

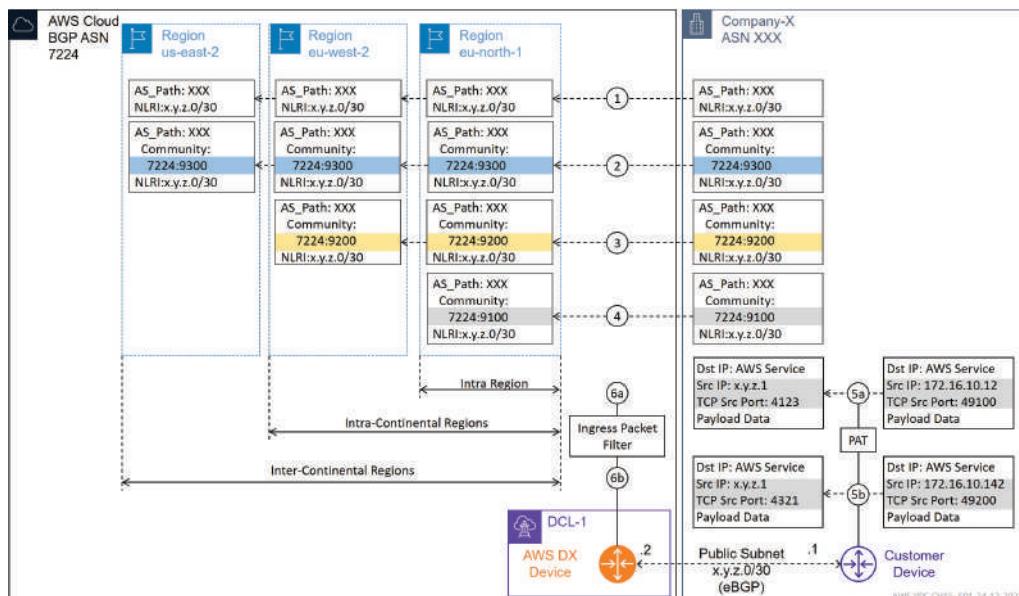


Figure 15-1: BGP Update Message from on-Prem to AWS.

BGP Updates from AWS to on-Prem

AWS adds BGP communities for BGP Updates advertised over Public VIF to customers. BGP communities 7224:8100 and NO_EXPORT have been added to all BGP Updates messages, which describe NLRI originated from the same AWS region in which we terminate our DX connection. The NO_EXPORT community states that this NLRI have to stay within our BGP domain. AWS adds BGP communities 7224:8200 and NO_EXPORT to all BGP Updates messages originated from the Intra-Continent AWS regions. The NLRI outside the continent, in which we have terminated our DX connection, carry only BGP community NO_EXPORT. Note that BGP Update has at minimum three BGP ASN listed in the AS-Path list.

In figure 15-2, AWS advertises three subnets to us. The subnet 52.46.220.0/22 is originated from the same AWS region (eu-north-1) in which we have terminated our DX connection, so the BGP Update message carries communities 7224:8100 and NO_EXPORT. The second advertised subnet is 52.56.0.0/16. From our perspective, it is originated from the Intra-Continental AWS region (eu-west-2). That is why the BGP Update carries the BGP community value 7224:8200. The subnet 54.245.0.0/16 is originated from the AWS region us-west-2, which is the Inter-Continent region from our point of view. The BGP Update message describing this subnet has only the NO_EXPORT BGP community.

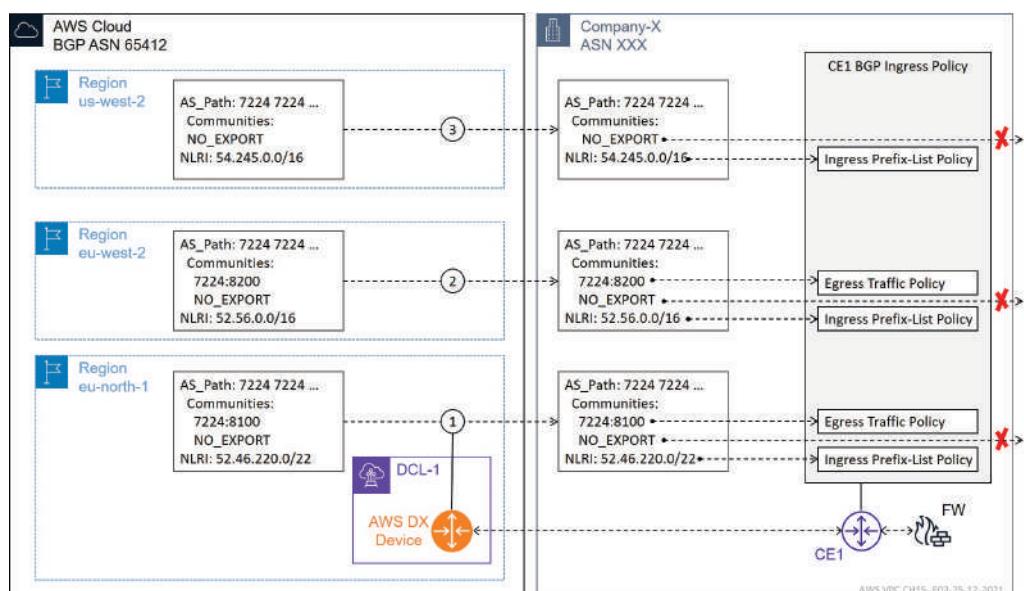


Figure 15-2: BGP Update Message from AWS to on-Prem.

We can define an egress policy based on BGP communities. For example, only BGP Updates with the BGP community 7224:8100 are eligible to be installed into BGP-Loc RIB and further to RIB (if it is the best route). Besides, we can set the BGP Local-Preference value based on the received BGP community value. If we want to allow connection initiation from the subnets from the AWS side, we also have to permit it in our Fire Wall policy rule.

Create Public VIF

Navigate to the *Virtual Interface* window and select the *Public* as a Virtual Interface type.

The screenshot shows the 'Create virtual interface' page in the AWS Direct Connect console. The navigation path is 'Direct Connect > Virtual interfaces > Create'. The main title is 'Create virtual interface'. A descriptive text explains that you can create a private virtual interface to connect to your VPC or a public virtual interface to connect to AWS services like S3 and Glacier. It notes that for private interfaces, one is needed per VPC, while a public interface can access all AWS services. A 'Learn more' link is provided. Below this, a section titled 'Virtual interface type' contains three options: 'Private', 'Public', and 'Transit'. The 'Public' option is selected, indicated by a blue outline and checked radio button. A tooltip for 'Public' states: 'A public virtual interface can access all AWS public services using public IP addresses.' The 'Private' option has a tooltip: 'A private virtual interface should be used to access an Amazon VPC using private IP addresses.' The 'Transit' option has a tooltip: 'A transit virtual interface is a VLAN that transports traffic from a Direct Connect gateway to one or more transit gateways.' At the bottom right of the page, there is a small watermark: 'AWS VPC CH15 - F03-26-12-2021'.

Figure 15-3: Configuring Public VI – Virtual Interface Type.

Scroll down to the *Public virtual interface settings* section. Give the name to the Public VIF and choose the DX connection to which you want to attach the VIF from the *Connection* drop-down menu. Besides, define the VLAN-Id for VIF.

Public virtual interface settings

Virtual interface name
A name to help you identify the new virtual interface.

Name must contain no more than 100 characters. Valid characters are a-z, 0-9, and – (hyphen)

Connection
The physical connection on which the new virtual interface will be provisioned.

Virtual interface owner
The account that will own the virtual interface.
 My AWS account
 Another AWS account

VLAN
The Virtual Local Area Network number for the new virtual interface

Valid ranges are 1 - 4094

AWS VPC CH15- F04-26-12-2021

Figure 15-4: Configuring Public VI – Virtual Interface Settings.

Fill in the BGP ASN field and select the IPv4 as BGP Address Family. We use eBGP peering IP address as a source IP address for egress traffic (remember that we are using PAT in this example). That is why the subnet has to be public. Subnet x.y.z.0/30 in the figure below simulates the public IP subnet. We are using only private IP subnets within the on-prem site. Therefore, we only have to advertise the eBGP peering subnet to AWS. Note that AWS verifies that we own the subnet we are advertising.

BGP ASN
The Border Gateway Protocol Autonomous System Number of your on-premises router for the new virtual interface.

Valid ranges are 1 - 2147483647.

Address family
Determines whether the virtual interface is created with an IPV4 or IPV6 peering.

IPV4
 IPV6

Your router peer ip
The BGP peer IP configured on your endpoint

Amazon router peer ip
The BGP peer IP configured on the AWS endpoint.

Prefixes you want to advertise
The list of IP prefix CIDRs that you want to advertise to Amazon over this virtual interface, including the peer IPs.

Specify up to 100 prefixes separated with commas or put each on a new line.

AWS VPC CH15- F05-26-12-2021

Figure 15-5: Configuring Public VI – eBGP Peering.

You can set the BGP authentication key and Tags in the Additional settings section. Click the Create virtual interface button when filling in all required information.

▼ Additional settings

AWS VPC CH15- F06-26-12-2021

BGP authentication key - optional
The password that will be used to authenticate the BGP session.

Tags
Specified tags to help identify a AWS Direct Connect resource.
No tags associated with the resource

Add tag

Cancel **Create virtual interface**

Figure 15-6: Configuring Public VI – Additional Settings, BGP Authentication Key.

References

AWS Direct Connect User Guide: Routing policies and BGP communities

<https://docs.aws.amazon.com/directconnect/latest/UserGuide/routing-and-bgp.html>

The List of AWS Public IP subnets

<https://ip-ranges.amazonaws.com/ip-ranges.json>

[RFC 1997] R. Chandra et al., “BGP Community Attribute”

RFC 1997, August 1996.

[RFC 8642] J. Borkenhagen et al., “Policy Behavior for Well-Known BGP

Communities” RFC 1997, August 2019.

In case you want to learn more about LISP, OMP, or BGP EVPN...
These books are available at [Leanpub.com](https://leanpub.com) (pdf) and Amazon (Kindle, paperback)

