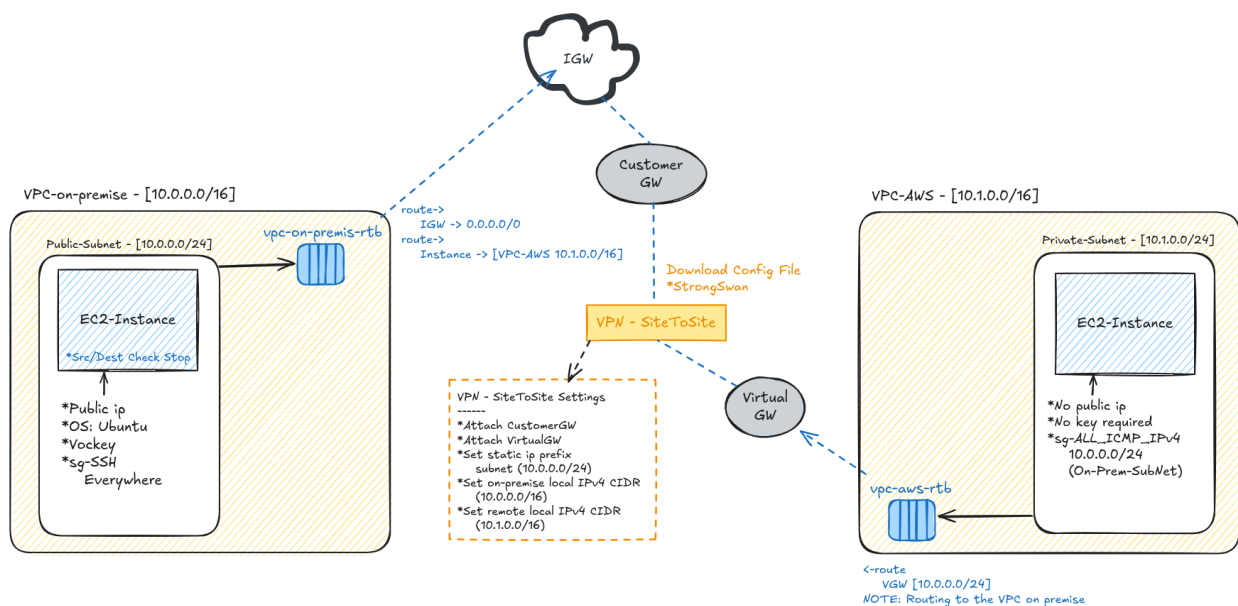


In this document, I will thoroughly go through the steps to implement a Site-to-Site Virtual Private Network (VPN) using an on-premises network simulated as a separate Virtual Private Cloud (VPC). And the other will be the VPC we connect to via the VPN.

Requirements

Topology

SITE TO SITE VPN



Resources

1. vpc-on-premise (10.0.0.0/16)
 1. public-subnet (10.0.0.0/24)
 1. vpc-on-premise-rtb
 1. To Internet Gateway (0.0.0.0/0)
 2. To Instance of the other side instance 10.1.0.0/16
 2. ec2-instance
 1. OS: Ubuntu
 2. Public IP
 3. VOCKEY

4. SG-SSH - Everywhere.
2. Internet Gateway
2. aws-vpc (10.1.0.0/16)
 1. private-subnet (10.1.0.0/24)
 1. vpc-aws-rtb
 1. To Virtual Gateway
 2. ec2-instance
 1. No public ip
 2. No key
 3. OS: Amazon Linux (default)
 4. SG-all icmp ipv4 from 10.0.0.0/24 (on-premise-public-subnet)
3. Customer Gateway (will use the public ip address of the on-premise-instance)
4. Internet Gateway (will be used in the on-premise-public-subnet)
5. Virtual Gateway (will be used on the aws-vpc private-subnet routing table)
6. Site to Site VPN Connection
 1. *Attach CustomerGW
 2. Attach VirtualGW
 3. Set static ip prefix subnet (10.0.0.0/24)
 4. Set on-premise local IPv4 CIDR (10.0.0.0/16)
 5. Set remote local IPv4 CIDR (10.1.0.0/16)
 6. In the static routes add the vpc-on-premise CIDR Block (10.0.0.0/16) or the public-subnet of the on-premise (10.0.0.0/24).

Lets Create VPCs and subnets

VPC-on-premise

[VPC](#) > [Your VPCs](#) > Create VPC

Create VPC [Info](#)

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

☒ VPC only ☐ VPC and more

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

VPC-on-premise

IPv4 CIDR block [Info](#)

☒ IPv4 CIDR manual input ☐ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

CIDR block size must be between /16 and /28.

Setting the IPv4 CIDR at 10.0.0.0/16

Another VPC to be connected to from the VPC-on-premis

[Create VPC](#) [Info](#)

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

☒ VPC only ☐ VPC and more

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

VPC

IPv4 CIDR block [Info](#)

☒ IPv4 CIDR manual input ☐ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.1.0.0/16

CIDR block size must be between /16 and /28.

Setting the IPv4 CIDR at 10.1.0.0/16

Your VPCs (4) Info							Last updated 1 minute ago	Actions	Create VPC
Search									
	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP			
<input type="checkbox"/>									
<input type="checkbox"/>									
<input type="checkbox"/>	VPC-on-premise	vpc-033a5e289c0a5b8d3	Available	10.0.0.0/16					
<input type="checkbox"/>	VPC	vpc-0ed691470d442b483	Available	10.1.0.0/16					

Lets create the subnets

On-Premise Public Subnet

First subnet will be created for the VPC-on-premise. Its going to be named `on-premise-public-subnet`. It needs a routing table and also needs an Internet Gateway (IGW). The on premise subnet will use the VPC-on-premise at CIDR block `10.0.0.0/16` and this subnet, will use `10.0.0.0/24` of the vpc-on-premise CIDR block, this will give the subnet 256 hosts.

I will be using only one host, for the EC2 instance to connect to the VPN.

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.

vpc-033a5e289c0a5b8d3 (VPC-on-premise) 1

Associated VPC CIDRs

IPv4 CIDRs
10.0.0.0/16

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.

on-premise-public-subnet 2

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.

No preference

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

10.0.0.0/16 3

IPv4 subnet CIDR block

10.0.0.0/24 256 IPs 4

< > ^ v

AWS VPC Private Subnet

Secondly, creating a subnet for the other VPC, `vpc`. For this one, I will name it `aws-private-subnet`. It needs a routing table as well. This subnet will use the VPC ip block `10.1.0.0/16` and this subnet, will use `10.1.0.0/24` from the given VPC block, this will give the subnet 256 hosts.

Create subnet Info

VPC

VPC ID

Create subnets in this VPC.

vpc-0ed691470d442b483 (VPC)

Associated VPC CIDRs

IPv4 CIDRs

10.1.0.0/16

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.

aws-private-subnet

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.

No preference

IPv4 VPC CIDR block Info

Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

10.1.0.0/16

IPv4 subnet CIDR block

10.1.0.0/24

256 IPs

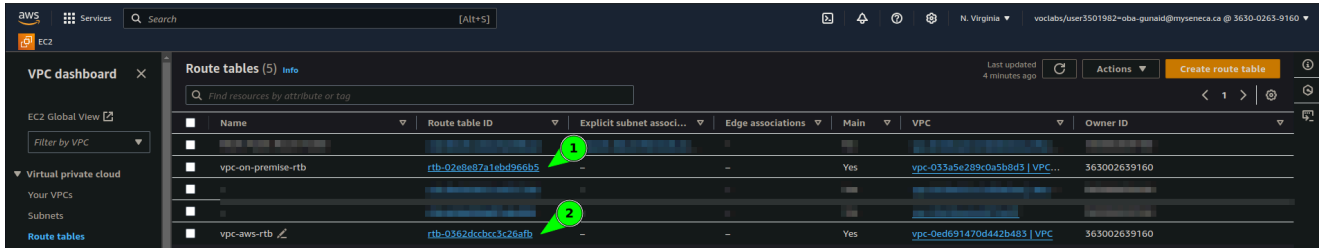
Total Subnets

Total of required subnets for this Site To Site VPN. Only two subnets, one will be used for the on-premise machine, and the other one will be used on other side, will be used to connect to the instances within the subnet using the tunnels by the Site To Site VPN.

Subnets (9) <small>Info</small>								
<input type="text" value="Find resources by attribute or tag"/>								
<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	
<input type="checkbox"/>	on-premise-public-subnet	subnet-0d6105fefece09c5d	Available	vpc-033a5e289c0a5b8d3 VPC...	10.0.0.0/24	-	-	1
<input type="checkbox"/>	aws-private-subnet	subnet-07049c6d70dd5a92	Available	vpc-0ed691470d442b483 VPC...	10.1.0.0/24	-	-	2

Lets edit or create routing table.

I will be using the routing tables that were created while creating the subnets. You can edit them and renamed them if you want or create new ones, thats totally up to you. I will go through editing their routings later.



Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
vpc-on-premise-rtb	rtb-02a8e87a1ebd966b5	-	-	Yes	vpc-033a5e289c0a5b8d3 VPC...	363002639160
vpc-on-premise-public-subnet-rtb	rtb-0362dccbccc3c26afb	-	-	Yes	vpc-0ed691470d442b483 VPC...	363002639160
vpc-aws-rtb	rtb-0362dccbccc3c26afb	-	-	Yes	vpc-0ed691470d442b483 VPC...	363002639160

Lets create an EC2 instance for vpc-on-premise

Its required since we need its public ip address for the *Customer Gateway* later. As well, I will be going through editing the routing table of the `on-premise-public-subnet`.

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

on-premise-instance

[Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents

Quick Start

Amazon
Linux



macOS



Ubuntu



Windows



Red Hat



SUSE LI



[Browse more AMIs](#)

Including AMIs from
AWS, Marketplace and
the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0866a3c8686eaebeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

I will name this instance `on-premise-instance` and use Ubuntu, as it includes pre-built packages. I will also use **strongSwan** to establish the connection to the Site-to-Site VPN.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

1

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

vockey

2

Create new key pair

I am going to be using the **vockey** provided for me. You can create a pair, don't forget to download the private key, we will be needing that to login to the instance through SSH.

▼ Network settings [Info](#)

VPC - *required* | [Info](#)

vpc-033a5e289c0a5b8d3 (VPC-on-premise)

10.0.0.0/16

1

Create new VPC

Subnet | [Info](#)

subnet-0d6105fefece09c5d

on-premise-public-subnet

VPC: vpc-033a5e289c0a5b8d3 Owner: 363002639160 Availability Zone: us-east-1f
Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.0.0/24

2

Create new subnet

Auto-assign public IP | [Info](#)

Enable

3

Additional charges apply when outside of free tier allowance

These are the network settings. Make sure to assign a public ip address for this instance.

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group 1 ☐ Select existing security group

Security group name - required 2

on-premise-sg-OnlySSH

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!\$*

Description - required | Info

on-premise-sg-OnlySSH

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) Remove

Type | Info 3 ssh

Protocol | Info TCP

Port range | Info 22

Source type | Info 4 Anywhere

Source | Info

Description - optional | Info

Warning: Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. X

Make sure to add SSH security group, we will want to connect to it later.

NOTE

You can assign your public IP address to limit access to that instance for secure login. To do this, make sure to select *My IP* from *Source type*. This ensures that only your home public IP address is allowed to connect to the instance.

Instances (1/2) Info							
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Bastion Host	i-03b911677fdd07037	Running	t2.micro	2/2 checks passed	Failed to fetch	us-east-1a	ec2-34-229-223-167.cc
on-premise-instance	i-0826dbf918993f902	Running	t2.micro	Initializing	Failed to fetch	us-east-1f	-

Denote public ip

Copy the public ip address of the created instance, we will be needing that later for the *Customer Gateway* as well as to login to that instance.

Also, we will need to stop Source and Destination check from the instance.

The screenshot shows the AWS Management Console 'Instances' page. A table lists two instances: 'Bastion Host' (ID: i-03b911677fdd07037, state: Running) and 'on-premise-instance' (ID: i-0826dbf918993f902, state: Running). The 'on-premise-instance' is selected, and the 'Actions' menu is open, showing 'Change source/destination check' as option 4. A modal dialog titled 'Change Source / destination check' is displayed. It contains the following information: Instance ID 'i-0826dbf918993f902 (on-premise-instance)', Network Interface 'eni-0d08db96553239ee1', and a 'Source / destination checking' section with a 'Stop' checkbox (labeled 1). At the bottom of the dialog are 'Cancel' and 'Save' buttons (labeled 2).

Lets create the required gateways

Internet Gateway

You will need to navigate to the **Internet Gateway** section and create a new Internet Gateway. I will name this `on-premise-igw`. Remember, we are only creating one Internet Gateway to be used for the `vpc-on-premises` VPC.

We will later add this to the routing table of the on-premise-vpc

aws

Services

Search [Alt+S]

EC2

VPC > Internet gateways > Create internet gateway

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.

on-premise-igw

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	
<div><div>Q Name</div><div>X</div></div>	<div><div>Q on-premise-igw</div><div>X</div></div>	<div>Remove</div>
<div>Add new tag</div> <div>You can add 49 more tags.</div>		

Cancel

Create internet gateway

[Alt+S]

VPC > Internet gateways > igw-0fc6e732e61eb3727

igw-0fc6e732e61eb3727 / on-premise-igw

Details [Info](#)

Internet gateway ID igw-0fc6e732e61eb3727	State Detached	VPC ID -	Owner 363002639160
--	-------------------	-------------	-----------------------

Tags

Search tags

Key	Value
Name	on-premise-igw

Actions

Attach to VPC

Detach from VPC

Manage tags

Delete

Manage tags

< 1 >

Don't forget to attach it to the **vpc-on-premise**

on-premise-public-subnet - Internet Access

Before we move on, lets take a look at the routing table of *on-premise-public-subnet*, we will need to add this **Internet Gateway** to it. This will ensure that all instances that are in this subnet, will have internet access.

VPC > Route tables > rtb-02e8e87a1ebd966b5

rtb-02e8e87a1ebd966b5 / vpc-on-premise-rtb

Actions ▾

Details Info

Route table ID rtb-02e8e87a1ebd966b5	Main Yes	Explicit subnet associations -	Edge associations -
VPC vpc-033a5e289c0a5b8d3 VPC-on-premise	Owner ID 363002639160		

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

Routes (1)

Filter routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No

VPC > Route tables > rtb-02e8e87a1ebd966b5 > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	Internet Gateway	-	No

Add route

Cancel Preview **Save changes**

Updated routes for rtb-02e8e87a1ebd966b5 / vpc-on-premise-rtb successfully

Details

VPC > Route tables > rtb-02e8e87a1ebd966b5

rtb-02e8e87a1ebd966b5 / vpc-on-premise-rtb

Actions ▾

Details Info

Route table ID rtb-02e8e87a1ebd966b5	Main Yes	Explicit subnet associations -	Edge associations -
VPC vpc-033a5e289c0a5b8d3 VPC-on-premise	Owner ID 363002639160		

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

Routes (2)

Filter routes

Destination	Target	Status	Propagated
0.0.0.0/0	lgw-0fc6e732e61eb3727	Active	No
10.0.0.0/16	local	Active	No

Customer Gateway

Under *Virtual Private Network* VPN section, navigate to *Customer Gateway*. Create a new one.

Create customer gateway Info

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

Details

Name tag - optional

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

Value must be 256 characters or less in length.

BGP ASN | Info

The ASN of your customer gateway device.

Value must be in 1 - 4294967294 range.

IP address | Info

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

Certificate ARN - optional

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

Device - optional

Enter a name for the customer gateway device.

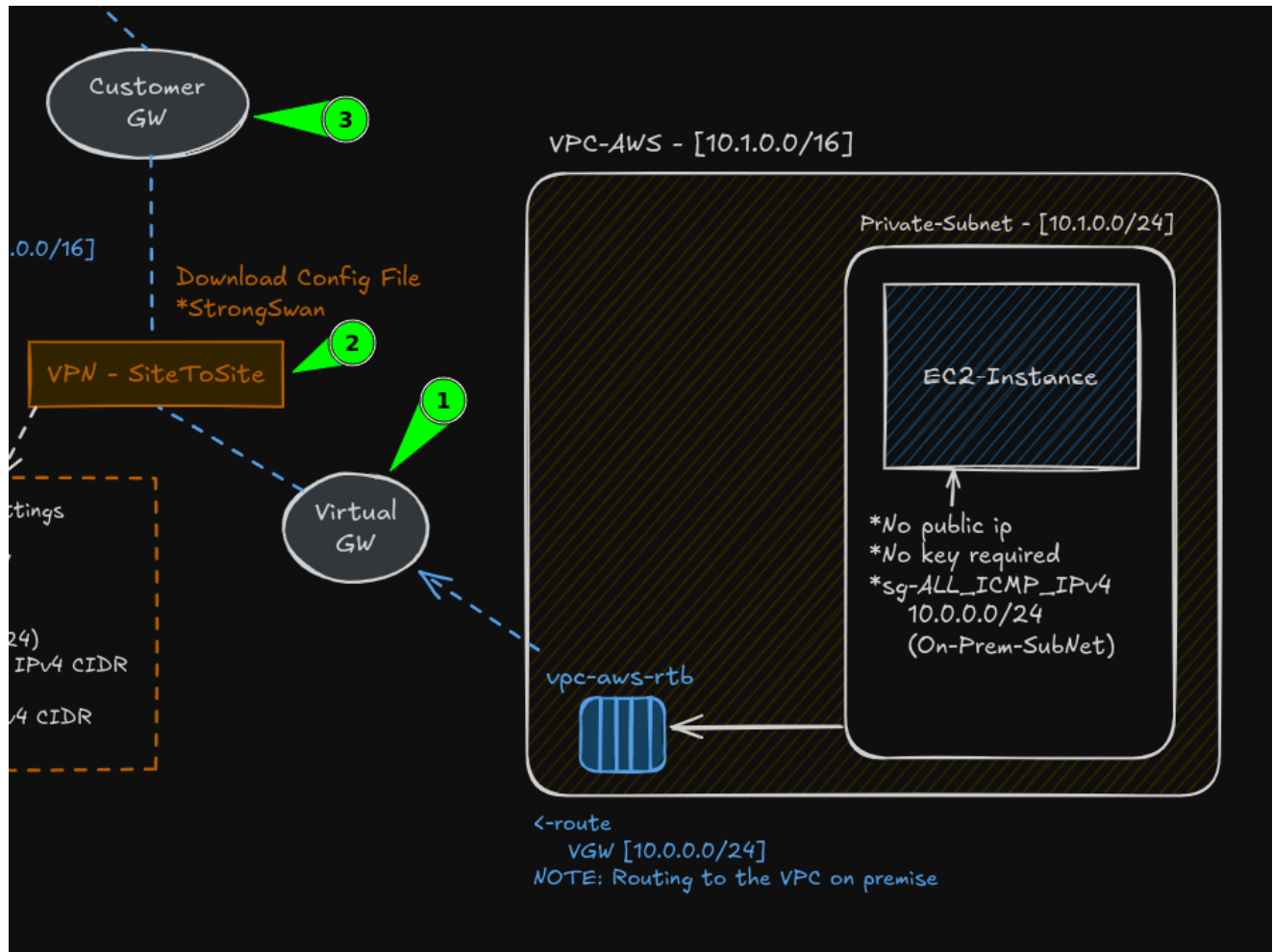
I used the public IP address of the `on-premise-instance` I created earlier. This instance will serve as the on-premises site.

You successfully created cgw-Od41f5ae7d2a2ee43 / on-premise-customer-gateway.							
Customer gateways (1) <small>Info</small>							
<input type="text" value="Find resource by attribute or tag"/>							
Name	Customer gateway ID	State	BGP ASN	IP address	Type	Certificate ARN	
on-premise-customer-gateway	cgw-Od41f5ae7d2a2ee43	Available	65000	100.24.122.7	Ipsec.1	-	

Virtual Private Gateway

The virtual gateway is required to establish a Site-to-Site VPN connection. This virtual gateway will be used by the Site-to-Site VPN and the `vpc`.

NOTE: Look at the topology.



Create virtual private gateway Info

A virtual private gateway is the VPN concentrator on the Amazon side of the site-to-site VPN connection.

Details

Name tag - optional

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

vpc-virtual-gateway

Value must be 256 characters or less in length.

Autonomous System Number (ASN)

☒ Amazon default ASN

☐ Custom ASN

Tags

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

Key

Name

Value - optional

vpc-virtual-gateway

Remove

Add new tag

You can add up to 49 more tags.

Cancel

Create virtual private gateway

This *Virtual Private Gateway* needs to be attached to the *aws-vpc* VPC .

Virtual private gateways (1/1) <small>Info</small>							Actions
Find resource by attribute or tag							Create virtual private gateway
Name	Virtual private gateway ID	State	Type	VPC	Amazon /		Attach to VPC
vpc-virtual-gateway	vgw-028b647e16784dbe3	Detached	ipsec.1	-	64512		Detach from VPC
							Manage tags
							Delete virtual private gateway

Attach to VPC Info

Details

Virtual private gateway ID

vgw-028b647e16784dbe3

Available VPCs

Attach the virtual private gateway to this VPC.

Select a VPC by ID or Name

Q

vpc-033a5e289c0a5b8d3 / VPC-on-premise

vpc-0ed691470d442b483 / VPC

vpc-0189c231e7d58e3ee / Work VPC

vpc-08e96ae4ed6f5a6fc

Cancel

Attach to VPC

Virtual private gateways (1) [info](#)

Find resource by attribute or tag

Name	Virtual private gateway ID	State	Type	VPC	Amazon ASN
vpc-virtual-gateway	vgw-028b647e16784dbe3	Attached	ipsec.1	vpc-0ed691470d442b483 VPC	64512

Buttons: Refresh, Actions, Create virtual private gateway

Now the VPC attached to the *Virtual Private Gateway*.

🔥 We are attaching the VPC not the vpc-on-premise

Site To Site VPN Connection

VPN connections [info](#)

Find resource by attribute or tag

Name	VPN ID	State	Virtual private gateway	Transit gateway	Customer gateway	Customer gateway add...	Inside IP
No VPN connections							
You do not have any VPN connections in this region							
Create VPN connection							

Buttons: Refresh, Actions, Download configuration, Create VPN connection

Now we will create a site to site vpn connection.

Create VPN connection Info

Select the resources and additional configuration options that you want to use for the site-to-site VPN connection.

Details

Name tag - optional

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

on-premise-vpn-aws-vpc

Value must be 256 characters or less in length.

Target gateway type Info

- ☒ Virtual private gateway
- ☐ Transit gateway
- ☐ Not associated

Select the virtual gateway we created

Virtual private gateway

vgw-028b647e16784dbe3

Customer gateway Info

- ☒ Existing
- ☐ New

Select the customer gateway we created

Customer gateway ID

cgw-0d41f5ae7d2a2ee43

Routing options Info

- ☐ Dynamic (requires BGP)
- ☒ Static

Will be using static routing.

Static IP prefixes Info

Add static IP prefix

10.0.0.0/24

For the 'Static IP prefix' this will be our on-premise-public-subnet CIDR Block range, which is 10.0.0.0/24.

Local IPv4 network CIDR - optional

The IPv4 CIDR range on the customer gateway (on-premises) side that is allowed to communicate over the VPN tunnels. The default is 0.0.0.0/0.

10.0.0.0/16

For the local IPv4 network, will be using the vpc-on-premise IP CIDR Block range, which is 10.0.0.0/16.

Remote IPv4 network CIDR - optional

The IPv4 CIDR range on the AWS side that is allowed to communicate over the VPN tunnels. The default is 0.0.0.0/0.

10.1.0.0/16

For the remote IPv4 network CIDR, will be using the AWS-VPC IP CIDR Block range, which is 10.1.0.0/16.

VPN connections (1/1) Info

Find resource by attribute or tag

Name	VPN ID	State	Virtual private gateway	Transit gateway	Customer gateway	Customer gateway add...
on-premise-vpn-aws-vpc	vpn-0a9a6a7f375bde893	Pending	vgw-028b647e16784dbe3	-	cgw-0d41f5ae7d2a2ee43	100.24.122.7

VPN connection vpn-0a9a6a7f375bde893 / on-premise-vpn-aws-vpc

Details | Tunnel details | Static routes | Tags

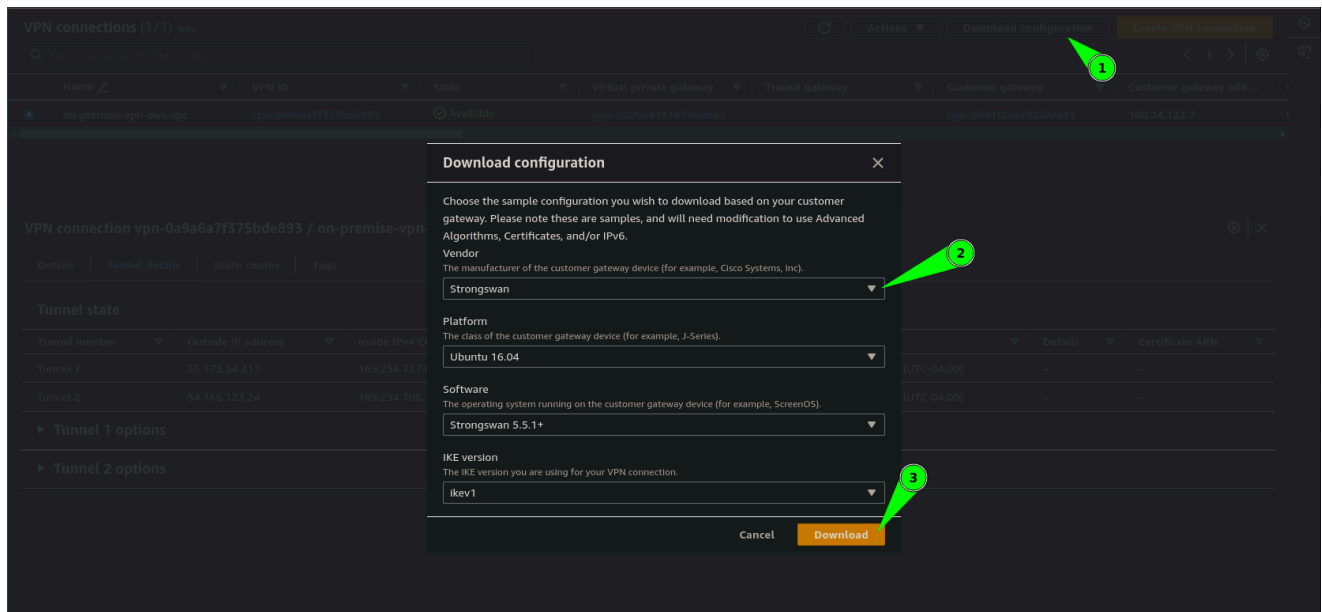
Tunnel state

Tunnel number	Outside IP address	Inside IPv4 CIDR	Inside IPv6 CIDR	Status	Last status change	Details	Certificate ARN
Tunnel 1	35.173.34.211	169.254.127.68/30	-	Down	October 25, 2024, 21:44:19 (UTC-04:00)	-	-
Tunnel 2	54.166.123.24	169.254.108.24/30	-	Down	October 25, 2024, 21:44:19 (UTC-04:00)	-	-

Tunnel 1 options

Tunnel 2 options

As you can see the tunnels are down. Lets download the configuration file for now. I will be using *strongSwan*.



Launch an EC2 instance in VPC the non-premise VPC

This instance won't have a public ip address, and no key is required. Just in the security group, we will need to allow Allow All ICMP IPv4 from 10.0.0.0/24 which is the on-premise-public-subnet.

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name


VPC-Instance

1

[Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your Instance. Search or Browse for AMIs if you don't see what you are looking for below

 Search our full catalog including 1000s of application and OS images

2

Recents

Quick Start

Amazon
Linux



macOS



Ubuntu



Windows



Red Hat



SUSE LI



[Browse more AMIs](#)

Including AMIs from
AWS, Marketplace and
the Community

Amazon Machine Image (AMI)

3

Amazon Linux 2023 AMI

ami-06b21ccaeff8cd686 (64-bit (x86), uefi-preferred) / ami-02801556a781a4499 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible



▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

☐ All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software


▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Proceed without a key pair (Not recommended)

Default value ▼

 [Create new key pair](#)

▼ Network settings [Info](#)

VPC - *required* | [Info](#)

vpc-0ed691470d442b483 (VPC)

10.1.0.0/16

Subnet | [Info](#)

subnet-07049c6d70dde5a92

aws-private-subnet

VPC: vpc-0ed691470d442b483 Owner: 363002639160
Availability Zone: us-east-1f Zone type: Availability Zone
IP addresses available: 251 CIDR: 10.1.0.0/24

Auto-assign public IP | [Info](#)

Disable

Now will need to ensure this instance accepts traffic from a specific IP CIDR block.
10.0.0.0/24

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

Security group name - required

Source on-premise - All ICMP IPv4

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and . _ - / () # , @ [] + = & ; { } ! \$ *

Description - required | Info

Source on-premise - All ICMP IPv4 created 2024-10-26T01:58:24.881Z

Inbound Security Group Rules

▼ Security group rule 1 (ICMP, All, 10.0.0.0/24) Remove

Type Info	Protocol Info	Port range Info
All ICMP - IPv4	ICMP	All
Source type Info	Source Info	Description - optional Info
Custom	<input type="text" value="10.0.0.0/24"/>	e.g. SSH for admin desktop

Add security group rule

► **Advanced network configuration**

Lets go back to route tables now.

Before starting to configure the strongswan, lets make sure the route tables are set correctly, lets start with the AWS-VPC private-subnet. Since its closer to the **Virtual Private Gateway**. It needs to route traffic to the other on-premise-public-subnet. using the virtual private gateway.

Vpc-aws routing table

VPC > Route tables > rtb-0362dccbcc3c26afb

rtb-0362dccbcc3c26afb / vpc-aws-rtb

Actions

Details Info

Route table ID rtb-0362dccbcc3c26afb	Main Yes	Explicit subnet associations -	Edge associations -
VPC vpc-0ed691470d442b483 VPC	Owner ID 363002639160		

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

Routes (1)

Filter routes

Destination	Target	Status	Propagated
10.1.0.0/16	local	Active	No

VPC > Route tables > rtb-0362dccbcc3c26afb > Edit routes

Edit routes

Destination 10.1.0.0/16	Target local	Status Active	Propagated No
10.0.0.0/24	Virtual Private Gateway	Active	No
	vgw-028b647e16784dbe3		

Add route

Cancel Preview Save changes

Updated routes for rtb-0362dccbcc3c26afb / vpc-aws-rtb successfully

Details

VPC > Route tables > rtb-0362dccbcc3c26afb

rtb-0362dccbcc3c26afb / vpc-aws-rtb

Actions

Details Info

Route table ID rtb-0362dccbcc3c26afb	Main Yes	Explicit subnet associations -	Edge associations -
VPC vpc-0ed691470d442b483 VPC	Owner ID 363002639160		

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

Routes (2)

Filter routes

Destination	Target	Status	Propagated
10.0.0.0/24	vgw-028b647e16784dbe3	Active	No
10.1.0.0/16	local	Active	No

On-premise-public-subnet route table

VPC > Route tables > rtb-02e8e87a1ebd966b5

rtb-02e8e87a1ebd966b5 / vpc-on-premise-rtb

Actions

Details Info

Route table ID rtb-02e8e87a1ebd966b5	Main Yes	Explicit subnet associations -	Edge associations -
VPC vpc-033a5e289c0a5b8d3 VPC-on-premise	Owner ID 363002639160		

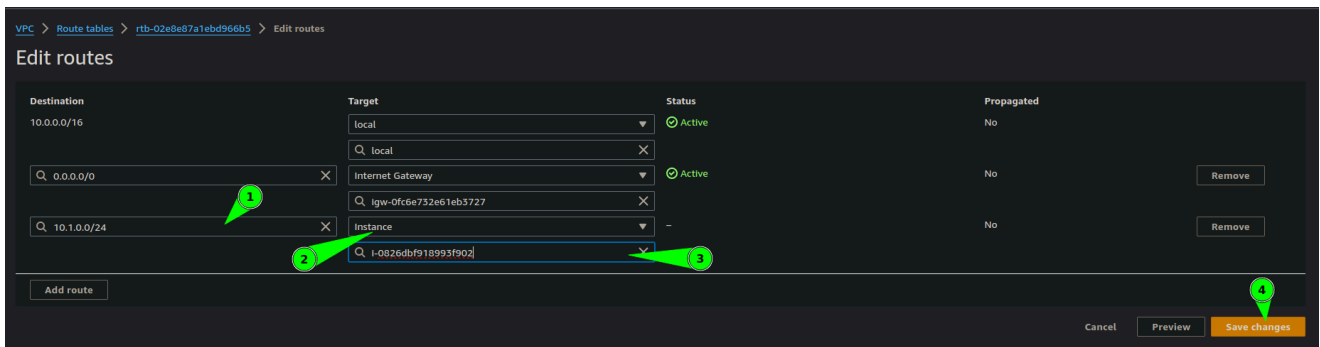
Routes | Subnet associations | Edge associations | Route propagation | Tags

Routes (2)

Filter routes

Destination	Target	Status	Propagated
0.0.0.0/0	lgw-0fc6e732e61eb3727	Active	No
10.0.0.0/16	local	Active	No

We need to add the the instance of the on-premise to route to the `aws-private-subnet` CIDR Block 10.1.0.0/24. This connection is not peering, but since Site to Site tunnels are connected, it should have a path to the other VPC's subnet.



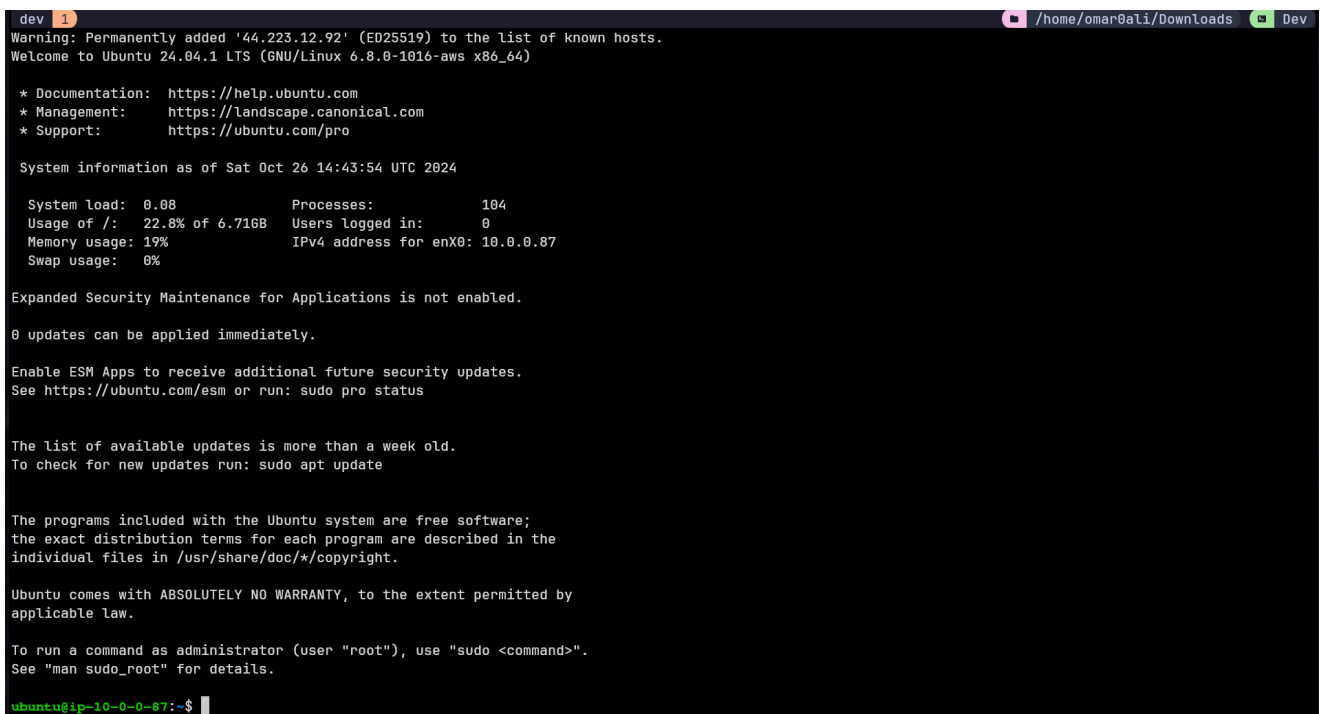
Lets configure the on-premise-instance

First, we need to login to our `on-premise-instance` to do that, we need to download our key from aws if you haven't. I did, and the first thing is to ensure the key can be only read. So using the linux command prompt type:

```
chmod 400 path/labsuser
```

Then we will need to connect to the instance, for example:

```
ssh -i labsuser.pem ubuntu@123.222.24.2
```



Configuration

Lets install `strongswan`


```
ubuntu@ip-10-0-0-87:~$ sudo apt install strongswan
```

Open `/etc/sysctl.conf`

Editor

I am using Vim to edit these files; you can use any editor, such as `nano`, which is also available.

Please refer to the documentation for either [Vim](#) or [Nano](#).

Check Your Configuration File

Check your configuration file downloaded from the Site To Site VPN Connection Page. You can follow the steps provided in the file. I am going through the steps used for strongswan.

```
sudo vim /etc/sysctl.conf
```

```

#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3

#####
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter
)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

"/etc/sysctl.conf" 64L, 2208B written                28,1                Top

```

Uncomment `net.ipv4.ip_forward=1`

Now we need to apply the changes:

```
sudo sysctl -p
```

Next, open `etc/ipsec.conf`

```
sudo vim /etc/ipsec.conf
```

```
dev 1 /home/omar0ali/Downloads Dev
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
    # strictcrpolicy=yes
    uniqueids = no
```

Uncomment `uniqueids = no`

And copy the following from your config file. And paste it at the end of `/etc/ipsec.conf` file.

```
# AWS VPN will also support AES256 and SHA256 for the "ike" (Phase 1) and "esp" (Phase 2) entries below.
# For Phase 1, AWS VPN supports DH groups 2, 14-18, 22, 23, 24. Phase 2 supports DH groups 2, 5, 14-18,
# 22, 23, 24
# To see Strongswan's syntax for these different values, please refer to https://wiki.strongswan.org/
# projects/strongswan/wiki/IKEv1CipherSuites

conn Tunnel1
    auto=start
    left=%defaultroute
    leftid=44.223.12.92
    right=3.216.254.210
    type=tunnel
    leftauth=psk
    rightauth=psk
    keyexchange=ikev1
    ike=aes128-sha1-modp1024
    ikelifetime=8h
    esp=aes128-sha1-modp1024
    lifetime=1h
    keyingtries=%forever
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    dpddelay=10s
    dpdtimeout=30s
    dpdaction=restart
    ## Please note the following line assumes you only have two tunnels in your Strongswan
    ## configuration file. This "mark" value must be unique and may need to be changed based on other entries in
    ## your configuration file.
    mark=100
    ## Uncomment the following line to utilize the script from the "Automated Tunnel Healthcheck and
    ## Failover" section. Ensure that the integer after "-m" matches the "mark" value above, and <VPC CIDR> is
    ## replaced with the CIDR of your VPC
    ## (e.g. 192.168.1.0/24)
    #leftupdown="/etc/ipsec.d/aws-updown.sh -ln Tunnel1 -ll 169.254.198.158/30 -lr 169.254.198.157/30
    -m 100 -r <VPC CIDR>"
```

We will need to modify the last line. Uncomment the last line that starts with `leftupdown=...` and update the `<VPC CIDR>` This CIDR range is the AWS VPC not the on-premise-vpc. Which is `10.1.0.0/16`

```

conn Tunnel1
    auto=start
    left=%defaultroute
    leftid=44.223.12.92
    right=3.216.254.210
    type=tunnel
    leftauth=psk
    rightauth=psk
    keyexchange=ikev1
    ike=aes128-sha1-modp1024
    ikelifetime=8h
    esp=aes128-sha1-modp1024
    lifetime=1h
    keyingtries=%forever
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    dpddelay=10s
    dpdtimeout=30s
    dpdaction=restart
    ## Please note the following line assumes you only have two tunnels in
    your Strongswan configuration file. This "mark" value must be unique and may
    need to be changed based on other entries in your configuration file.
    mark=100
    ## Uncomment the following line to utilize the script from the "Automated
    Tunnel Healthcheck and Failover" section. Ensure that the integer after "-m
    " matches the "mark" value above, and <VPC CIDR> is replaced with the CIDR of
    your VPC
    ## (e.g. 192.168.1.0/24)
    leftupdown="/etc/ipsec.d/aws-updown.sh -ln Tunnel1 -ll 169.254.198.158
    /30 -lr 169.254.198.157/30 -m 100 -r 10.1.0.0/16"

```

This is **tunnel 1**, we will do the same thing for **tunnel 2**. First, find it from your configuration file.

```

conn Tunnel2
    auto=start
    left=%defaultroute
    leftid=44.223.12.92
    right=34.236.97.73
    type=tunnel
    leftauth=psk
    rightauth=psk
    keyexchange=ikev1
    ike=aes128-sha1-modp1024
    ikelifetime=8h
    esp=aes128-sha1-modp1024
    lifetime=1h
    keyingtries=%forever
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    dpddelay=10s
    dpdtimeout=30s
    dpdaction=restart
    ## Please note the following line assumes you only have two tunnels in your Strongswan
    configuration file. This "mark" value must be unique and may need to be changed based on other entries in
    your configuration file.
    mark=200
    ## Uncomment the following line to utilize the script from the "Automated Tunnel Healthcheck and
    Failover" section. Ensure that the integer after "-m" matches the "mark" value above, and <VPC CIDR> is
    replaced with the CIDR of your VPC
    ## (e.g. 192.168.1.0/24)
    #leftupdown="/etc/ipsec.d/aws-updown.sh -ln Tunnel2 -ll 169.254.246.50/30 -lr 169.254.246.49/30 -
    m 200 -r <VPC CIDR>"

```

Then copy it and uncommment the last line that starts with `leftupdown=...` and update the

<VPC CIDR> This CIDR range is the AWS-VPC not the on-premise-vpc. Which is 10.1.0.0/16

Next, open /etc/ipsec.secrets

```
sudo vim /etc/ipsec.secrets
```

```
4) Create a new file at /etc/ipsec.secrets if it doesn't already exist, and append this line to the file
(be mindful of the spacing!). This value authenticates the tunnel endpoints:
44.223.12.92 3.216.254.210 : PSK "vZwFmB5mCecewKoWIXHDwI7.AYe05eai"
```

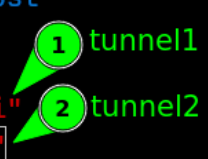
This is the shared secret for the tunnel 1, there is also another shared secret for tunnel 2.

```
4) Create a new file at /etc/ipsec.secrets if it doesn't already exist, and append this line to the file
(be mindful of the spacing!). This value authenticates the tunnel endpoints:
44.223.12.92 34.236.97.73 : PSK "yYXnYEzfZLwzmUd5ofzv.oSMUy3C1ojR"
```

```
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

44.223.12.92 3.216.254.210 : PSK "vZwFmB5mCecewKoWIXHDwI7.AYe05eai"
44.223.12.92 34.236.97.73 : PSK "yYXnYEzfZLwzmUd5ofzv.oSMUy3C1ojR"
~
~
~
~
```



Copy them and add them to the file.

Next, we will create a new file. At /etc/ipsec.d/aws-updown and copy the following code from your configuration file.

```
sudo vim /etc/ipsec.d/aws-updown.sh
```

```

=== HOW-TO ===
1) Create a new file at /etc/ipsec.d/aws-updown.sh if it doesn't already exist, and append the following
script to the file:

#!/bin/bash

while [[ $# > 1 ]]; do
    case ${1} in
        -ln|--link-name)
            TUNNEL_NAME="${2}"
            TUNNEL_PHY_INTERFACE="${PLUTO_INTERFACE}"
            shift
            ;;
        -ll|--link-local)
            TUNNEL_LOCAL_ADDRESS="${2}"
            TUNNEL_LOCAL_ENDPOINT="${PLUTO_ME}"
            shift
            ;;
        -lr|--link-remote)
            TUNNEL_REMOTE_ADDRESS="${2}"
            TUNNEL_REMOTE_ENDPOINT="${PLUTO_PEER}"
            shift
            ;;
        -m|--mark)
            TUNNEL_MARK="${2}"
            shift
            ;;
        -r|--static-route)
            TUNNEL_STATIC_ROUTE="${2}"
    esac
    shift
done

```

Copy the whole code

There is also a minor tweak required on this code. There is a function called `add_route()`

```

add_route() {
    IFS=' ' read -ra route <<< "${TUNNEL_STATIC_ROUTE}"
    for i in "${route[@]"; do
        ip route add ${i} dev ${TUNNEL_NAME} metric ${TUNNEL_MARK} src 10.0.0.87
    done
    iptables -t mangle -A FORWARD -o ${TUNNEL_NAME} -p tcp --tcp-flags SYN,RST SYN -j TCPM
SS --clamp-mss-to-pmtu
    iptables -t mangle -A INPUT -p esp -s ${TUNNEL_REMOTE_ENDPOINT} -d ${TUNNEL_LOCAL_ENDP
OINT} -j MARK --set-xmark ${TUNNEL_MARK}
    ip route flush table 220
}

```

That is my local private ip address on the `on-premise-instance`. After saving the files.

Run `sudo chmod 744 /etc/ipsec.d/aws-updown.sh`

We will need to restart the `ipsec` with the following command.

```
sudo ipsec restart
```

```

ubuntu@ip-10-0-0-87:~$ sudo chmod 744 /etc/ipsec.d/aws-updown.sh
ubuntu@ip-10-0-0-87:~$ sudo ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.9.13 IPsec [starter]...
ubuntu@ip-10-0-0-87:~$

```

Lets check the status of the `ipsec`

```
sudo ipsec status
```

```
ubuntu@ip-10-0-0-87:~$ sudo ipsec status
Security Associations (2 up, 0 connecting):
    Tunnel2[2]: ESTABLISHED 91 seconds ago, 10.0.0.87[44.223.12.92] ... 34.236.97.73[34.236.97.73]
    Tunnel2{1}:  INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: cbfe0d3c_i cd5f38cb_o
    Tunnel2{1}:  10.0.0.0/16 == 10.1.0.0/16
    Tunnel1[1]: ESTABLISHED 91 seconds ago, 10.0.0.87[44.223.12.92] ... 3.216.254.210[3.216.254.210]
    Tunnel1{2}:  INSTALLED, TUNNEL, reqid 2, ESP in UDP SPIs: caa9f1a2_i cb012ac7_o
    Tunnel1{2}:  10.0.0.0/16 == 10.1.0.0/16
ubuntu@ip-10-0-0-87:~$
```

All the tunnels are established and installed.

Lets check `ifconfig` but first we need to install `net-tools`

```
sudo apt install net-tools
```

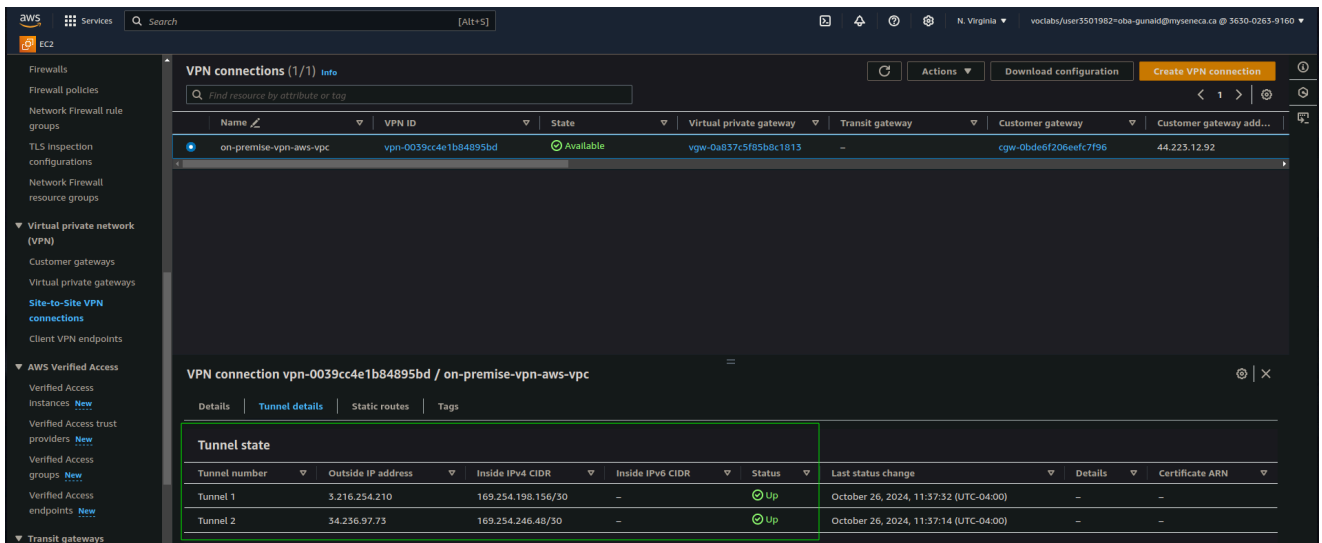
Then execute `ifconfig`

```
ubuntu@ip-10-0-0-87:~$ ifconfig
Tunnel1: flags=209<UP,POINTOPOINT,RUNNING,NOARP> mtu 1419
    inet 169.254.198.158 netmask 255.255.255.252 destination 169.254.198.157
    inet6 fe80::5efe:a00:57 prefixlen 64 scopeid 0x20<link>
    tunnel txqueuelen 1000 (IPIP Tunnel)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Tunnel2: flags=209<UP,POINTOPOINT,RUNNING,NOARP> mtu 1419
    inet 169.254.246.50 netmask 255.255.255.252 destination 169.254.246.49
    inet6 fe80::5efe:a00:57 prefixlen 64 scopeid 0x20<link>
    tunnel txqueuelen 1000 (IPIP Tunnel)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

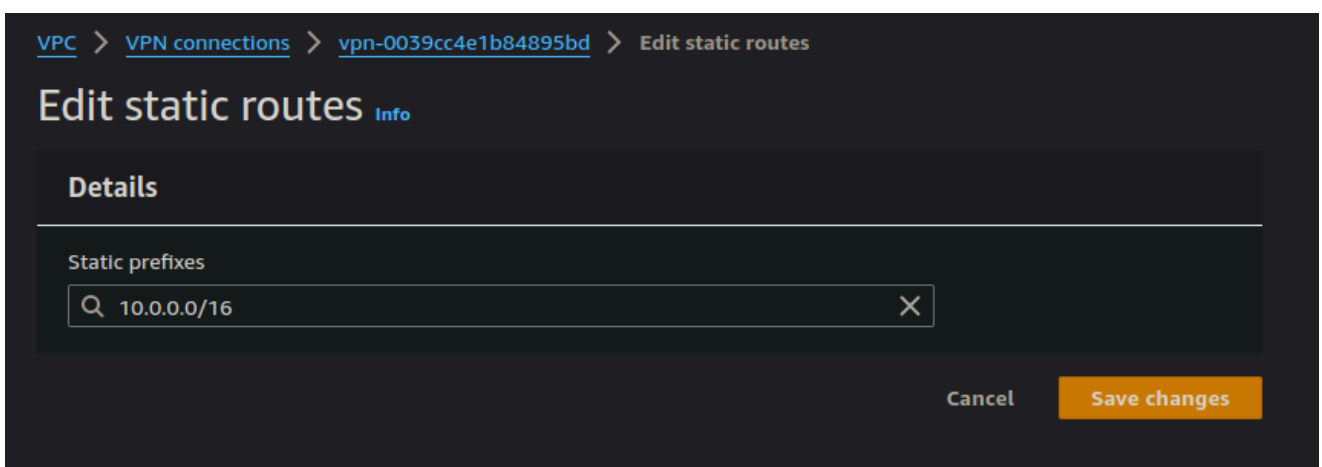
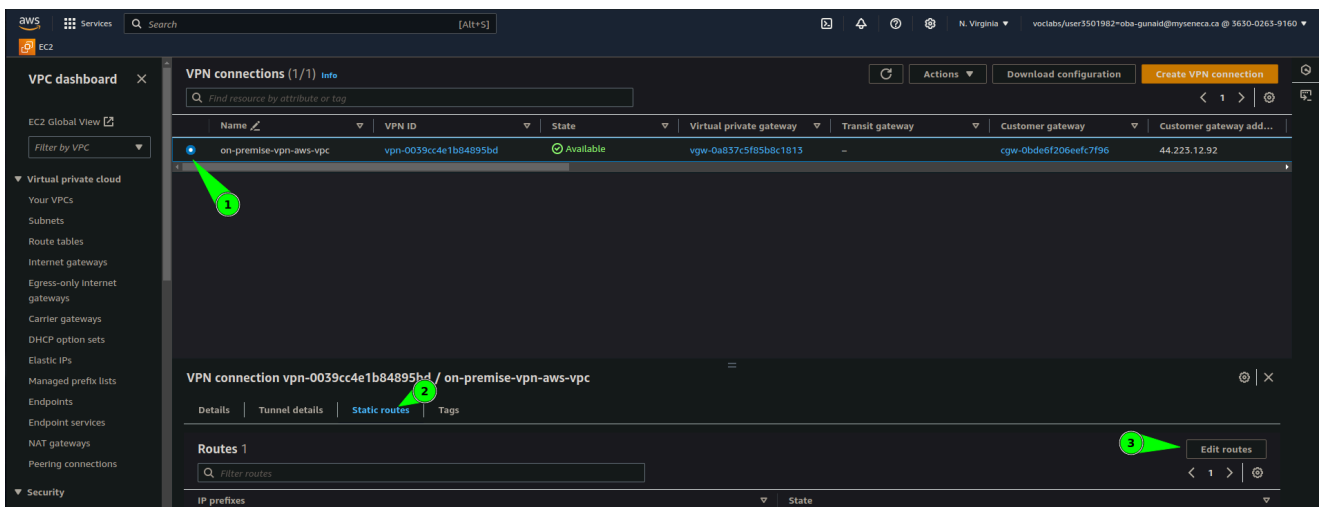
You can see, all the tunnels are up.

Lets check our **Site To Site VPN** and confirm that the tunnels are working.



The last part (static route & ping)

Before we ping the private aws-instance using its private ip address, we will need to add the `on-premise-instance` private IP address or its VPC CIDR block, in the static routes of the Site To Site VPN we created.



You can set the whole range of the on-premise-vpc, or just a specific IP address i.e the on-premise-instance.

Edit static routes Info

Details

Static prefixes

Q 10.0.0.87/32 X

Cancel Save changes

[VPC](#) > [VPN connections](#) > [vpn-0039cc4e1b84895bd](#) > **Edit static routes**

Edit static routes Info

Details

Static prefixes

Q 10.0.0.0/16 X

Cancel Save changes

You can add one of these two.

⚠ Don't forget to click on it from the drop-down list and save changes.

Edit static routes Info

Details

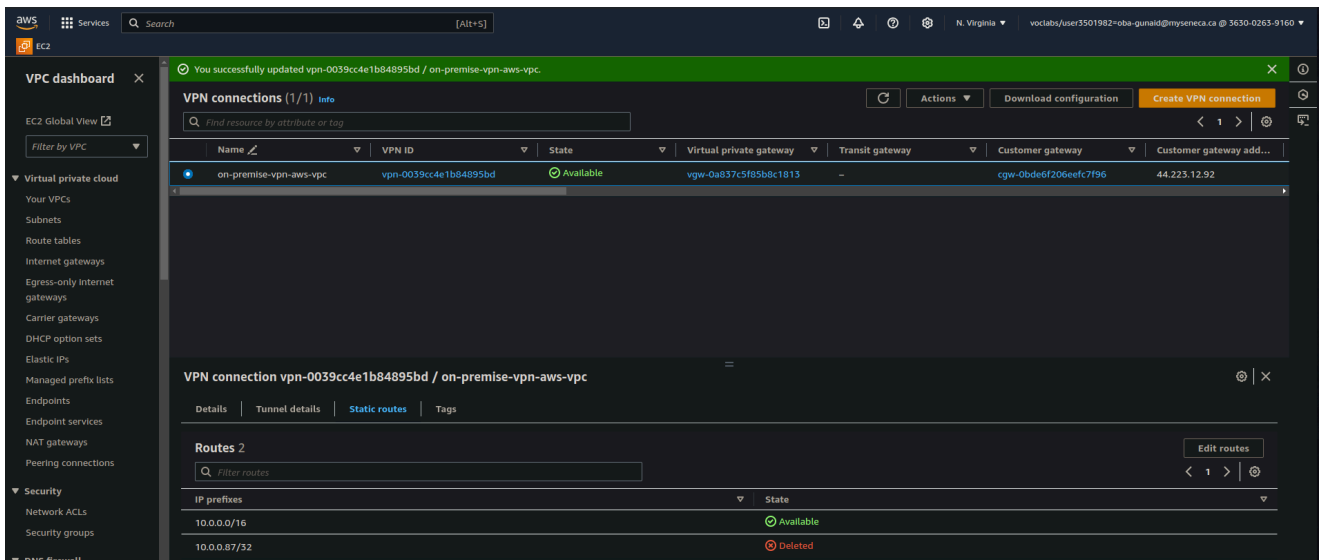
Static prefixes

Q 10.0.0.87/32 X

Use: 10.0.0.87/32

10.0.0.87/32 Cancel Save changes


1 2 3



For testing

I have tested it in both ways, from a specific range i.e 10.0.0.0/16 (vpc-on-premise), and from a specific private ip address of an instance 10.0.0.87/32 (on-premise-instance).

Ping from 10.0.0.87 (on-premise-instance) to 10.1.0.179 (aws-instance)

```
ubuntu@ip-10-0-0-87:~$ ping 10.1.0.179 
PING 10.1.0.179 (10.1.0.179) 56(84) bytes of data.
64 bytes from 10.1.0.179: icmp_seq=1 ttl=127 time=2.13 ms
64 bytes from 10.1.0.179: icmp_seq=2 ttl=127 time=2.08 ms
64 bytes from 10.1.0.179: icmp_seq=3 ttl=127 time=2.36 ms
64 bytes from 10.1.0.179: icmp_seq=4 ttl=127 time=2.18 ms
64 bytes from 10.1.0.179: icmp_seq=5 ttl=127 time=2.70 ms
64 bytes from 10.1.0.179: icmp_seq=6 ttl=127 time=2.36 ms
64 bytes from 10.1.0.179: icmp_seq=7 ttl=127 time=2.11 ms
64 bytes from 10.1.0.179: icmp_seq=8 ttl=127 time=2.16 ms
64 bytes from 10.1.0.179: icmp_seq=9 ttl=127 time=1.85 ms
64 bytes from 10.1.0.179: icmp_seq=10 ttl=127 time=2.36 ms
^C
--- 10.1.0.179 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 1.850/2.226/2.697/0.217 ms
```

With a few additional modifications to the AWS VPC, we can add more public and private subnets and possibly another bastion host or multiple instances. We can also SSH into any AWS instance using only its private IP address, but we must enable SSH access in the security groups.

Additionally, it's important to note that, since we included the entire on-premises CIDR range in the Site-to-Site VPN configuration *static IPs*, any instance within that on-premises CIDR block can securely access resources within the AWS VPC using their private ip addresses, as long as appropriate routing and security group rules are configured.

Thank you,
Omar BaGunaid