Week 6 Homework

This is the network diagram from last night. I'll upload the actual Visio file also. Your homework is to replicate this within your groups and send me the results. There are your constraints.

1) Transit Gateway must be used.

2) Bastion Windows box must be in a public subnet.

3) Bastion must ping to a Linux box within a private subnet in an AZ without a public subnet.

4) Screenshot must show the following 2 commands: Ipconfig and ping

5) TGW screenshot must show attachments.

Advice: Return to Maarek and study Section 27 again.

This solution will guide you through replicating the architecture using AWS Management Console (GUI) and Terraform. The CIDR ranges provided are:

* CIDR Range 1: 10.100.0.0/16
* CIDR Range 2: 10.101.0.0/16

The architecture:

1. Transit Gateway (TGW) connecting multiple VPCs.

2. A Bastion Windows EC2 instance in a public subnet.

3. A Linux EC2 instance in a private subnet within a VPC without a public subnet.

4. VPN or direct internet access for external communication.

5. ICMP (ping) connectivity from the Bastion Windows EC2 to the Linux EC2 instance via the Transit Gateway.

6. Transit Gateway displaying attachments to different VPCs.

Steps to Set Up in AWS Management Console (GUI):

1. Create two VPCs:

* Prod VPC (10.100.0.0/16) with public and private subnets in AZ-A.
* Dev VPC (10.101.0.0/16) with a private subnet in AZ-B.

2. Subnets Configuration

* Prod VPC (10.100.0.0/16):
  + Public Subnet (10.100.1.0/24) in AZ-A for the Windows Bastion EC2.
  + Private Subnet (10.100.11.0/24) in AZ-A for internal resources.
* Dev VPC (10.101.0.0/16):
  + Private Subnet (10.101.2.0/24) in AZ-B for the Linux EC2.

3. Create Internet Gateway (IGW)

* Attach an Internet Gateway to the Prod VPC.
* Add a route for public traffic from 0.0.0.0/0 to the IGW in the Prod public subnet route table.

4. Create a Transit Gateway (TGW)

* Open the AWS Console and go to the VPC Dashboard.
* In the left-hand menu, click Transit Gateways under Transit Gateway.
* Click Create Transit Gateway.
* Provide a name for your TGW
* Leave other options as default unless you need specific configuration for Amazon side ASN, Multicast support, etc.
* Click Create Transit Gateway.

5. Attach VPCs to the Transit Gateway (TGW): Once your Transit Gateway is created, you need to attach the Prod VPC (with both public and private subnets) and the Dev VPC (with the private subnet).

* In the VPC Dashboard, navigate to Transit Gateway Attachments.
* Click Create Transit Gateway Attachment.

6. Prod VPC Attachment (Public and Private Subnets):

* Transit Gateway ID: Select your Transit Gateway
* VPC ID: Select your Prod VPC.
* Subnets: Select both the public and private subnets for the Prod VPC.
* Click Create attachment.

7. Dev VPC Attachment (Private Subnet Only):

* Transit Gateway ID: Select your Transit Gateway
* VPC ID: Select your Dev VPC.
* Subnets: Select the private subnet for the Dev VPC.
* Click Create attachment.

8. Modify Route Tables: Now that the VPCs are attached to the Transit Gateway, you need to modify the route tables for each VPC so that traffic can flow between them.

9. Prod VPC Route Table (Public and Private Subnets):

* Go to the VPC Dashboard, and click on Route Tables in the left-hand menu.
* Find the route table associated with the Prod VPC private subnet and public subnet.
* In the private subnet and public subnet route tables:
  + Click Edit routes.
  + Add a route for the Dev VPC CIDR.
  + For Target, select the Transit Gateway.
* Click Save routes.

10. Dev VPC Route Table (Private Subnet):

* In the VPC Dashboard, click on Route Tables.
* Find the route table associated with the Dev VPC private subnet.
* Click Edit routes.
* Add a route for the Prod VPC CIDR (10.100.0.0/16).
* For Target, select the Transit Gateway.
* Click Save routes.

11. Verify Connectivity: Ensure that the Prod VPC and Dev VPC are now able to communicate with each other through the Transit Gateway.

* You can ping instances between the Prod and Dev VPCs to verify the connectivity.

12. Windows Bastion Host Setup

* Launch a Windows EC2 instance in the public subnet (10.100.1.0/24).
* Open RDP (port 3389) in the Security Group for remote access.

13. Linux Instance Setup

* Launch a Linux EC2 instance in the private subnet (10.101.2.0/24).
* Open ICMP (ping) and SSH in the Security Group to allow communication with the Bastion Host.

14. Route Tables & Security Groups

* Public subnet route table in Prod VPC allows internet access via IGW.
* Private subnets use the Transit Gateway for cross-VPC communication:
  + In Prod VPC, traffic destined for 10.101.0.0/16 goes through the TGW.
  + In Dev VPC, traffic to 10.100.0.0/16 routes via the TGW.
* Security Groups should allow:
  + RDP and ICMP on the Windows Bastion.
  + ICMP and SSH on the Linux EC2.

15. Testing

* Connect to the Windows Bastion via RDP.
* Run ipconfig in the command prompt to verify the internal IP.
* Run ping <Linux Private IP> to test connectivity from Bastion to the Linux EC2.

Explanation:

1. Transit Gateway: A central connection point for routing traffic between the Prod VPC and Dev VPC.

2. Public Subnet in Prod VPC: Contains the Windows Bastion with direct internet access via the Internet Gateway.

3. Private Subnets in both VPCs: The Linux EC2 resides in the Dev VPC, accessible via the Transit Gateway.

4. Security Groups: Windows Bastion allows RDP (3389) and ICMP (ping). Linux EC2 allows ICMP and internal VPC traffic.

Terraform Code:

provider "aws" {

region = "us-east-1"

}

# Prod VPC (10.100.0.0/16)

resource "aws\_vpc" "prod\_vpc" {

cidr\_block = "10.100.0.0/16"

}

# Dev VPC (10.101.0.0/16)

resource "aws\_vpc" "dev\_vpc" {

cidr\_block = "10.101.0.0/16"

}

# Public Subnet for Prod VPC in AZ-A

resource "aws\_subnet" "prod\_public\_subnet" {

vpc\_id = aws\_vpc.prod\_vpc.id

cidr\_block = "10.100.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

}

# Private Subnet for Prod VPC in AZ-A

resource "aws\_subnet" "prod\_private\_subnet" {

vpc\_id = aws\_vpc.prod\_vpc.id

cidr\_block = "10.100.2.0/24"

availability\_zone = "us-east-1a"

}

# Private Subnet for Dev VPC in AZ-B

resource "aws\_subnet" "dev\_private\_subnet" {

vpc\_id = aws\_vpc.dev\_vpc.id

cidr\_block = "10.101.2.0/24"

availability\_zone = "us-east-1b"

}

# Internet Gateway for Prod VPC

resource "aws\_internet\_gateway" "prod\_igw" {

vpc\_id = aws\_vpc.prod\_vpc.id

}

# Route table for public subnet in Prod VPC

resource "aws\_route\_table" "prod\_public\_route\_table" {

vpc\_id = aws\_vpc.prod\_vpc.id

}

# Route for public subnet to send traffic to the internet via IGW

resource "aws\_route" "prod\_public\_route" {

route\_table\_id = aws\_route\_table.prod\_public\_route\_table.id

destination\_cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.prod\_igw.id

}

# Associate the route table to the public subnet

resource "aws\_route\_table\_association" "prod\_public\_route\_table\_association" {

subnet\_id = aws\_subnet.prod\_public\_subnet.id

route\_table\_id = aws\_route\_table.prod\_public\_route\_table.id

}

# Transit Gateway (TGW)

resource "aws\_ec2\_transit\_gateway" "tgw" {

description = "My Transit Gateway"

}

# TGW attachment for Prod VPC (only attach the private subnet)

resource "aws\_ec2\_transit\_gateway\_vpc\_attachment" "prod\_tgw\_attach" {

transit\_gateway\_id = aws\_ec2\_transit\_gateway.tgw.id

vpc\_id = aws\_vpc.prod\_vpc.id

subnet\_ids = [aws\_subnet.prod\_private\_subnet.id]

depends\_on = [aws\_ec2\_transit\_gateway.tgw]

}

# TGW attachment for Dev VPC (private subnet)

resource "aws\_ec2\_transit\_gateway\_vpc\_attachment" "dev\_tgw\_attach" {

transit\_gateway\_id = aws\_ec2\_transit\_gateway.tgw.id

vpc\_id = aws\_vpc.dev\_vpc.id

subnet\_ids = [aws\_subnet.dev\_private\_subnet.id]

depends\_on = [aws\_ec2\_transit\_gateway.tgw]

}

# Prod VPC private route table

resource "aws\_route\_table" "prod\_private\_route\_table" {

vpc\_id = aws\_vpc.prod\_vpc.id

}

# Route for Prod VPC private subnet to send traffic to Dev VPC via TGW

resource "aws\_route" "prod\_private\_route\_tgw" {

route\_table\_id = aws\_route\_table.prod\_private\_route\_table.id

destination\_cidr\_block = "10.101.0.0/16"

transit\_gateway\_id = aws\_ec2\_transit\_gateway.tgw.id

depends\_on = [aws\_ec2\_transit\_gateway\_vpc\_attachment.prod\_tgw\_attach]

}

# Associate route table with the private subnet in Prod VPC

resource "aws\_route\_table\_association" "prod\_private\_route\_table\_association" {

subnet\_id = aws\_subnet.prod\_private\_subnet.id

route\_table\_id = aws\_route\_table.prod\_private\_route\_table.id

}

# Dev VPC route table

resource "aws\_route\_table" "dev\_private\_route\_table" {

vpc\_id = aws\_vpc.dev\_vpc.id

}

# Route for Dev VPC to send traffic to Prod VPC via TGW

resource "aws\_route" "dev\_private\_route\_tgw" {

route\_table\_id = aws\_route\_table.dev\_private\_route\_table.id

destination\_cidr\_block = "10.100.0.0/16"

transit\_gateway\_id = aws\_ec2\_transit\_gateway.tgw.id

depends\_on = [aws\_ec2\_transit\_gateway\_vpc\_attachment.dev\_tgw\_attach]

}

# Associate route table with the private subnet in Dev VPC

resource "aws\_route\_table\_association" "dev\_private\_route\_table\_association" {

subnet\_id = aws\_subnet.dev\_private\_subnet.id

route\_table\_id = aws\_route\_table.dev\_private\_route\_table.id

}