AWS Use Case – 1

Cost Optimization - Centralized outbound routing to the internet using AWS Transit Gateway

Managing and securing external connectivity can be challenging and expensive when an organization's workload is split between many isolated VPC environments and accounts.

Let's consider a use case where an organization has vpc-a, vpc-b and vpc-c shared workload deployed on private subnets in 3 isolated AWS VPC environments. All of these workloads must be able to fetch data from the public internet.

Apart from adding some management overhead and complexity, the main downside here is the cost of this solution.

Here we mitigated the issue by centralized outbound routing setup using AWS Transit Gateway (TGW) offers several benefits for cost optimization and other. Here are some key points:

1. **Reduced NAT Gateway Costs**

Single NAT Gateway: By centralizing the NAT Gateway in VPC C, you only need to provision one NAT Gateway, which serves multiple VPCs. This reduces the cost compared to deploying separate NAT Gateways in each VPC.

1. **Optimized Data Transfer Costs**

**Consolidated Egress Points**: All VPCs route their internet-bound traffic through the centralized NAT Gateway. This can reduce data transfer costs because inter-VPC traffic over a TGW is typically less expensive than multiple VPC-to-internet data transfers.

**Free Intra-Region Transit Gateway Data Transfer**: Intra-region data transfer between VPCs through TGW is free, which helps in reducing costs associated with data transfers between VPCs located in the same region.

1. **Reduced Operational Overhead:**

Managing a single NAT Gateway and a centralized TGW setup is simpler and less time-consuming compared to managing multiple NAT Gateways across different VPCs.

1. **Scalability and Flexibility**

**Scalable Architecture**: A centralized NAT Gateway setup can easily scale to handle increased traffic from multiple VPCs without the need to modify the setup in each individual VPC.

1. **Centralized Security and Compliance**

**Unified Security Policies**: Implementing security policies in a single place (VPC C) can help in maintaining compliance and ensuring that all VPCs adhere to the same security standards, reducing the risk of configuration drift and potential security incidents.

**Hands On:**

**Reference:**

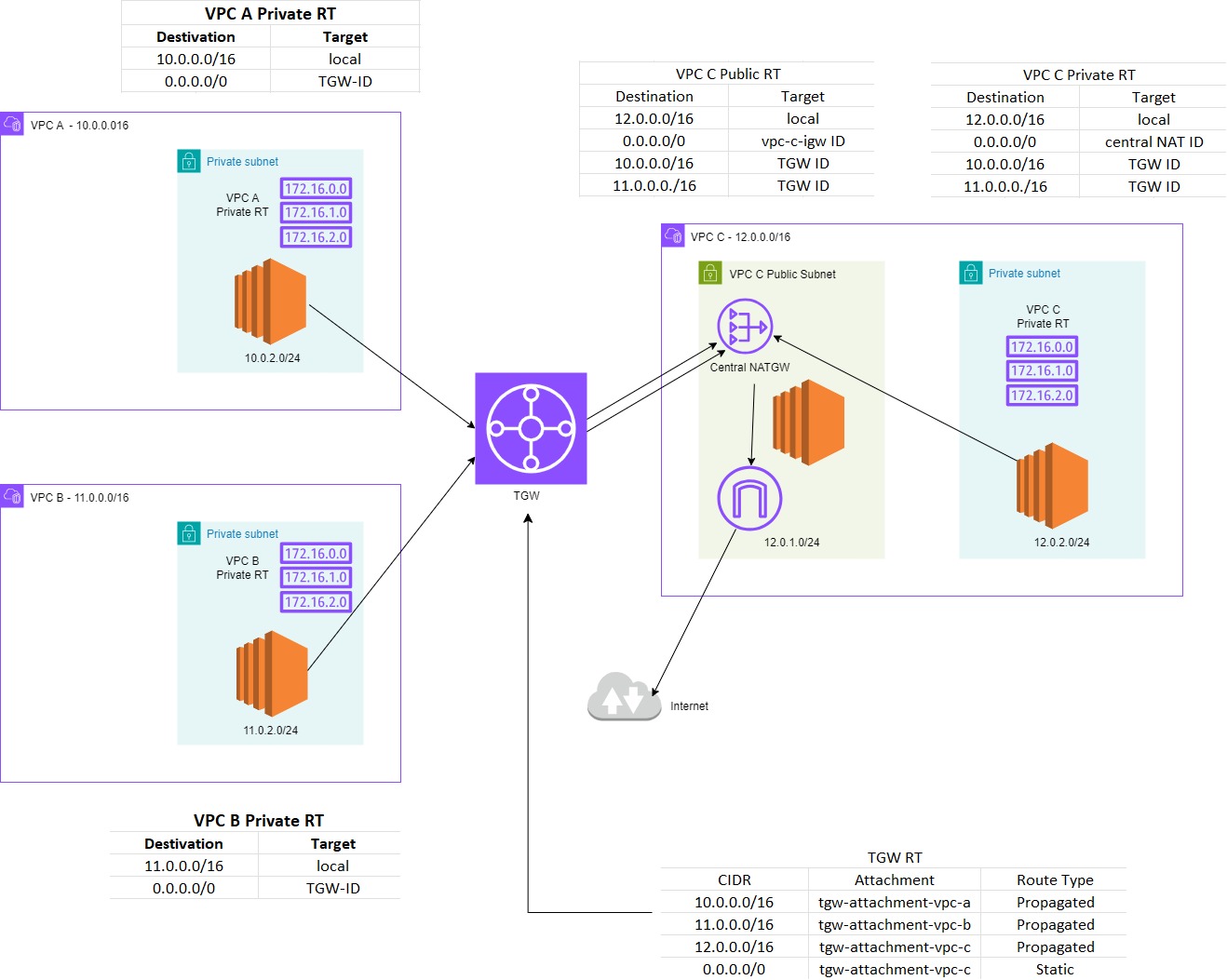
[**https://docs.aws.amazon.com/vpc/latest/tgw/transit-gateway-nat-igw.html**](https://docs.aws.amazon.com/vpc/latest/tgw/transit-gateway-nat-igw.html)

You can configure a transit gateway to route outbound internet traffic from a VPC without an internet gateway to a VPC that contains a NAT gateway and an internet gateway.

The following diagram shows the key components of the configuration for this scenario. You have applications in VPC A and VPC B that need outbound only internet access. You configure VPC C with a public NAT gateway and an internet gateway, and a private subnet for the VPC attachment. Connect all VPCs to a transit gateway. Configure routing so that outbound internet traffic from VPC A and VPC B traverses the transit gateway to VPC C. The NAT gateway in VPC C routes the traffic to the internet gateway.

***CIDR blocks for each VPC propagate to the transit gateway route table. For VPC C, you must create the attachment using the private subnet. If you create the attachment using the public subnet, the instance traffic is routed to the internet gateway, but the internet gateway drops the traffic because the instances don't have public IP addresses. By placing the attachment in the private subnet, the traffic is routed to the NAT gateway, and the NAT gateway sends the traffic to the internet gateway using its Elastic IP address as the source IP address.***

**Architecture Diagram**



**Configuration Steps:**

**1. Set Up the VPC’s**

* **Create VPC-A** with CIDR block 10.0.0.0/16

Create a private subnet 10.0.2.0/24 (vpc-a-private-subnet)

* **Create VPC-B** with CIDR block 11.0.0.0/16

Create a private subnet 11.0.2.0/24(vpc-b-private-subnet)

* Launch an ec2 instances (vpc-a-private-ec2 and vpc-b-private-ec2) in private subnets of vpc-a and vpc-b, with a security group with inbound rules as:
* Allow SSH: from CIDR block of vpc-c (consider ec2 instance in vpc-c public subnet as bastion host for both instances in private subnet of vpc-a and vpc-b)
* ICMP ALL IPV4: from CIDR block other vpc’s
* **Create VPC-C** with CIDR block 12.0.0.0/16

Create a public subnet 12.0.1.0/24 (vpc-c-public-subnet)

Create a private subnet 12.0.2.0/24(vpc-c-private-subnet)

* Launch an ec2 instance (vpc-c-publis-ec2 and vpc-c-private-ec2) in private and public subnet of vpc-c, with a security group with inbound rules as:
* Allow SSH: from anywhere ipv4 for ec2 in public subnet.

from CIDR block of vpc-c for ec2 in private subnet.

* ICMP ALL IPV4: from CIDR blocks other vpc’s
* Create and attach an Internet Gateway (vpc-c-igw) to the public subnet of vpc-c.

**2. Create and Attach the TGW**

**Create Transit Gateway**

* In the navigation pane, click on Transit Gateways in VPC Dashboard.
* Click on - Create Transit Gateway.
* Fill in the required details:
* Name tag: TGW
* Description: Centralized TGW for outbound routing
* Amazon side ASN: leave as default state
* Auto-accept shared attachments: Enable
* Default route table association: Enable
* Default route table propagation: Enable
* Click Create Transit Gateway.

**Attach VPC A, VPC B, and VPC C to the TGW**

* Click on Create Transit Gateway Attachment in VPC Dashboard.
* Transit Gateway ID: Select the TGW you created
* Attachment type: Select VPC.
* VPC ID: Select VPC A (vpc-a).
* Subnet IDs: Select the private subnet in VPC A (10.0.2.0/24)
* Name tag: (TGW-vpc-a-attachment)
* Click Create Transit Gateway Attachment.

Repeat the above steps for vpc-b and vpc-c.

***Ensure you attach the private subnets of VPC A and VPC B***

***Attach both the public and private subnets of VPC C***

**3. Configure VPC Route Tables**

* VPC A

|  |  |
| --- | --- |
| **vpc-a-private-rt** | |
| Destination | Target |
| 10.0.0.0/16 | local |
| 0.0.0.0/16 | TGW -ID |

* VPC B

|  |  |
| --- | --- |
| **vpc-b-private-rt** | |
| Destination | Target |
| 11.0.0.0/16 | local |
| 0.0.0.0/16 | TGW -ID |

* VPC C

|  |  |
| --- | --- |
| **vpc-c-private-rt** | |
| Destination | Target |
| 12.0.0.0/16 | local |
| 0.0.0.0/16 | Central-NAT ID |
| 10.0.0.0/16 | TGW - ID |
| 11.0.0.0/16 | TGW - ID |

|  |  |
| --- | --- |
| **vpc-c-public-rt** | |
| Destination | Target |
| 12.0.0.0/16 | local |
| 0.0.0.0/16 | vpc-c-igw ID |
| 10.0.0.0/16 | TGW - ID |
| 11.0.0.0/16 | TGW - ID |

**4. Configure the TGW Route Table**

|  |  |  |
| --- | --- | --- |
| **TGW-RT** | | |
| **CIDR** | **Attachment** | **Route Type** |
| 10.0.0.0/16 | tgw-attachment-vpc-a | Propagated |
| 11.0.0.0/16 | tgw-attachment-vpc-b | Propagated |
| 12.0.0.0/16 | tgw-attachment-vpc-c | Propagated |
| 0.0.0.0/0 | tgw-attachment-vpc-c | Static |

**5. Create and Configure the Central NAT Gateway**

* Create the NAT Gateway (central-NAT) with an Elastic IP in the public subnet of vpc-c (12.0.1.0/24).
* Update vpc-c Public Route Table:

0.0.0.0/0 – central-NAT ID (mentioned in step 3)

**6. Verify the Configuration**

* Test Internet Connectivity:

Ensure instances in VPC A and VPC B can access the internet through the central NAT Gateway in VPC C.

Use tools like ping google.com to verify internet access.