

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:

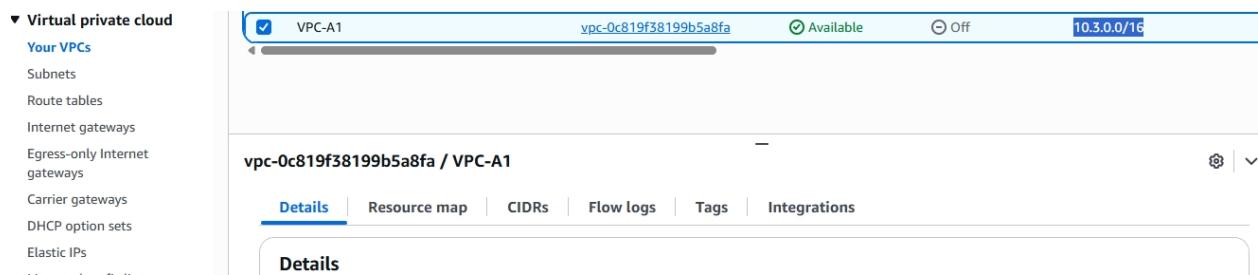
- create the transit gateway and transit attachments in the both regions
- create vpc transit attachment and peer attachment on both regions
- After creating configure in the route tables of VCPGs.
- configure the transit route table in both regions with remote vpc cidr ip's

Output screenshots :

creating of two vpcs and machines screenshots:

==>

VPC-A1 And Ec2instance machine on region-01 us-east-1:



EC2 > Instances

Instances (1/1) Info

| 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 |

i-01067c9df5eb4e87a (Machine01)

==>

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

VPC > Your VPCs

Your VPCs (1/2) Info

| Name | VPC ID | State | Block Public... | IPv4 CIDR |
|---------------|-----------------------------|------------------|-----------------|--------------------|
| - | vpc-050ed719frd8d45fa | Available | Off | 172.31.0.0/16 |
| VPC-A2 | vpc-0ddd76cd1e393ee1 | Available | Off | 10.4.0.0/16 |

vpc-0ddd76cd1e393ee1 / VPC-A2

Details | Resource map | CIDs | Flow logs | Tags | Integrations

Details

EC2 > Instances

Instances (1/1) Info

| 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:~# ping 10.4.1.75
PING 10.4.1.75 (10.4.1.75) 56(84) bytes of data.
```

machine02:

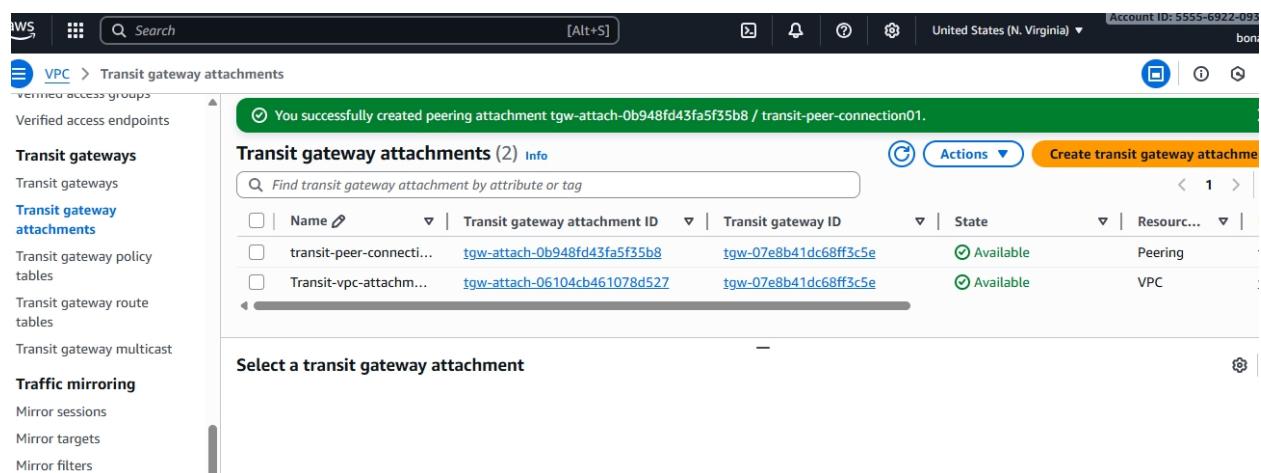


```
root@ip-10-4-1-75:~# ping 10.3.1.120
PING 10.3.1.120 (10.3.1.120) 56(84) bytes of data.
```

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 VPC console with the 'Transit gateway attachments' page. A success message at the top states: "You successfully created peering attachment tgw-attach-0b948fd43fa5f35b8 / transit-peer-connection01." Below this, a table lists two transit gateway attachments:

| Name | Transit gateway attachment ID | Transit gateway ID | State | Resource |
|-------------------------|--|---------------------------------------|-----------|----------|
| transit-peer-connect... | tgw-attach-0b948fd43fa5f35b8 | tgw-07e8b41dc68ff3c5e | Available | Peering |
| Transit-vpc-attachm... | tgw-attach-06104cb461078d527 | tgw-07e8b41dc68ff3c5e | Available | VPC |

Below the table, a section titled "Select a transit gateway attachment" is visible.

Region02:

The screenshot shows the AWS VPC console with the 'Transit gateway attachments' page. A success message at the top indicates 'Accept transit gateway peering attachment(tgw-attach-0b948fd43fa5f35b8) succeeded.' The main table lists two entries:

| Name | Transit gateway attachment ID | Transit gateway ID | Status | Resource Type | Region |
|-------------------------|-------------------------------|-----------------------|-----------|---------------|--------|
| peer-connection02 | tgw-attach-0b948fd43fa5f35b8 | tgw-0b0c998afe7c43408 | Available | Peering | tgw-l |
| vpc02-transit-attach... | tgw-attach-04d7f40429c9895fa | tgw-0b0c998afe7c43408 | Available | VPC | vpc-f |

Configure the route tables in the both regions:

Region01:

The screenshot shows the 'Edit routes' section of the AWS VPC Route tables page. It displays three routes:

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

Region02:

The screenshot shows the 'Edit routes' section of the AWS VPC Route tables page. It displays three routes:

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

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-10-4-1-120:~# 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-10-4-1-120:~#
```

Tools ans 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.