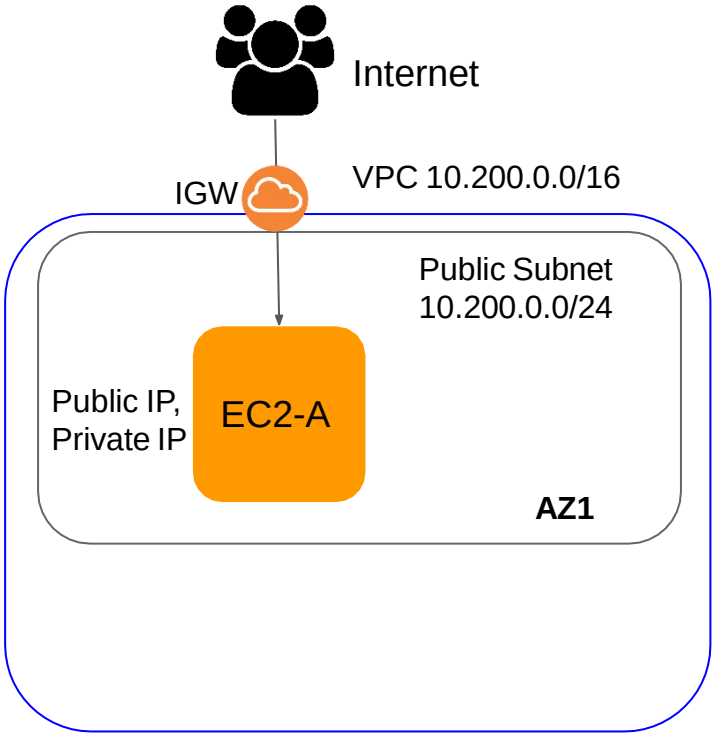
VPC with Single Public Subnet



Public Subnet Route Table

Destination	Target
10.200.0.0/16	local
0.0.0.0/0	igw-xxx

Note: For EC2 instance to be reachable from internet, it must be in Public Subnet and must have Public IP

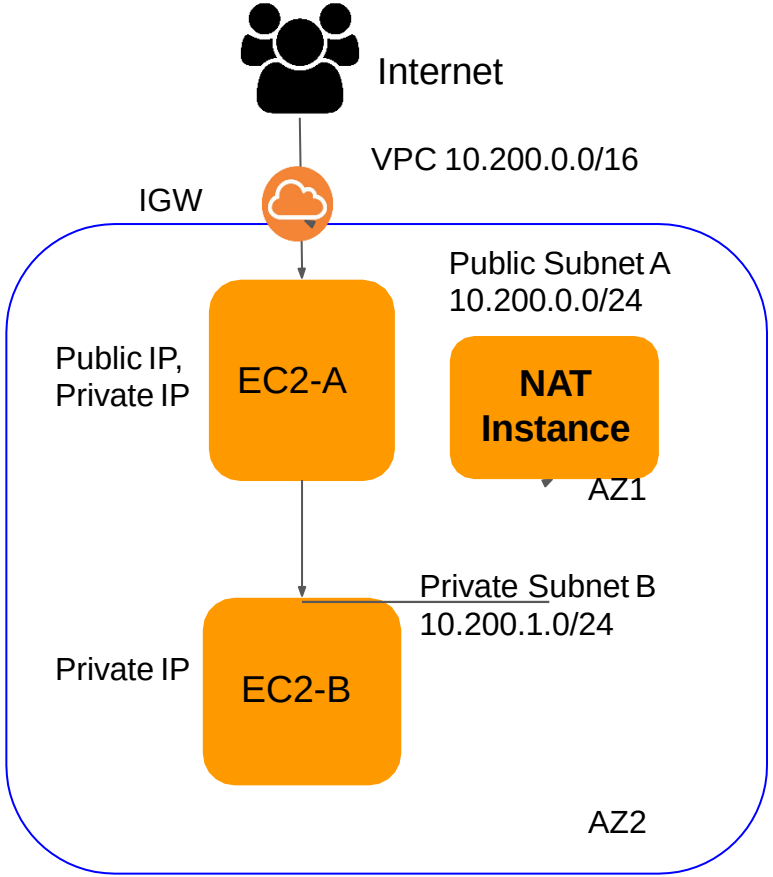
Steps

1. Create VPC
   a. Go to VPC service -> Your VPCs -> Create VPC (Name: MyVPC, CIDR: 10.200.0.0/16) -> Create
2. Create Internet Gateway
   a. Internet Gateways -> Create internet gateway
3. Attach Internet Gateway to VPC
   a. Select Internet gateway -> Actions -> Attach to VPC -> Select your VPC
4. Create Subnet
   a. Subnets -> Create subnet (Name: MyVPC-Public, VPC: MyVPC, AZ: Select any az, CIDR: 10.200.0.0/24)
   b. Select Subnet -> Action -> Modify Auto Assign Public IP -> Enable -> Save
5. Create Public Route table
   a. Route Tables -> Create Route Table (Name: MyVPC-Public, VPC: MyVPC)
   b. Select Route table -> Routes -> Edit -> Add another route (Destination: 0.0.0.0/0, Target: Internet gateway -> igw-xxx) -> Save

Steps

6. Associate Route table with Subnet to make it Public subnet
    a. Select Route table -> Subnet Associations -> Edit -> Check the MyVPC-Public subnet -> Save
7. Launch EC2 instance in newly created Public Subnet
    a. Go to EC2 Service -> Instances
    b. Launch EC2 Instance -> Select Amazon Linux 2 -> Select t2.micro
    c. Configure Instance Details:
        i. Network: MyVPC
        ii. Subnet: MyVPC-Public (rest all defaults)
    d. Add storage (all defaults)
    e. Add Tags
        i. Key=Name, Value=**EC2-A**
    f. Configure Security Group
        i. Add rule for SSH port 22 for source as MyIP
    g. Review and Launch
8. Connect to EC2 instance (Public IP) from your workstation using Putty or terminal (ec2-user)

Using NAT Instance



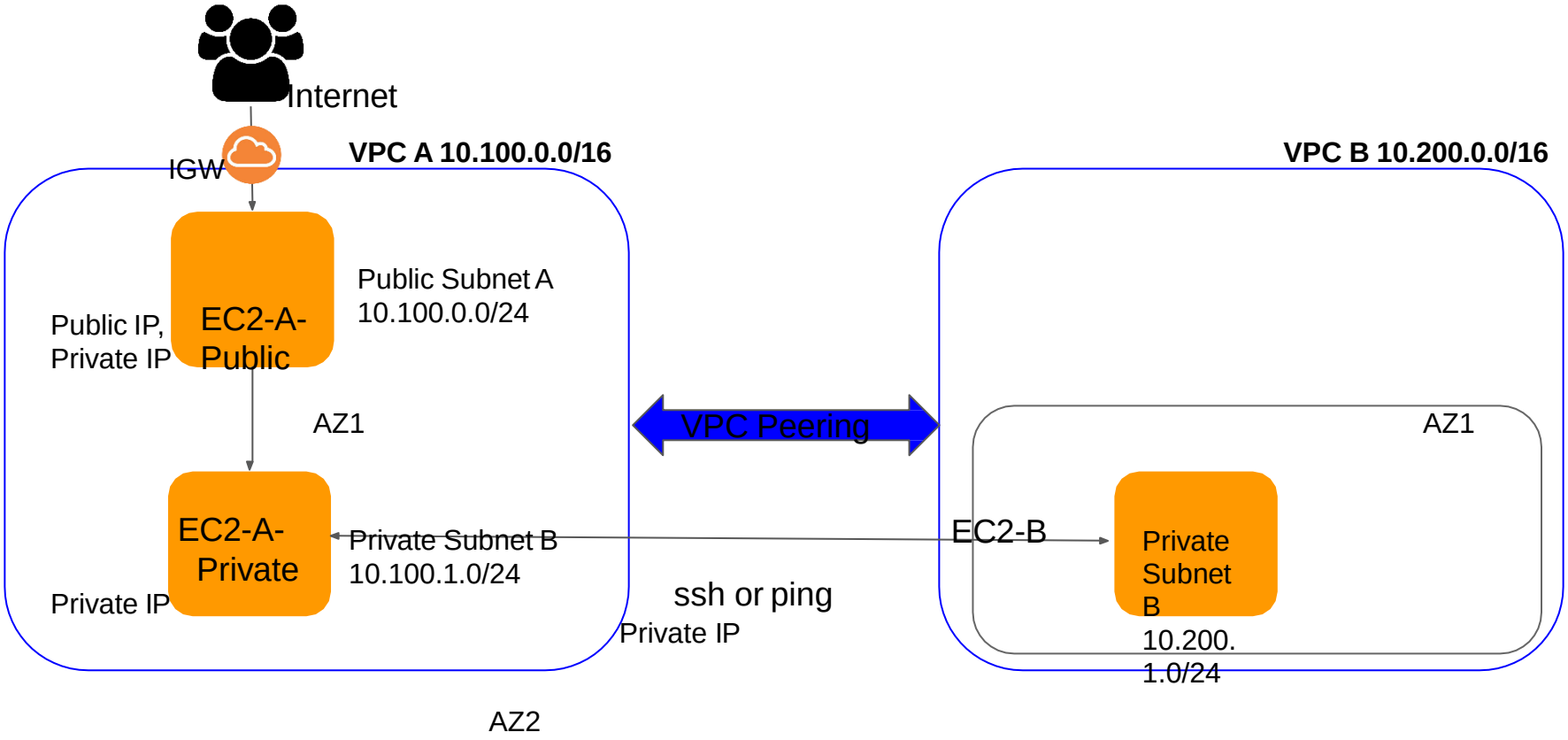
Private Subnet Route Table

Destination	Target
10.200.0.0/16	local
0.0.0.0/0	NAT-INSTANCE-ID

NAT Instance

Continuing with the Lecture 2 setup (VPC Public and Private Subnets)

1. Create a NAT **EC2 Instance** in public subnet using NAT AMI amzn-ami-vpc-nat
2. Security group to allow traffic on port 80 from private subnet CIDR
3. Disable source/destination check
4. Update route table of private subnet and add route for internet traffic (0.0.0.0/0) using NAT instance id
5. Login to EC2-A and from there login to EC2-B over ssh
6. Now try to access internet from EC2-B instance. Should be successful
   a. ping google.com


Internet
IGW
VPC A 10.100.0.0/16
VPC B 10.200.0.0/16
Public IP,
Private IP
EC2-A-
Public
Public Subnet A
10.100.0.0/24
AZ1
VPC Peering
AZ1
EC2-A-
Private
Private Subnet B
10.100.1.0/24
EC2-B
Private
Subnet
B
10.200.
1.0/24
Private IP
ssh or ping
Private IP
AZ2

VPC Peering

1. Create 2 VPCs - VPC-A and VPC-B with non-overlapping CIDR range
2. Create 1 public, 1 private subnets in each VPC-A
3. Create 1 private subnet in VPC-B
4. Create 1 instance in each VPC in subnets created above.
5. For VPC-A Public EC2 instance open security group 22 for your Public IP (Myip)
6. For VPC-A Private EC2 instance open security group 22 for VPC-A public subnet CIDR
7. For VPC-B EC2, open security group to allow port 22 and ICMP from other VPC-A private subnet CIDR
8. Login (ssh) to VPC-A Public EC2 instance. From there connect to VPC-A Private instance. You should be able to connect. (You need to have ssh key)
9. From VPC-A Private EC2, try to connect to VPC-B EC2 over Private IP. Does not connect.
10. Now create VPC peering between VPC-A and VPC-B.
11. Accept peering request from requested VPC-B
12. Modify route tables for both the subnets and add corresponding routes. i.e in VPC-A Subnet route table add route for destination as VPC-B CIDR and vice-a-versa.
13. Now try again to connect or ping from VPC-A EC2 to VPC-B EC2 using **Private IP**. You should be able to connect/ping

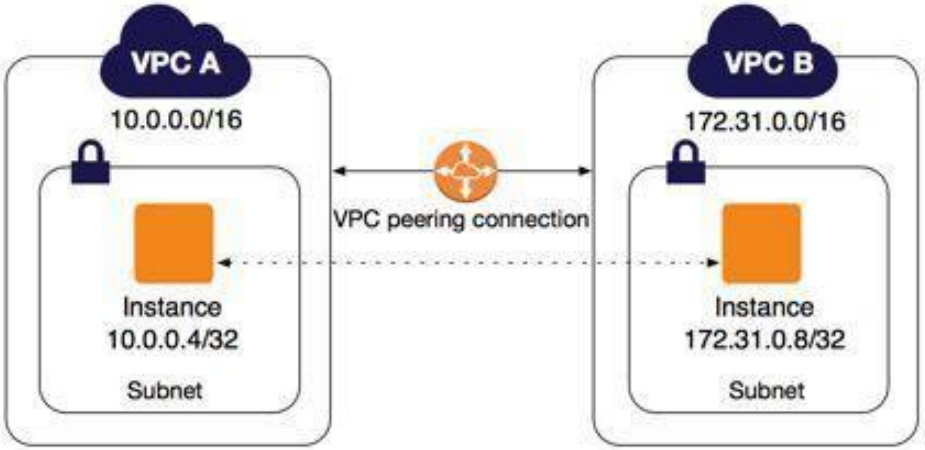
Understanding AWS VPC Peering

Features, Benefits, Scenarios, Limitations
and unsupported configurations

Inter VPC communication - VPC Peering

Features

- Enables private communication between 2 or more AWS VPCs
- Intra region and Inter region
- Across different AWS accounts= EFS
- Can connect single VPC to multiple VPCs
- There is no single point of failure for communication or a bandwidth bottleneck.



Benefits

- No need to have Internet gateway or VPN connection between VPCs.
- Saves VPN connection ($0.05/ hr) cost. You only pay in/out data transfer charges of $0.01/GB

VPC Peering Scenarios - 1

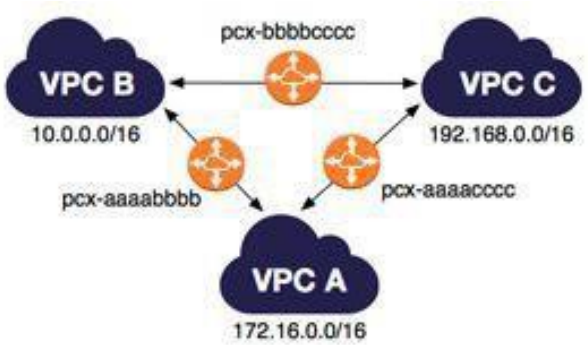
2 VPCs Peered together

Example:

You have different departments Finance, Accounting hosting machines into separate AWS VPCs. You can provide full access to all the resources from one VPC to another VPC and vice-a-versa

VPC Peering Scenarios - 2

3 VPCs Peered together

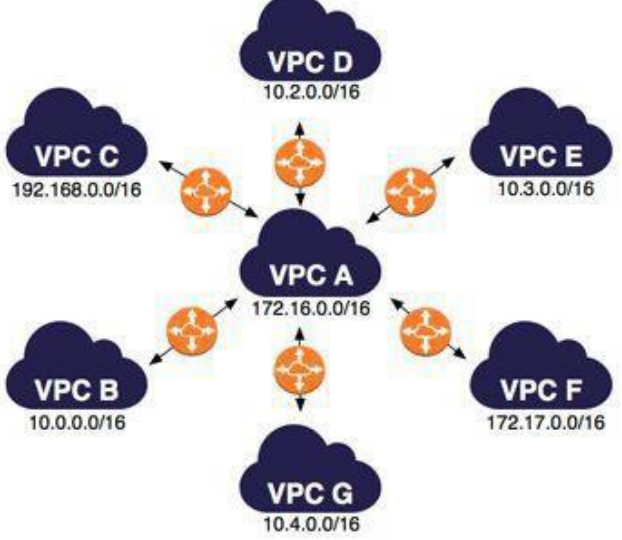


Example:

You have different office branches having separate VPCs in same of different AWS accounts and resources in these branches needs to access resources in all other branches

VPC Peering Scenarios - 3

Multiple VPCs Peered to single VPC (Spoke model)

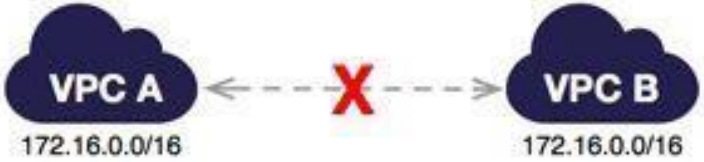


Examples:

1. When you host a centralized services e.g File sharing, AD into one VPC and need to provide access to people or applications hosted into different VPCs
2. You offer SaaS service to customers having their own workloads into their AWS accounts. You can provide SaaS access to all customers making sure individual customers VPCs can not communicate with each other.
3. You have centralized Management VPC (e.g Chef server, Jenkins etc) and you have individual customer specific VPCs where you have hosted your single tenant services in each VPC. You can push your configurations as well as monitor health Infra and applications through management VPC.

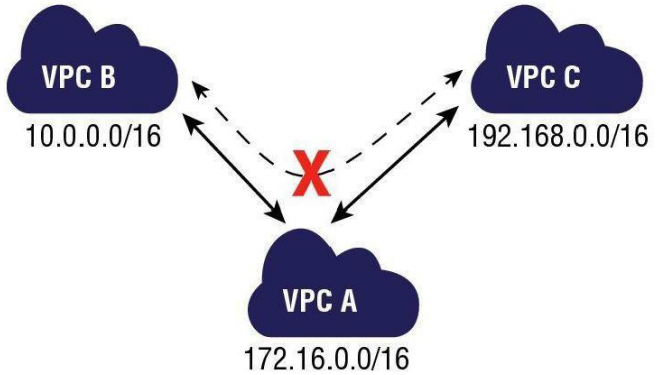
VPC Peering Limitations

1. VPC CIDR's should be Non-overlapping



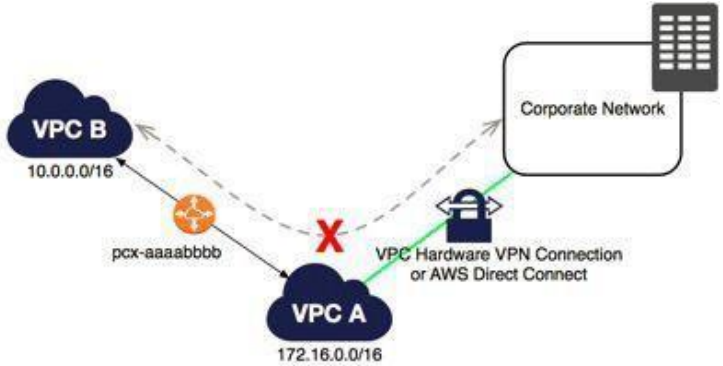
1. Only one peering between given 2 VPCs

1. Peering connection is **not transitive**

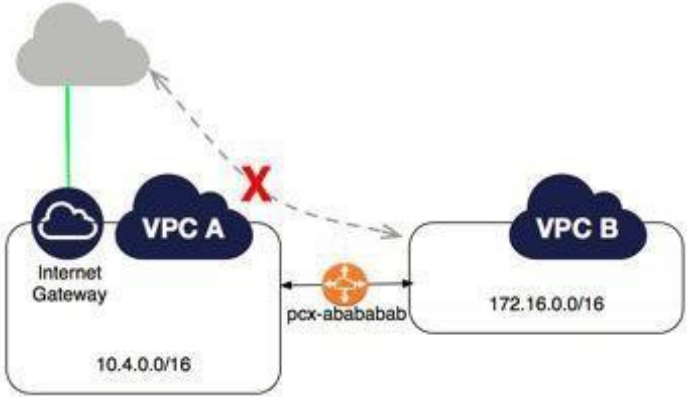


VPC Peering invalid scenarios

1. A VPN connection or an AWS Direct Connect connection to a corporate network

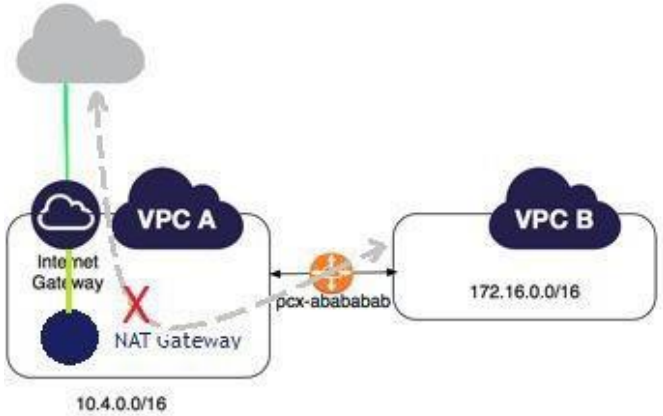


1. An internet access through an internet gateway

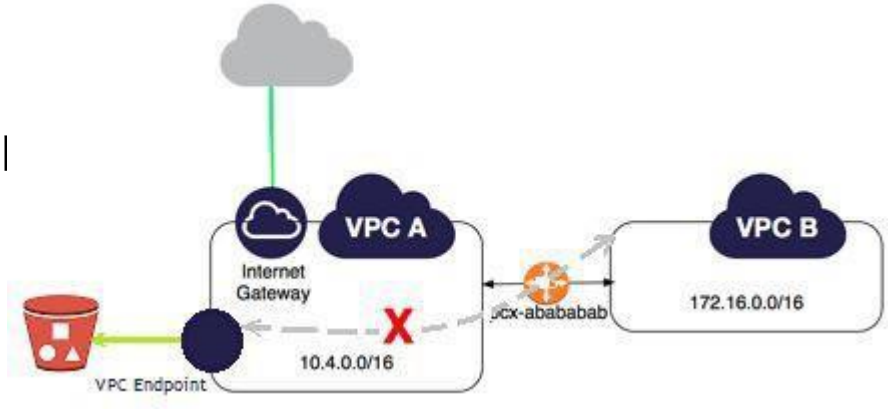


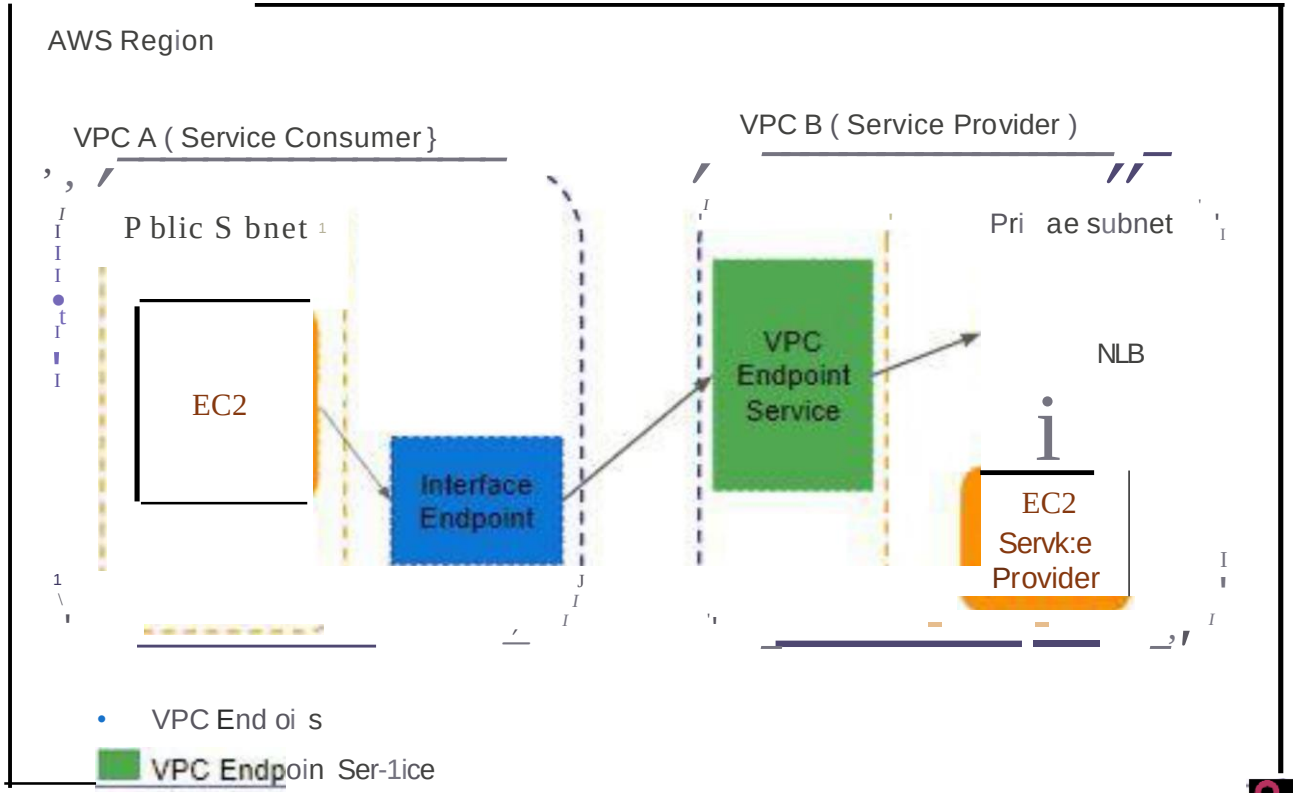
VPC Peering invalid scenarios

3. An internet access through a NAT device

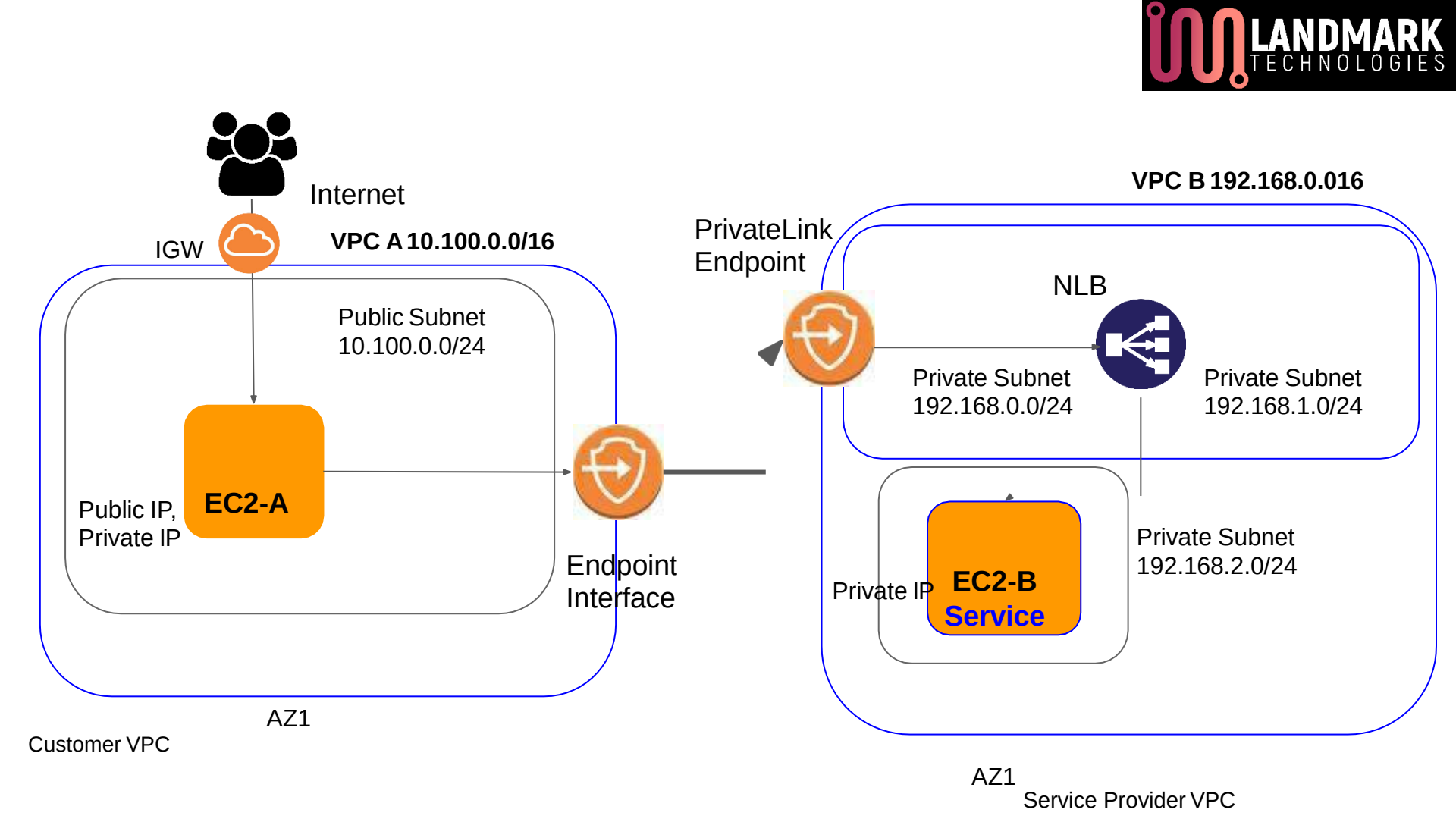


4. A VPC endpoint to an AWS service(End to Amazon S3, DynamoDB)





LANDMARK TECHNOLOGIES


Internet
IGW
VPC A 10.100.0.0/16
Public Subnet
10.100.0.0/24
EC2-A
Public IP,
Private IP
Endpoint
Interface
AZ1
Customer VPC
PrivateLink
Endpoint
VPC B 192.168.0.016
NLB
Private Subnet
192.168.0.0/24
Private Subnet
192.168.1.0/24
Private Subnet
192.168.2.0/24
Private IP
EC2-B
Service
AZ1
Service Provider VPC

Prerequisites for this exercise

Must have pre-configured Web server AMI to launch EC2 instance in Private subnet

Steps to create AMI

1. Launch EC2 instance (Amazon Linux) in default VPC with following Userdata

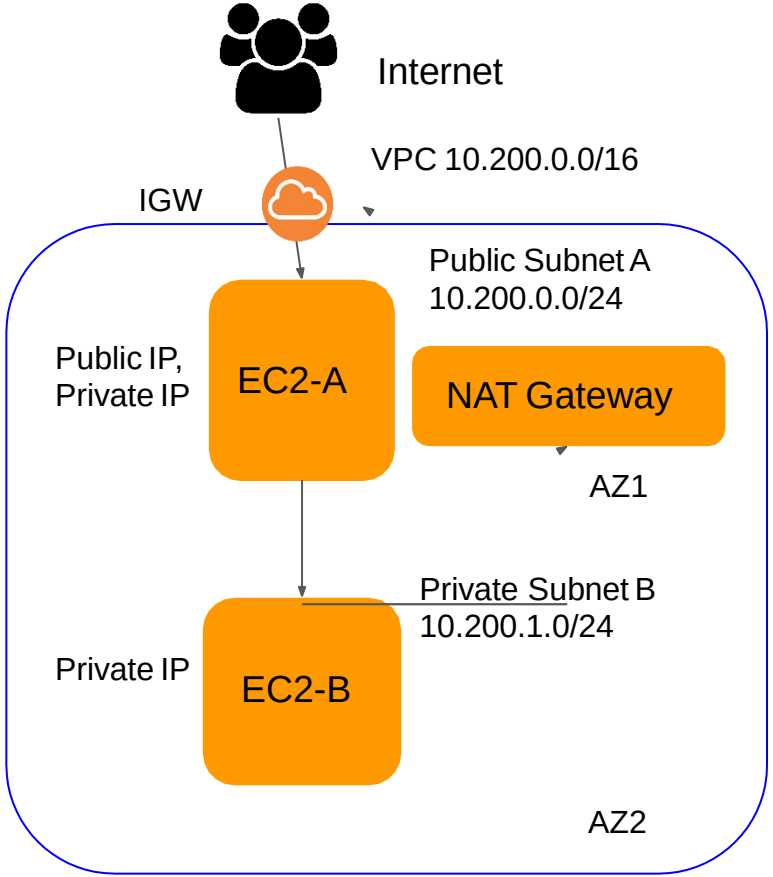
```
#!/bin/bash
yum install httpd -y
service httpd start
chkconfig httpd on
echo "This is my web server" > /var/www/html/index.html
echo successful
```

2. Verify that Web server is accessible by hitting EC2 Public IP
3. Stop the instance and create Amazon Machine Image (AMI)
4. Terminate EC2 instance

Steps

1. Create VPC-A in say **N. Virginia (us-east-1)** region. Create and attach IGW.
2. In VPC-A, create Public Subnet VPC-A-Public and launch EC2-A instance with Public IP. Open Security group port 22 for MyIP
3. Create VPC-B in same region
4. In VPC-B, create 2 Private subnets (VPC-B-NLB-1 and VPC-B-NLB-2) across AZs for hosting Network Load Balancer
5. In VPC-B, create one more Private subnet VPC-B-Private in AZ-1
6. In VPC-B private subnet VPC-B-Private, launch EC2-B instance from **pre-configured AMI** (web server). Open security group for HTTP port 80 for NLB subnets
7. In VPC-B, create Network Load Balancer in VPC-B-NLB-1 and VPC-B-NLB-2 subnets, create target group and add EC2-B instance to target group
8. In VPC-B, create **VPC Endpoint Service** with target as Network Load Balancer
9. In VPC-A, create **VPC Endpoint interface** for the VPC Endpoint service created above
    - Make sure you create endpoint interface in same AZ as VPC-A public subnet
    - Also create security group for VPC endpoint interface to allow HTTP traffic from VPC-A CIDR
10. Login to VPC-A EC2 instance over SSH and access VPC Endpoint DNS using curl command
    $curl <vpc endpoint dns>
11. You should be able to get the response from EC2-B webserver

Using NAT Gateway



Private Subnet Route Table

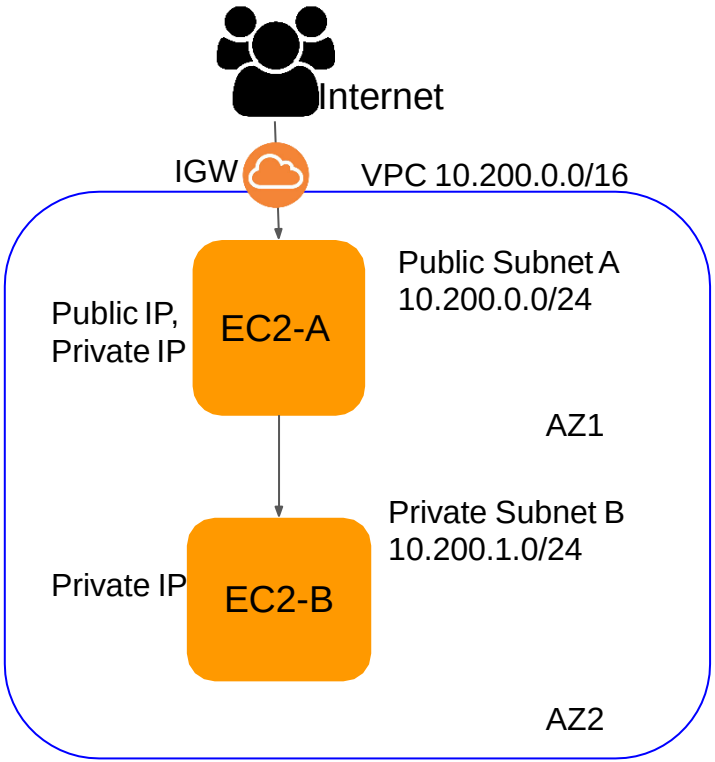
Destination	Target
10.200.0.0/16	local
0.0.0.0/0	NAT-Gateway-ID

Add a NAT Gateway

Continuing with the previous setup (VPC Public and Private Subnets)

1. Create a NAT Gateway in public subnet
2. Update route table of private subnet and add route for internet traffic (0.0.0.0/0) using NAT Gateway
3. Login to EC2-A and from there login to EC2-B over ssh
4. Now try to access internet from EC2-B instance. Should be successful
   a. ping google.com
5. Delete NAT gateway and Release EIP

VPC with Public and Private subnets



Public Subnet Route Table

Destination	Target
10.200.0.0/16	local
0.0.0.0/0	igw-xxx

EC2-A Security Group

Port	Source
22	MyIp

Private Subnet Route Table

Destination	Target
10.200.0.0/16	local

EC2-B Security Group

Port	Source
22	10.200.0.0/24
ICMP IPv4 All	10.200.0.0/24

LANDMARK TECHNOLOGIES

VPC with Public and Private subnets

1. Create a new VPC with CIDR 10.200.0.0/16
2. Create Internet Gateway (IGW) and Associate with VPC
3. Create a Public subnet 10.200.0.0/24. Enable auto assign public ip.
4. Create a Private subnet 10.200.1.0/24.
5. Create 2 security groups. For Public EC2 allow SSH from your ip. For Private EC2, allow SSH and, ICMP IPv4 All traffic from VPC CIDR 10.200.0.0/16
6. With previously created ssh key, Launch an EC2 instance (A) in Public Subnet. Instance should have Public IP and Private IP.
7. Launch other EC2 instance (B) in private subnet. Instance should have only Private IP.
8. Connect to EC2-A over Public IP using SSH from your workstation
9. Create your SSH key file on EC2-A (.pem), modify permissions to 600
10. SSH from EC2-A to EC2-B over EC2-B private IP

VPC with Public and Private subnets