

PROJECT

Project Title : Multi-Region VPC Connectivity Using AWS Transit Gateway

Date : 20/10/2025

Name of the person : Gadi Thirupathi

Name of the project : Multi-Region VPC Connectivity Using AWS Transit Gateway

Objective of the project : To create a scalable, secure, and highly available network architecture that connects VPCs across different AWS regions using Transit Gateway inter-region peering.

Steps Done in the project :

step1:

==>

Create 2 VPCs (e.g., VPC-A1, VPC-A2 in us-east-1 and ap-south-1)

Each with:

- CIDR: /16 (e.g., 10.3.0.0/16, 10.4.0.0/16, etc.)
- 2 Subnets: In each vpcs
- Internet Gateway attached
- Route tables configured

step2:

==>

Launch Ec2 instances on VPC-A1 and VPC-A2 give the name tag Machine01 and Machine02

step3:

- Login in to the Ec2 instances
- Use the command #ping<pvip>
- we are unable to see the connection between two machines.

step4:

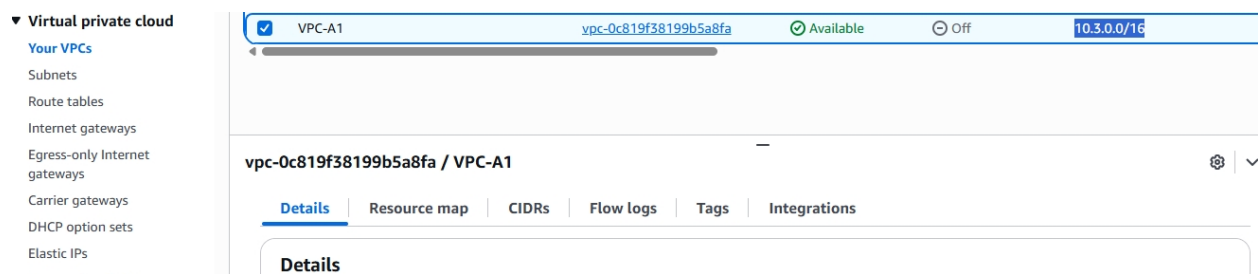
- create the transit gateway and transit attachments in the both regions
- create vpc transit attachment and peer attachment on both regions
- After creating configure in the route tables of VPCs.
- configure the transit route table in both regions with remote vpc cidr ip's

Output screenshots :

creating of two vpcs and machines screenshots:

==>

VPC-A1 And Ec2intance machine on region-01 us-east-1:



EC2 > Instances

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Machine01	i-01067c9df5eb4e87a	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a

Unselect instance: Machine01

i-01067c9df5eb4e87a (Machine01)

==>

VPC-A2 And Ec2 machine on region02 ap-south-1:

VPC > Your VPCs

VPC dashboard

AWS Global View

Filter by VPC:

Virtual private cloud

Your VPCs

Subnets

Route tables

Internet gateways

Egress-only Internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Your VPCs (1/2) Info

Find VPCs by attribute or tag

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
-	vpc-050ed719fcd8d45fa	Available	Off	172.31.0.0/16	-
VPC-A2	vpc-0ddd76cdd1e393ee1	Available	Off	10.4.0.0/16	-

vpc-0ddd76cdd1e393ee1 / VPC-A2

Details Resource map CIDRs Flow logs Tags Integrations

Details

EC2 > Instances

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Machine02	i-0a5988761f4e9b1b0	Running	t3.micro	Initializing	View alarms +	ap-south-1a

i-0a5988761f4e9b1b0 (Machine02)

Details Status and alarms Monitoring Security Networking Storage Tags

Checking the connection between two machines with their private ip's:

machine01:

```
root@ip-10-3-1-120: ~  
root@ip-10-3-1-120:~# ping 10.4.1.75  
PING 10.4.1.75 (10.4.1.75) 56(84) bytes of data.
```

machine02:

```
root@ip-10-4-1-75: ~  
root@ip-10-4-1-75:~# ping 10.3.1.120  
PING 10.3.1.120 (10.3.1.120) 56(84) bytes of data.
```

Note: Here we didn't have connection between two machines.

Create the transit gateway and vpc,peer transit attachments:

Region01:

The screenshot shows the AWS Management Console interface for the 'United States (N. Virginia)' region. The left-hand navigation pane is open, showing the 'VPC' section with 'Transit gateway attachments' selected. A green notification banner at the top states: 'You successfully created peering attachment tgw-attach-0b948fd43fa5f35b8 / transit-peer-connection01.' Below this, the 'Transit gateway attachments (2)' section displays a table with two entries:

	Name	Transit gateway attachment ID	Transit gateway ID	State	Resource
<input type="checkbox"/>	transit-peer-connecti...	tgw-attach-0b948fd43fa5f35b8	tgw-07e8b41dc68ff3c5e	Available	Peering
<input type="checkbox"/>	Transit-vpc-attachm...	tgw-attach-06104cb461078d527	tgw-07e8b41dc68ff3c5e	Available	VPC

Below the table, there is a section titled 'Select a transit gateway attachment' with a search bar and a settings icon.

Region02:

The screenshot shows the AWS Management Console for Region02 (Asia Pacific (Mumbai)). The left sidebar shows the navigation menu with 'VPC' selected. The main content area displays 'Transit gateway attachments (2)'. A green notification bar at the top states: 'Accept transit gateway peering attachment(tgw-attach-0b948fd43fa5f35b8) succeeded.' Below this, a table lists the attachments:

Name	Transit gateway attachment ID	Transit gateway ID	State	Resource	Resource
peer-connection02	tgw-attach-0b948fd43fa5f35b8	tgw-0b0c998afe7c43408	Available	Peering	tgw-0b0c998afe7c43408
vpc02-transit-attach...	tgw-attach-04d7f40429c9895fa	tgw-0b0c998afe7c43408	Available	VPC	vpc-04d7f40429c9895fa

Below the table, the details for the attachment 'tgw-attach-0b948fd43fa5f35b8 / peer-connection02' are shown, with tabs for 'Details', 'Flow logs', and 'Tags'.

Configure the route tables in the both regions:

Region01:

The screenshot shows the AWS Management Console for Region01, displaying the 'Edit routes' page for route table 'rtb-004683c8b493d336c'. The table lists the following routes:

Destination	Target	Status	Propagated	Route Origin	Actions
10.3.0.0/16	local	Active	No	CreateRouteTable	
10.4.0.0/16	Transit Gateway	Active	No	CreateRoute	Remove
0.0.0.0/0	Internet Gateway	Active	No	CreateRoute	Remove

An 'Add route' button is located at the bottom left of the table.

Region02:

The screenshot shows the AWS Management Console for Region02, displaying the 'Edit routes' page for route table 'rtb-004683c8b493d336c'. The table lists the following routes:

Destination	Target	Status	Propagated	Route Origin	Actions
10.4.0.0/16	local	Active	No	CreateRouteTable	
10.3.0.0/16	Transit Gateway	Active	No	CreateRoute	Remove
0.0.0.0/0	Internet Gateway	Active	No	CreateRoute	Remove

An 'Add route' button is located at the bottom left of the table.

Here we can have the connection between two vpcs in different regions :

machine01 :

```
root@ip-10-3-1-120: ~# ping 10.4.1.75
PING 10.4.1.75 (10.4.1.75) 56(84) bytes of data.
64 bytes from 10.4.1.75: icmp_seq=1 ttl=61 time=188 ms
64 bytes from 10.4.1.75: icmp_seq=2 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=3 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=4 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=5 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=6 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=7 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=8 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=9 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=10 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=11 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=12 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=13 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=14 ttl=61 time=186 ms
64 bytes from 10.4.1.75: icmp_seq=15 ttl=61 time=186 ms
^Z
[1]+  Stopped                  ping 10.4.1.75
root@ip-10-3-1-120: ~#
```

machine02 :

```
root@ip-10-3-1-120: ~# ping 10.3.1.120
PING 10.3.1.120 (10.3.1.120) 56(84) bytes of data.
64 bytes from 10.3.1.120: icmp_seq=1 ttl=61 time=187 ms
64 bytes from 10.3.1.120: icmp_seq=2 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=3 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=4 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=5 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=6 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=7 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=8 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=9 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=10 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=11 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=12 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=13 ttl=61 time=185 ms
64 bytes from 10.3.1.120: icmp_seq=14 ttl=61 time=185 ms
^Z
[1]+  Stopped                  ping 10.3.1.120
root@ip-10-4-1-75: ~#
```

Tools and services used :

- AWS VPC
- AWS EC2
- AWS TRANSIT GATEWAY AND ATTACHMENTS
- COMMAND PROMT

Conclusion:

This project successfully demonstrated how AWS Transit Gateway (TGW) can be used to build a scalable, secure, and efficient multi-region network architecture. By deploying Transit Gateways in different AWS regions and creating inter-region peering attachments, we were able to establish seamless connectivity between multiple VPCs across geographically separated locations.