AWS VPC Transit Gateway

Kalyan Reddy Daida

Course Objectives

- Understand the importance of Transit Gateway and problems it solve when compared to VPC peering connections, VPN Connection features
- Create VPC, Subnets, Route Tables and EC2 VMs required for Transit Gateway
- Understand and implement Transit Gateway concepts (Attachments, Association and Propagation)
- Scenario#1: You will learn practically implementing Transit Gateway with default route tables which are auto generated (Full Mesh Architecture)
- Scenario#2: You will learn practically implementing Transit Gateway sharing across cross accounts to enable connectivity to cross account VPC's.
- Scenario#3: You will learn practically implementing Transit Gateways with custom Route Tables (Control the connectivity between VPC's using TGW Route Tables)
- You will learn practically implementing AWS Resource Access Manager basics when implementing cross account transit gateway sharing.

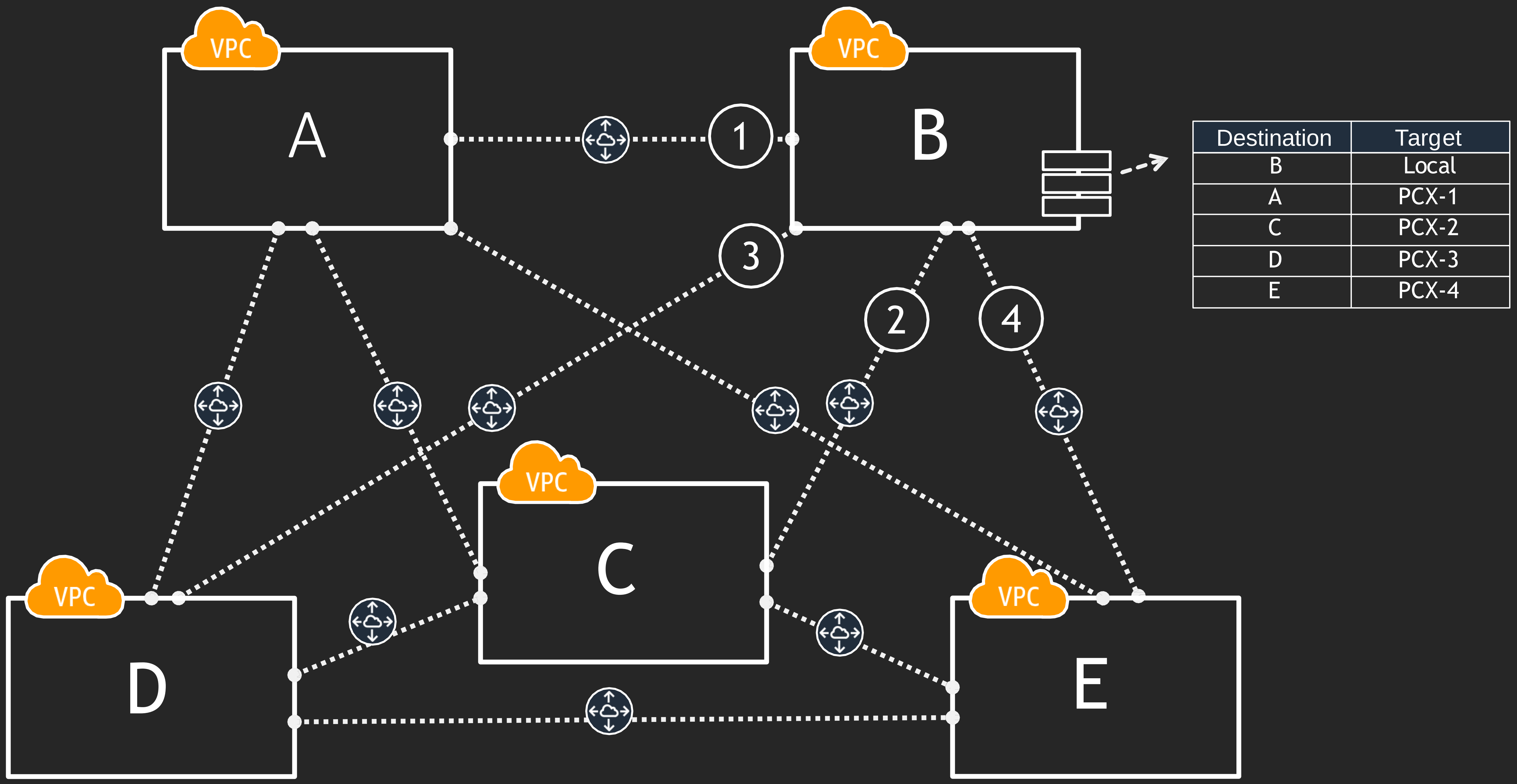
Course Structure

- Section#1: Transit Gateway Introduction
- Section#2: Pre-requisite Environment Setup
- Section#3: Create Transit Gateway with Default Route Table
- Section#4: Share Transit Gateway with Other AWS Accounts (Cross Account Sharing)
- Section#5: Transit Gateway with Custom Route Tables

Section#1: Transit Gateway Introduction

- Before Transit Gateway
  - VPC Peering
  - Transit VPC with IPsec
  - VPN Connection per VPC
- After Transit Gateway
  - Lot of things solved with single Transit Gateway.

Before: VPC Peering



Destination	Target
B	Local
A	PCX-1
C	PCX-2
D	PCX-3
E	PCX-4

Full mesh: How many Amazon VPC Peering connections do I need (full mesh)?

$$\frac{n(n-1)}{2}$$

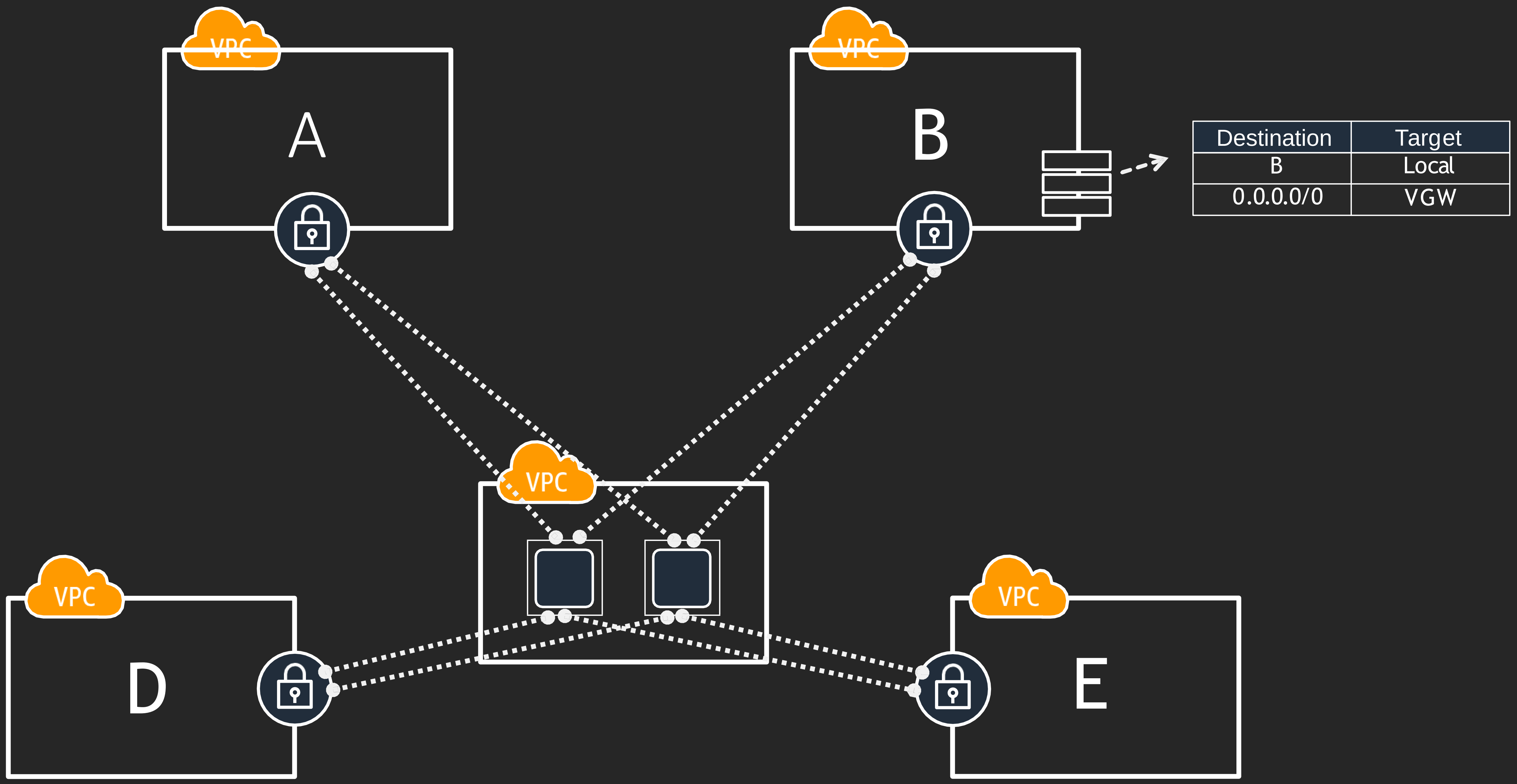
10 VPC = 45 VPC peering connections

100 VPC = 4500 VPC peering connections

Static routes per Amazon VPC route table
100

Amazon VPC Peering connections per Amazon VPC
125

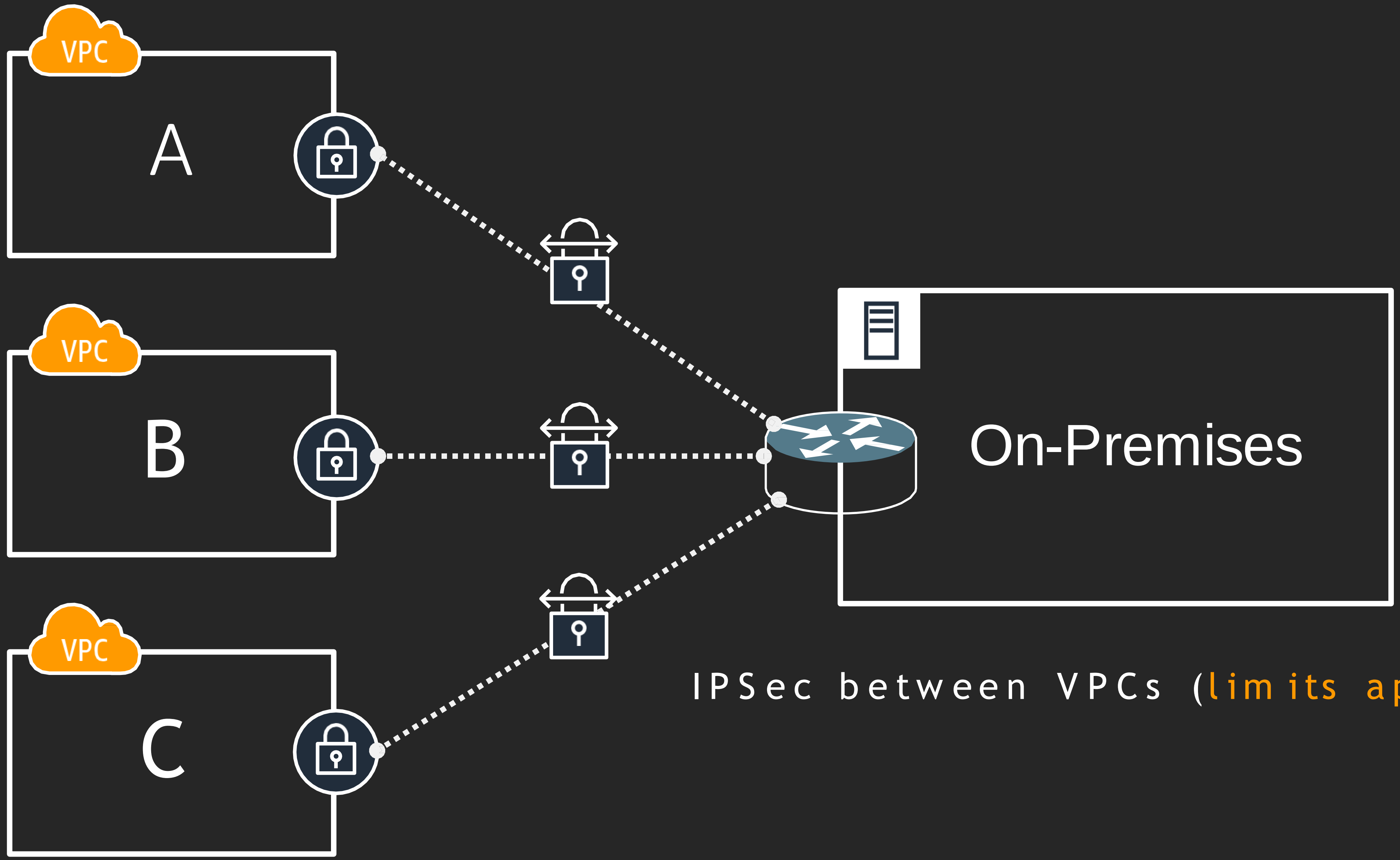
Before: Transit VPC with IPSec



Destination	Target
B	Local
0.0.0.0/0	VGW

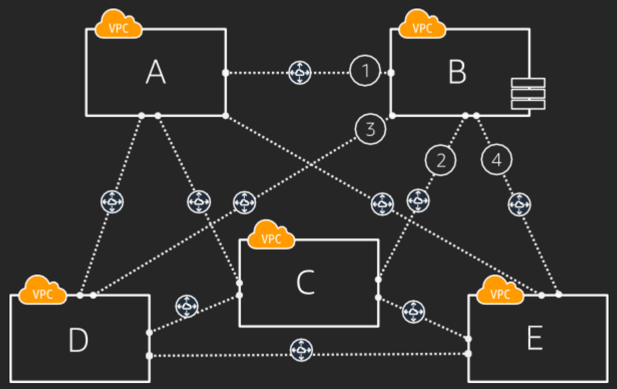
IPSec between VPCs (limits apply)

Before: VPN Connection per VPC

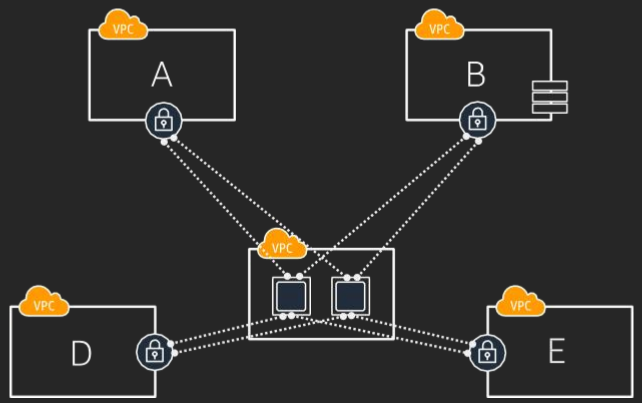


IPSec between VPCs (limits apply)

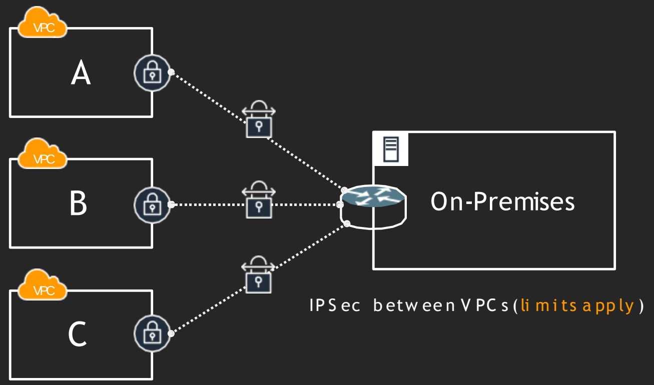
Amazon VPC Peering for full mesh connectivity



Instance based Transit Amazon VPC



VPN Connection per Amazon VPC



After TGW

1.25Gbps per VPN Connection with ECMP

*With ECMP, you can distribute traffic over multiple tunnels, e.g. 8 tunnels = 10Gbps

Multiple TGW route tables for finer routing control

50 Gbps of bandwidth per attachment per availability zone

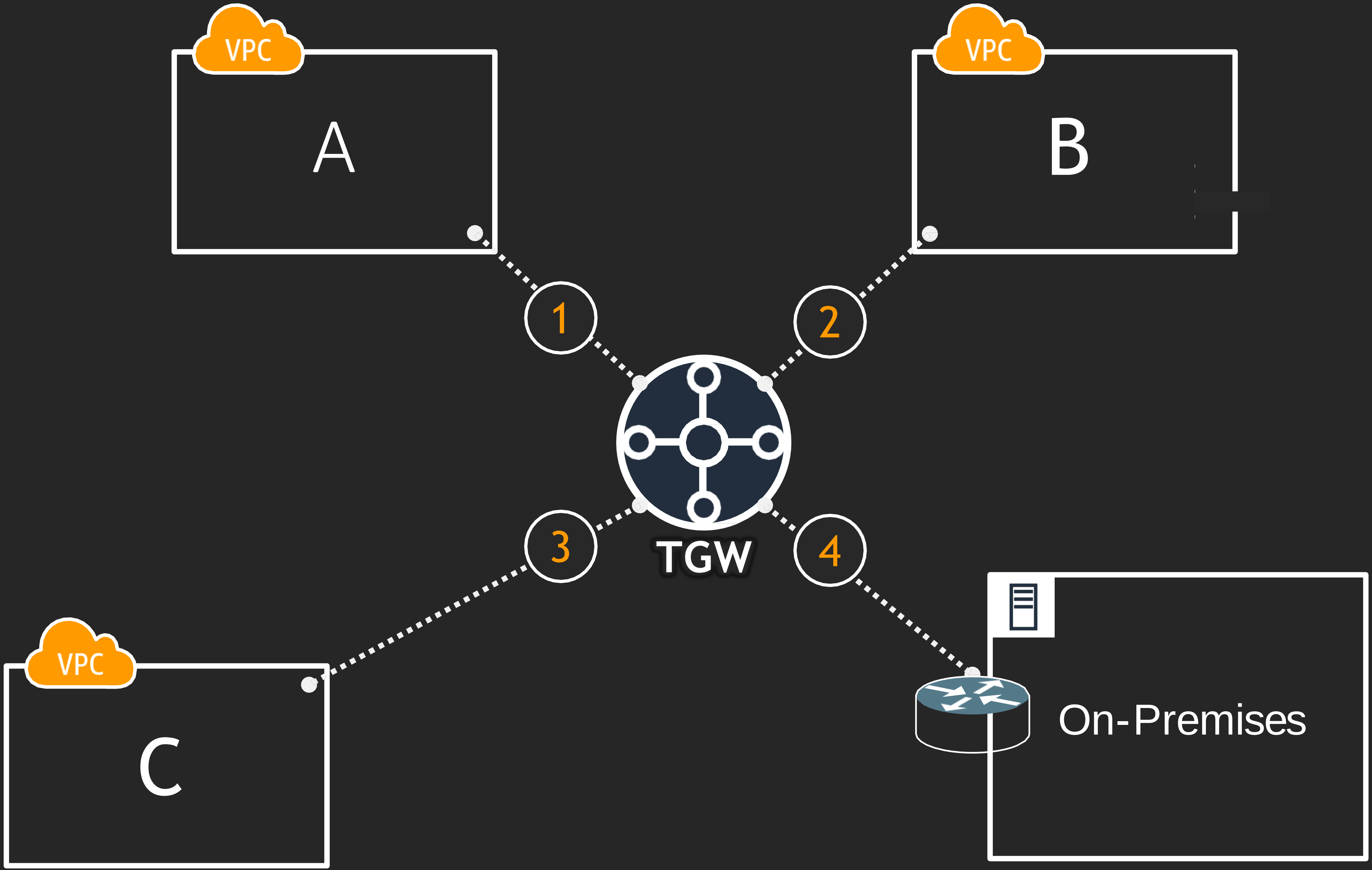
TGW is a region level construct today

10,000 routes per TGW

Up to 5000 Amazon VPC attachments per TGW

Centralized hub for routing between Amazon VPCs and on-premises to AWS

After: AWS Transit Gateway (TGW)



Attachment

The connection from a Amazon VPC and VPN to a TGW

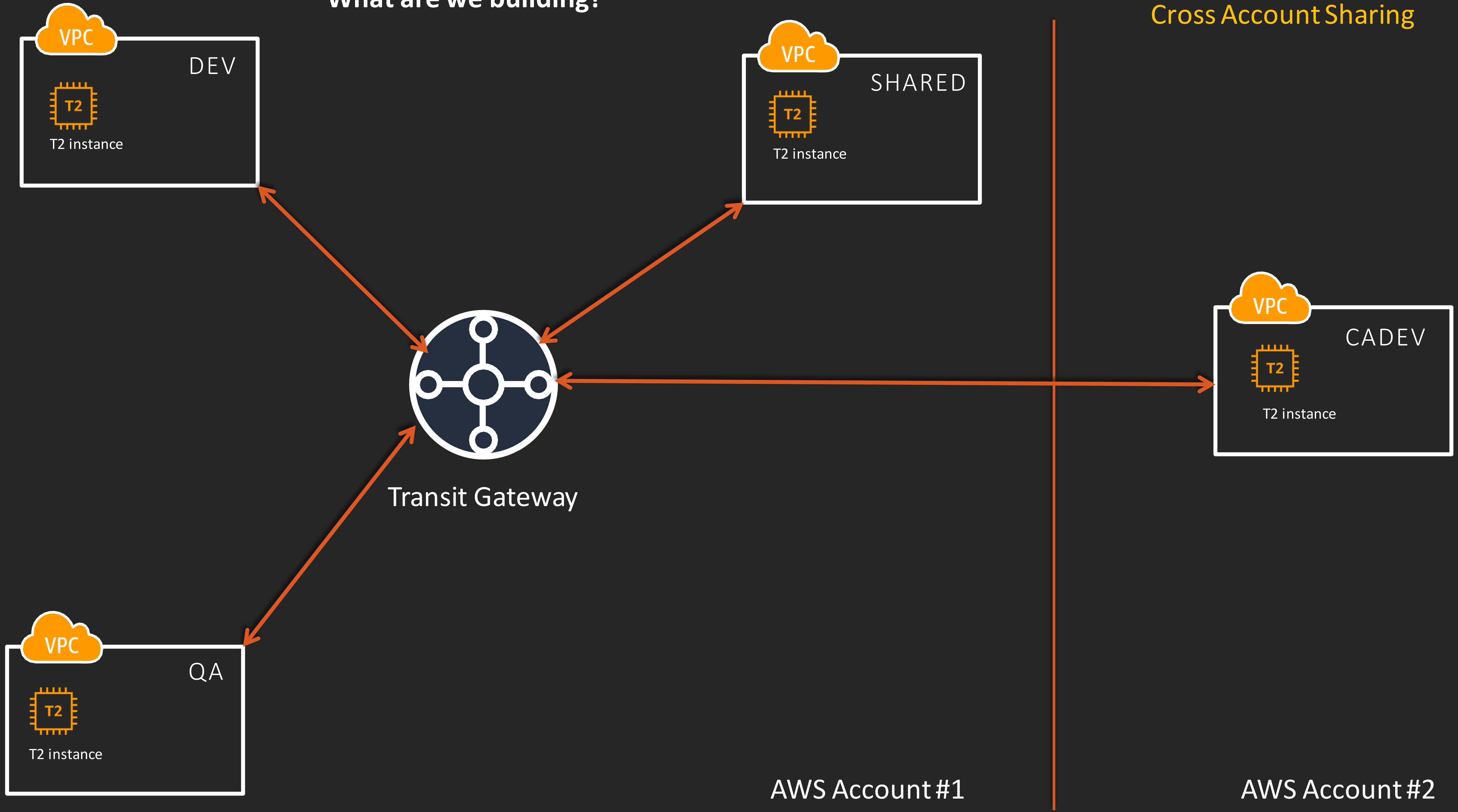
Association

The route table used to route packets coming from an attachment (from an Amazon VPC and VPN)

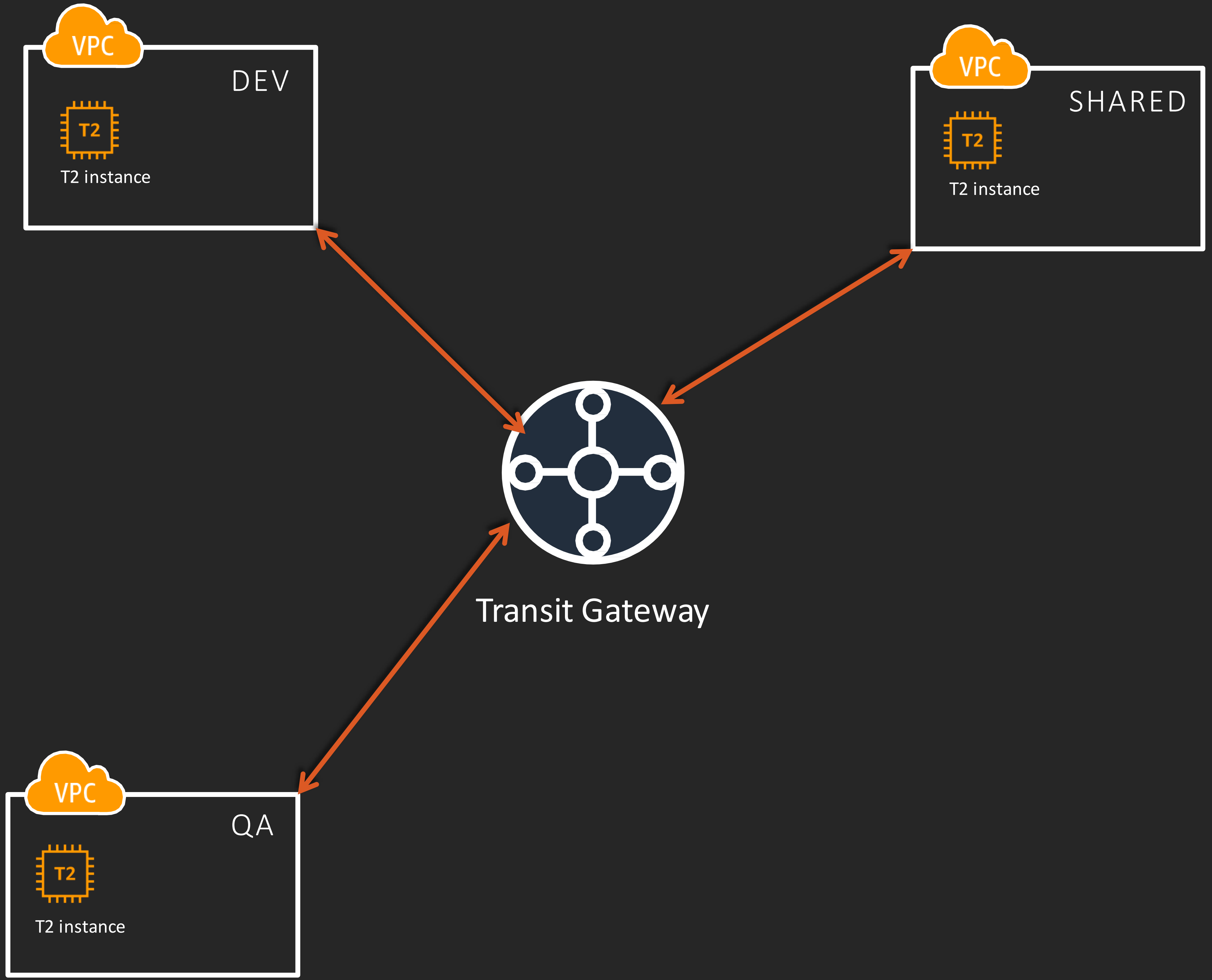
Propagation

The route table where the attachment’s routes are installed

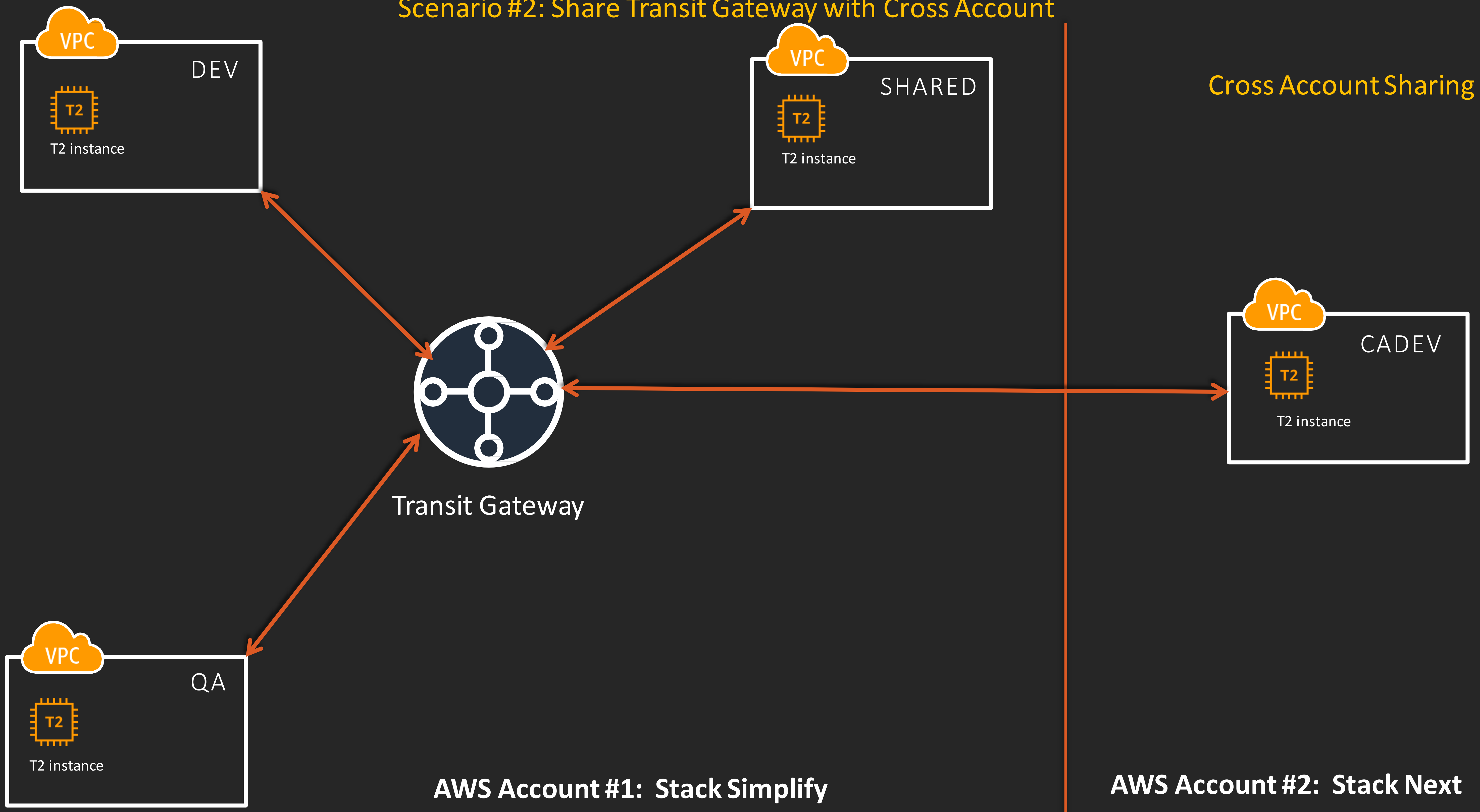
What are we building?



Scenario #1: Default Route Table



Scenario #2: Share Transit Gateway with Cross Account



Share Transit Gateway with Cross Account

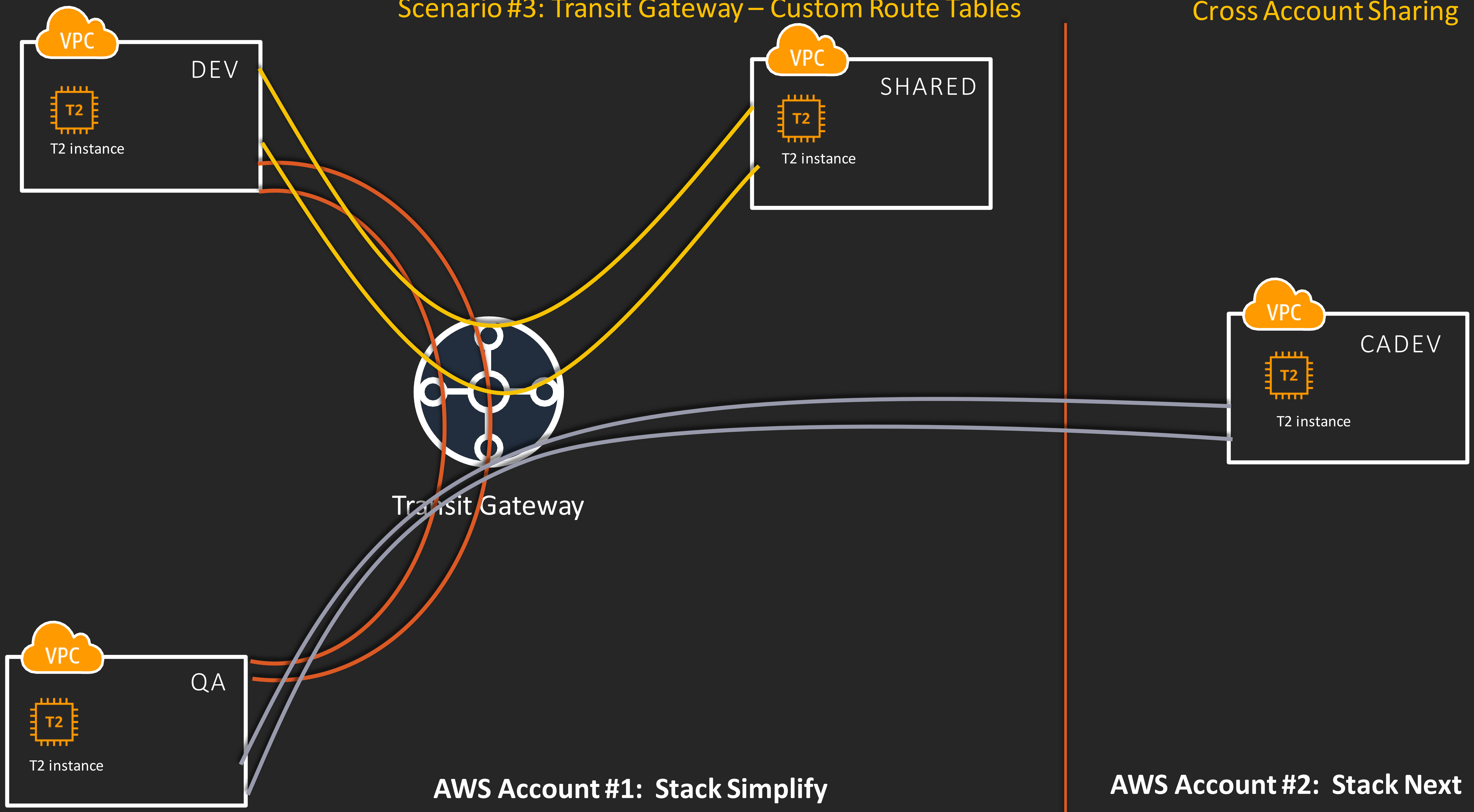
- AWS Accounts
  - First Account: Stack Simplify
  - Second Account: Stack Next
- Step#1: Create Resource Shares
  - First Account: Create Resource Share using AWS Resource Access Manager
  - Second Account: Accept the Resource Share
- Step#2: Second Account: Create VPC, Subnet, Routes, IGW & EC2 VM
- Step#3: Create VPC Attachment
  - Second Account: Create VPC Attachment
  - First Account: Accept the VPC Attachment
  - First Account: Verify the Association, Propagation & Routes for Cross Account Dev VPC.
- Step#4: Perform the telnet tests.

Transit Gateway – Custom Route Tables

- AWS Accounts
  - First Account: Stack Simplify
  - Second Account: Stack Next
- Step#1: Clean up current associations in default route table
- Step#2: Implement Custom Route Table between Dev & QA VPC
- Step#3: Implement Custom Route Table between Dev & shrd VPC
- Step#4: (Cross Account Custom Route) Implement Custom Route Table between qa & cadev VPC
- Step#5: Perform Negative Tests
  - dev to cadev → should fail
  - qa to shrd → should fail
  - cadev to dev → should fail
  - Cadev to shrd →

Scenario #3: Transit Gateway – Custom Route Tables

Cross Account Sharing



AWS Account #1: Stack Simplify

AWS Account #2: Stack Next

Transit Gateway – Custom Route Tables

- AWS Accounts
  - First Account: Stack Simplify
  - Second Account: Stack Next
- Step#1: Clean up current associations in default route table
- Step#2: Implement Custom Route Table between Dev & QA VPC
  1. Create Route Table – dev-rt
     1. Create Association – Dev VPC Attachment
     2. Create Propagation – QA VPC Attachment
     3. Verify Routes
  2. Create Route Table – qa-rt
     1. Create Association - QA VPC Attachment
     2. Create Propagation - Dev VPC Attachment
     3. Verify Routes
  3. Test Connectivity between Dev and QA

Transit Gateway – Custom Route Tables

- Step#3: Implement Custom Route Table between Dev & shrd VPC
  1. Create Route Table – dev-rt → Already exists
     1. Create Association – Dev VPC Attachment → Already exists
     2. Create Propagation – shrd VPC Attachment
     3. Verify Routes
  2. Create Route Table – shrd-rt
     1. Create Association - shrd VPC Attachment
     2. Create Propagation - Dev VPC Attachment
     3. Verify Routes
  3. Test Connectivity between Dev and SHRD

Transit Gateway – Custom Route Tables

- Step#4: (Cross Account Custom Route) Implement Custom Route Table between qa & cadev VPC
  1. Create Route Table – qa-rt → Already exists
     1. Create Association – QA VPC Attachment → Already exists
     2. Create Propagation – cadev VPC Attachment
     3. Verify Routes
  2. Create Route Table – cadev-rt
     1. Create Association - cadev VPC Attachment
     2. Create Propagation - qa VPC Attachment
     3. Verify Routes
  3. Test Connectivity between QA and CADEV

Transit Gateway – Custom Route Tables

- Step#5: Perform Negative Tests
  - dev to cadev → should fail
  - qa to shrd → should fail
  - cadev to dev → should fail
  - cadev to shrd → should fail