

Peering multiple VPCs using Transit Gateway

By Chisom Uketui

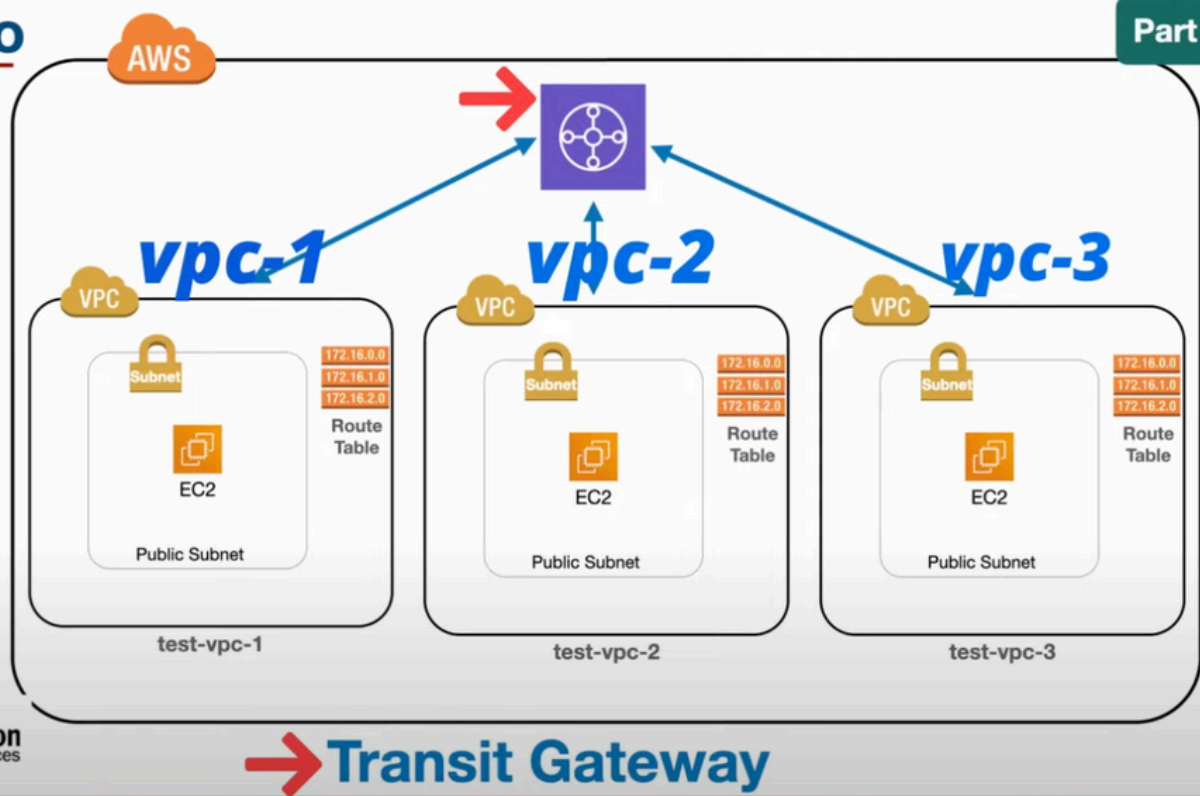
Introducing Today's Project!

Today I am going to showcase how to utilise a Transit gateway to network multiple VPCs present inside an AWS account to be able to communicate with one another. First we will create 3 different VPCs, each would have a public subnet and inside each subnet we will spin up an EC2 instance. We will then set up a Transit gateway that will be attached to all three VPCs and thereafter perform a test where we access one of the Ec2 instance in one VPC and try to access the another Ec2 instance running into another VPC so that we know our Transit Gateway is working.

What is a Transit Gateway?

- **Definition:** AWS Transit Gateway is a networking service that connects multiple Virtual Private Clouds (VPCs) and on-premises networks(VPNs) via a central hub.
- **Importance:**
 - Simplifies network management by eliminating complex peering relationships.
 - Acts as a scalable and efficient solution for connecting multiple VPCs across regions.

Below is an overview of what I will be doing in this project.



In the first part of my project...

Step 1 – Set up my VPCs

In this step, I am going to create 3 VPCs, test- VPC-1, test-VPC-2 and test-VPC-3.

Will start by creating the first VPC, subnet and internet gateway.

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

test-vpc-1

IPv4 CIDR block [Info](#)

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

IPv4 CIDR

12.0.0.0/16

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

IPv6 CIDR block [Info](#)

- No IPv6 CIDR block
- IPAM-allocated IPv6 CIDR block
- Amazon-provided IPv6 CIDR block
- IPv6 CIDR owned by me

Tenancy [Info](#)

Default

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	
Name	test-vpc-1	Remove tag

✔ The following internet gateway was created: igw-099a983e14d178c10 - igw-test-vpc-1. You can now attach to a VPC to enable the VPC to communicate with the internet. [Attach to a VPC](#) ✕

VPC > Internet gateways > igw-099a983e14d178c10

igw-099a983e14d178c10 / igw-test-vpc-1

Actions ▾

Details [Info](#)

Internet gateway ID	State	VPC ID	Owner
igw-099a983e14d178c10	Detached	-	242396018804

Tags

Manage tags

Search tags

< 1 > ⚙

Key	Value
Name	igw-test-vpc-1

Attach the internet gateway to our VPC.

VPC > Internet gateways > Attach to VPC (igw-099a983e14d178c10)

Attach to VPC (igw-099a983e14d178c10) [Info](#)

VPC
Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs
Attach the internet gateway to this VPC.

 ✕

Create a subnet and attach our internet gateway to it to make it a public subnet

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

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.

 ▼

IPv4 VPC CIDR block [Info](#)

Choose the IPv4 VPC CIDR block to create a subnet in.

 ▼

IPv4 subnet CIDR block

 256 IPs

< > ^ v

▼ **Tags - optional**

Key

 ✕

Value - optional

 ✕

Remove

Create a route table that will direct traffic to our subnet and then associate this route table with our subnet.

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

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

VPC
The VPC to use for this route table.

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"/>	<input type="text" value="rt-test-vpc-1"/>	<input type="button" value="Remove"/>

You can add 49 more tags.

Edit routes to allow public access to our subnet, adding the internet gateway as target.

Edit routes

Destination	Target	Status
12.0.0.0/16	<input type="text" value="local"/> <input type="text" value="local"/>	✔ Active
<input type="text" value="0.0.0.0/0"/>	<input type="text" value="Internet Gateway"/> <input type="text" value="igw-099a983e14d178c10"/>	-

^Create our resource - EC2 instance.

The screenshot displays the AWS Management Console interface for configuring an EC2 instance. On the left, the 'Inbound Security Group Rules' section shows two rules. Rule 1 is for SSH (Type: ssh, Protocol: TCP, Port range: 22, Source type: Anywhere, Source: 0.0.0.0/0, Description: e.g. SSH for admin desktop). Rule 2 is for HTTP (Type: HTTP, Protocol: TCP, Port range: 80, Source type: Custom, Source: 0.0.0.0/0, Description: e.g. SSH for admin desktop). A yellow warning box at the bottom of the rules section states: '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.' On the right, the 'Summary' section shows the instance configuration: Number of instances: 1, Software image (AMI): Canonical, Ubuntu, 22.04 LTS, Virtual server type (instance type): t2.micro, Firewall (security group): New security group, Storage (volumes): 1 volume(s) - 8 GiB. A blue information box at the bottom of the summary states: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on...'. At the bottom right, there are 'Cancel' and 'Launch instance' buttons.

Here, we are adding a script in the user data for installing the Apache web server. The user data section is used to install certain packages onto your Ec2 instance when setting it up for the first time.

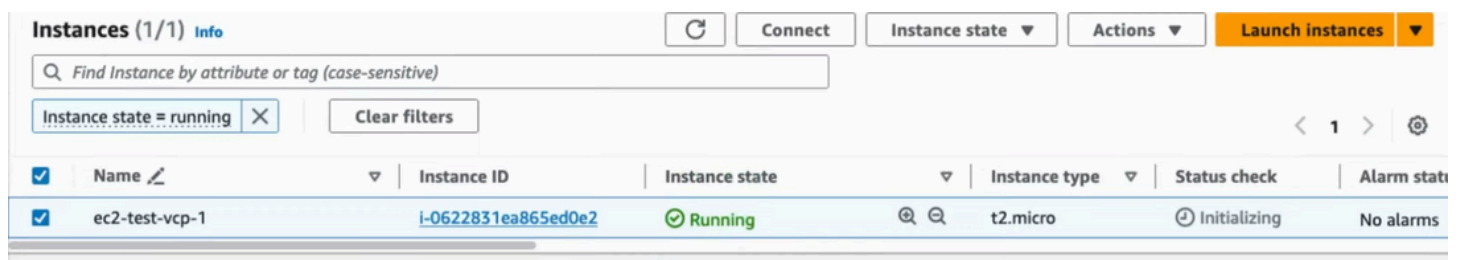
User data - optional [Info](#)

Upload a file with your user data or enter it in the field.

 Choose file

```
#!/bin/bash
yes | sudo apt update
yes | sudo apt install apache2
echo "<h1>Server Details</h1><p><strong>Hostname:</strong> $(hostname)
</p><p><strong>IP Address:</strong> $(hostname -I | cut -d" " -f1)</p>" >
/var/www/html/index.html
sudo systemctl restart apache2|
```

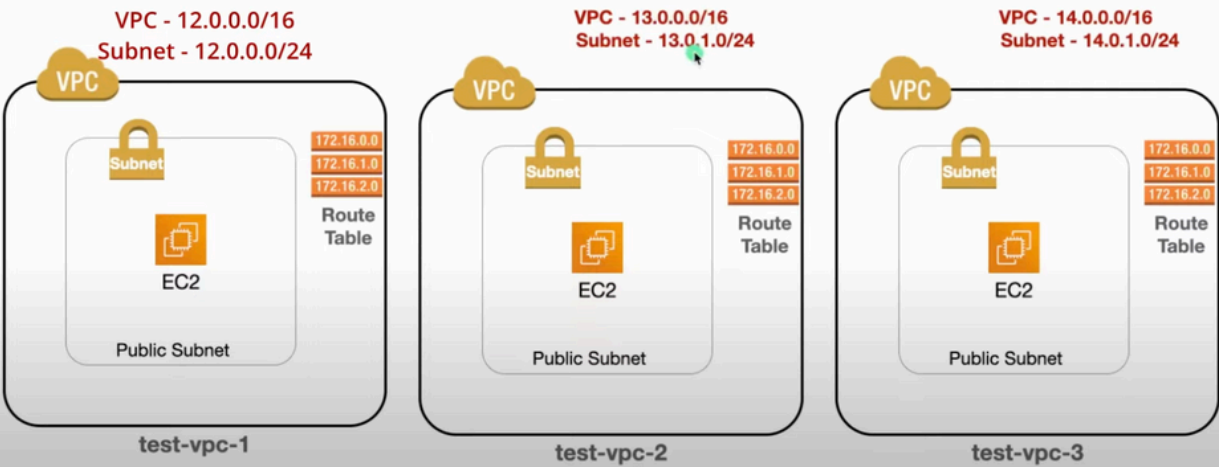
Ec2 instance is running!



The screenshot shows the AWS Management Console 'Instances' page. The instance 'ec2-test-vcv-1' is in a 'Running' state. The table below summarizes the instance details.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
ec2-test-vcv-1	i-0622831ea865ed0e2	Running	t2.micro	Initializing	No alarms

We will then set up the remaining 2 VPCs and subnets using the CIDR range and blocks shown in the diagram below.



Transit gateway

In this step we will go ahead to set up our transit gateway

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

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

Description [Info](#)

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

Configure the transit gateway

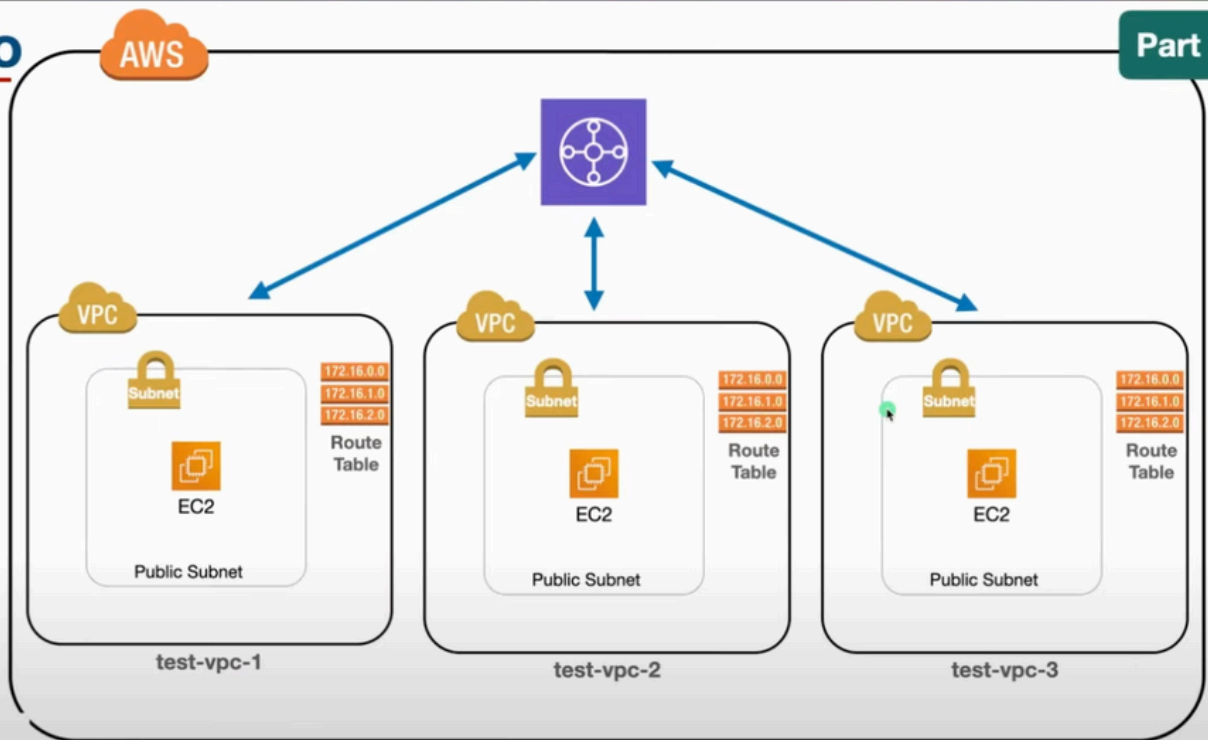
Amazon side Autonomous System Number (ASN) [Info](#)

Transit gateways (1) [Info](#)



<input type="checkbox"/>	Name	Transit gateway ID	Owner ID	State
<input type="checkbox"/>	tg-vpc-1-vpc-2-vpc-3	tgw-00f76b6b5d186a784	242396018804	Available

Transit gateway attachment



Transit Gateway

In this step we are going to attach the transit gateway to our VPCs. Click on the transit gateway attachment button and create attachment based on each VPC (VPC1, VPC2 and VPC3). We will also define the VPC so first attachment we are going to create is towards the VPC1.

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](#)

VPC attachment

Select and configure your VPC attachment.

✓ You successfully created VPC attachment tgw-attach-0943f41c7e0291503 / tg-attachment-vpc-1.

ℹ You can visualize and monitor your Transit Gateway(s) from the [AWS Network Manager](#). Register your Transit Gateway by creating a [global network](#) to get started.

Transit gateway attachments (1) [Info](#)



Actions ▾

Create transit gateway attachment

🔍 Filter transit gateway attachments

< 1 > ⚙️

Transit gateway attachment ID: tgw-attach-0943f41c7e0291503 ✕

Clear filters

<input type="checkbox"/>	Name ▾	Transit gateway attachment ID ▾	Transit gateway ID ▾	Resource type ▾	Resource ID ▾
<input type="checkbox"/>	tg-attachment-vpc-1	tgw-attach-0943f41c7e0291503	tgw-00f76b6b5d186a784	VPC	vpc-0e7e00d591614b47e

Create the second attachment for VPC 2:

You successfully created VPC attachment tgw-attach-0c3390359de1d2525 / tg-attachment-vpc-2.

You can visualize and monitor your Transit Gateway(s) from the [AWS Network Manager](#). Register your Transit Gateway by creating a [global network](#) to get started.

Transit gateway attachments (1/2) Info

Filter transit gateway attachments

Name	Transit gateway attachment ID	Transit gateway ID	Resource type	Resource ID
tg-attachment-vpc-1	tgw-attach-0943f41c7e0291503	tgw-00f76b6b5d186a784	VPC	vpc-0e7e00d591614b47e
tg-attachment-vpc-2	tgw-attach-0c3390359de1d2525	tgw-00f76b6b5d186a784	VPC	vpc-0c682d002ac6dbc6c

And finally for the third VPC:

You successfully created VPC attachment tgw-attach-0938f818ca9e867fc / tg-attachment-vpc-3.

You can visualize and monitor your Transit Gateway(s) from the [AWS Network Manager](#). Register your Transit Gateway by creating a [global network](#) to get started.

Transit gateway attachments (3) Info

Filter transit gateway attachments

Name	Transit gateway attachment ID	Transit gateway ID	Resource type	Resource ID
tg-attachment-vpc-3	tgw-attach-0938f818ca9e867fc	tgw-00f76b6b5d186a784	VPC	vpc-032529420f799a5a8
tg-attachment-vpc-1	tgw-attach-0943f41c7e0291503	tgw-00f76b6b5d186a784	VPC	vpc-0e7e00d591614b47e
tg-attachment-vpc-2	tgw-attach-0c3390359de1d2525	tgw-00f76b6b5d186a784	VPC	vpc-0c682d002ac6dbc6c

Transit Gateway vs VPC Peering

Transit Gateway (TGW) and **VPC Peering** are both AWS networking solutions but serve different purposes:

- **VPC Peering** connects two VPCs directly, allowing traffic to flow between them as if they were part of the same network. It's a one-to-one connection and doesn't scale well if you need to connect multiple VPCs.
- **Transit Gateway** is a centralized hub that connects multiple VPCs, on-premises networks, and other resources. It simplifies large-scale networking by acting as an intermediary, reducing the complexity of managing many individual peering connections.

In summary, **VPC Peering** is best for simpler, smaller setups, while **Transit Gateway** is ideal for larger, more complex network architectures where multiple VPCs or hybrid cloud environments are involved.

Now lets Update our routes:

In this step we are going to update our route table so our subnet knows how to route the request to the transit gateway to enable communication with the other VPCs.

VPC > Route tables > rtb-0bb4f8bb2d72bbb91 > Edit routes

Edit routes

Destination	Target	Status
12.0.0.0/16	local	Active
13.0.0.0/16	Transit Gateway	Active
0.0.0.0/0	Internet Gateway	Active
14.0.0.0/16	Transit Gateway	-

Add route

Now we have our connectivity between VPC 2 and VPC 3 with our VPC 1.

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

Routes (4)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-099a983e14d178c10	Active	No
12.0.0.0/16	local	Active	No
13.0.0.0/16	tgw-00f76b6b5d186a784	Active	No
14.0.0.0/16	tgw-00f76b6b5d186a784	Active	No

Now let's work on VPC 2 route table to create a connectivity with VPC 1 and VPC 3.

Edit routes

Destination	Target	Status	Propagated	
13.0.0.0/16	local	Active	No	
<input type="text" value="0.0.0.0/0"/>	Internet Gateway	Active	No	<input type="button" value="Remove"/>
<input type="text" value="12.0.0.0/16"/>	Transit Gateway	-	No	<input type="button" value="Remove"/>
<input type="text" value="14.0.0.0/16"/>	Transit Gateway	-	No	<input type="button" value="Remove"/>

Routes (4)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-059e75cd41544fbd3	Active	No
12.0.0.0/16	tgw-00f76b6b5d186a784	Active	No
13.0.0.0/16	local	Active	No
14.0.0.0/16	tgw-00f76b6b5d186a784	Active	No

Now let's work on VPC 3 route table to create a connectivity with VPC 1 and VPC 2.

Destination	Target	Status	Propagated	
14.0.0.0/16	local	Active	No	
<input type="text" value="0.0.0.0/0"/>	Internet Gateway	Active	No	<input type="button" value="Remove"/>
<input type="text" value="12.0.0.0/16"/>	Transit Gateway	-	No	<input type="button" value="Remove"/>
<input type="text" value="13.0.0.0/16"/>	Transit Gateway	-	No	<input type="button" value="Remove"/>

Routes	Subnet associations	Edge associations	Route propagation	Tags
Routes (4)				Both ▼
Filter routes				
Destination	Target	Status	Propagated	
0.0.0.0/0	igw-04be70f7412bd57a5	Active	No	
12.0.0.0/16	tgw-00f76b6b5d186a784	Active	No	
13.0.0.0/16	tgw-00f76b6b5d186a784	Active	No	
14.0.0.0/16	local	Active	No	

Test VPC peering

In this step, we will SSH into our Ec2 instances and try to curl the pages of our Apache homepages in our Ec2.

Connect to instance Info

Connect to your instance `i-0622831ea865ed0e2` (ec2-test-vcp-1) using any of these options

- EC2 Instance Connect
- Session Manager
- SSH client**
- EC2 serial console

Instance ID
[i-0622831ea865ed0e2](#) (ec2-test-vcp-1)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `ec2-test-vcp-1-key.pem`
3. Run this command, if necessary, to ensure your key is not publicly viewable.


```
chmod 400 ec2-test-vcp-1-key.pem
```
4. Connect to your instance using its Public IP:

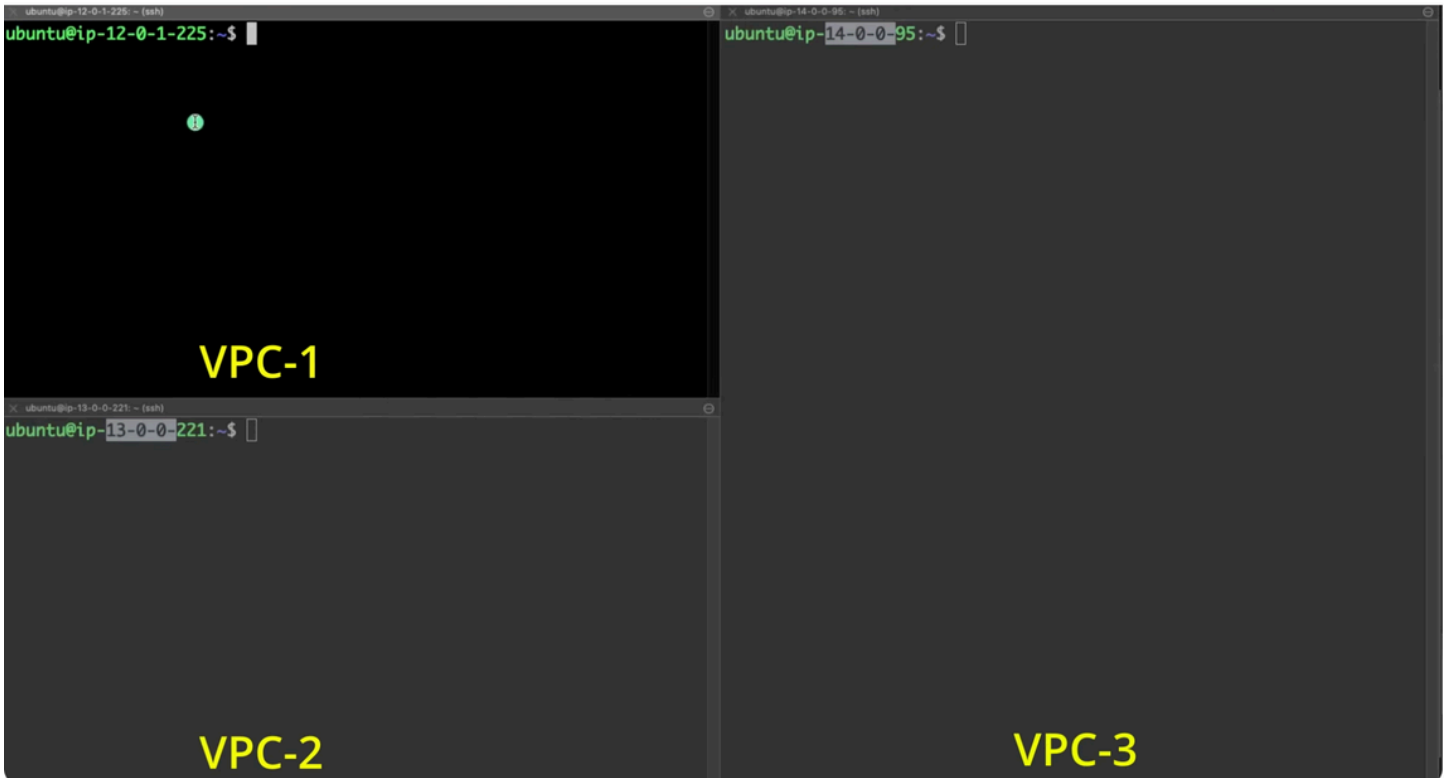

```
18.153.95.203
```

Example:

```
ssh -i "ec2-test-vcp-1-key.pem" ubuntu@18.153.95.203
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Note the IP addresses shown below. These are IPs of VPC 1, VPC 2 and VPC 3.



Below shows we are able to access the Apache page in the different instances deployed on separate VPCs using a curl command. From VPC 1, we were able to access Apaches page in VPC 2 and VPC 3 and vice versa. This shows all three VPCs have been successfully peered together using a transit gateway/transit gateway attachment and they are able to communicate with one another!!! 😊

