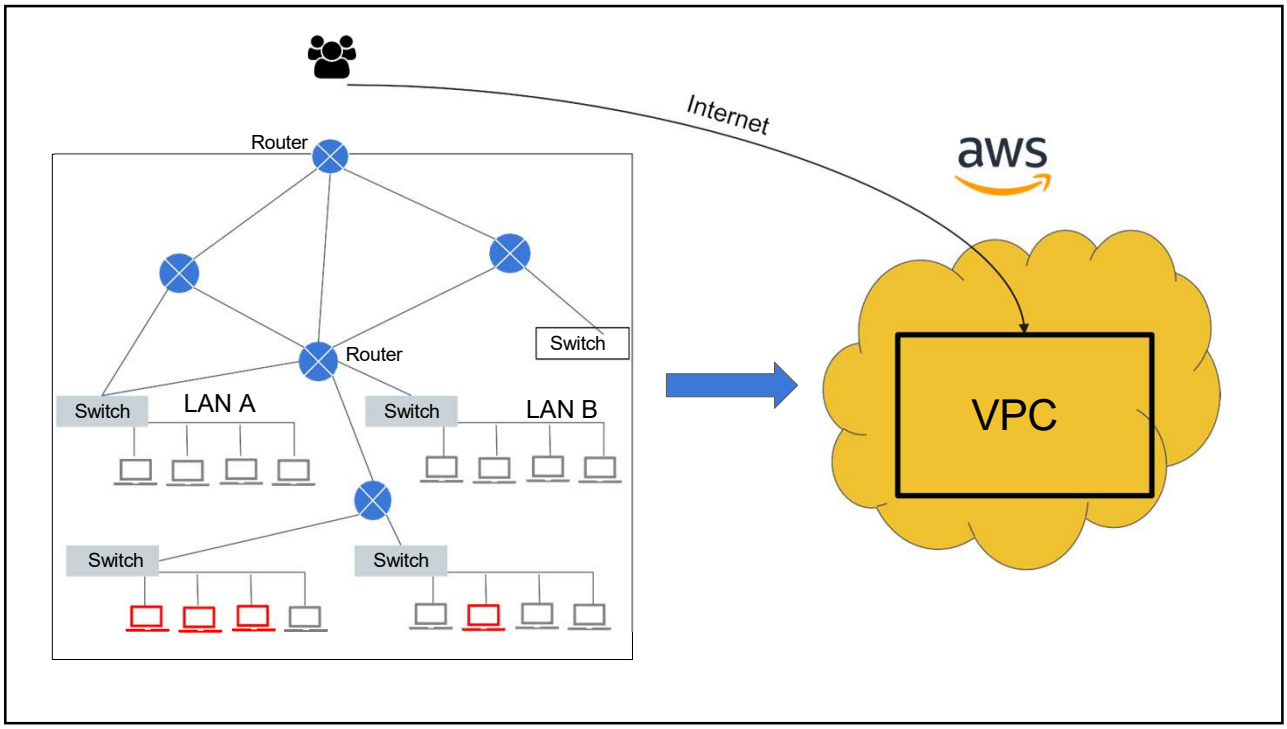
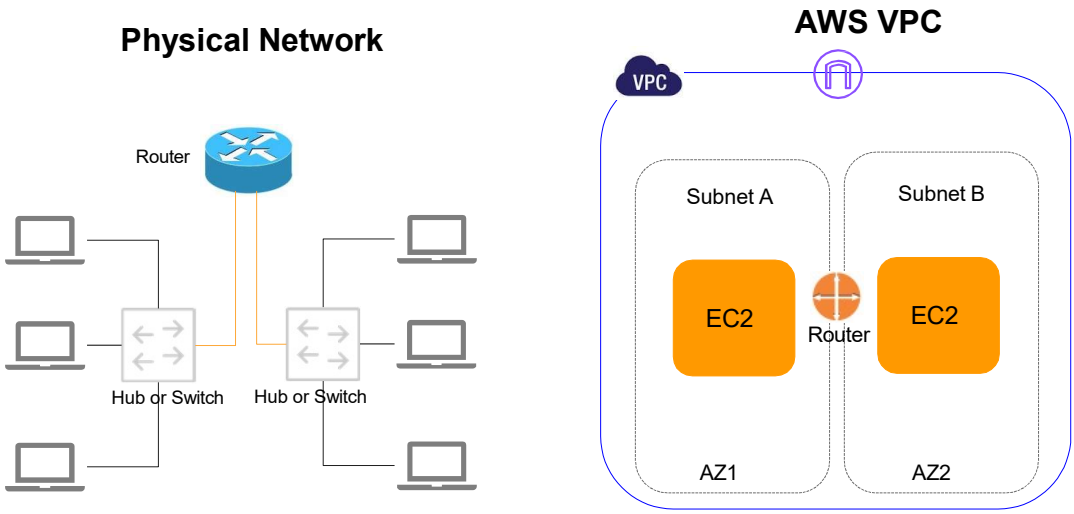
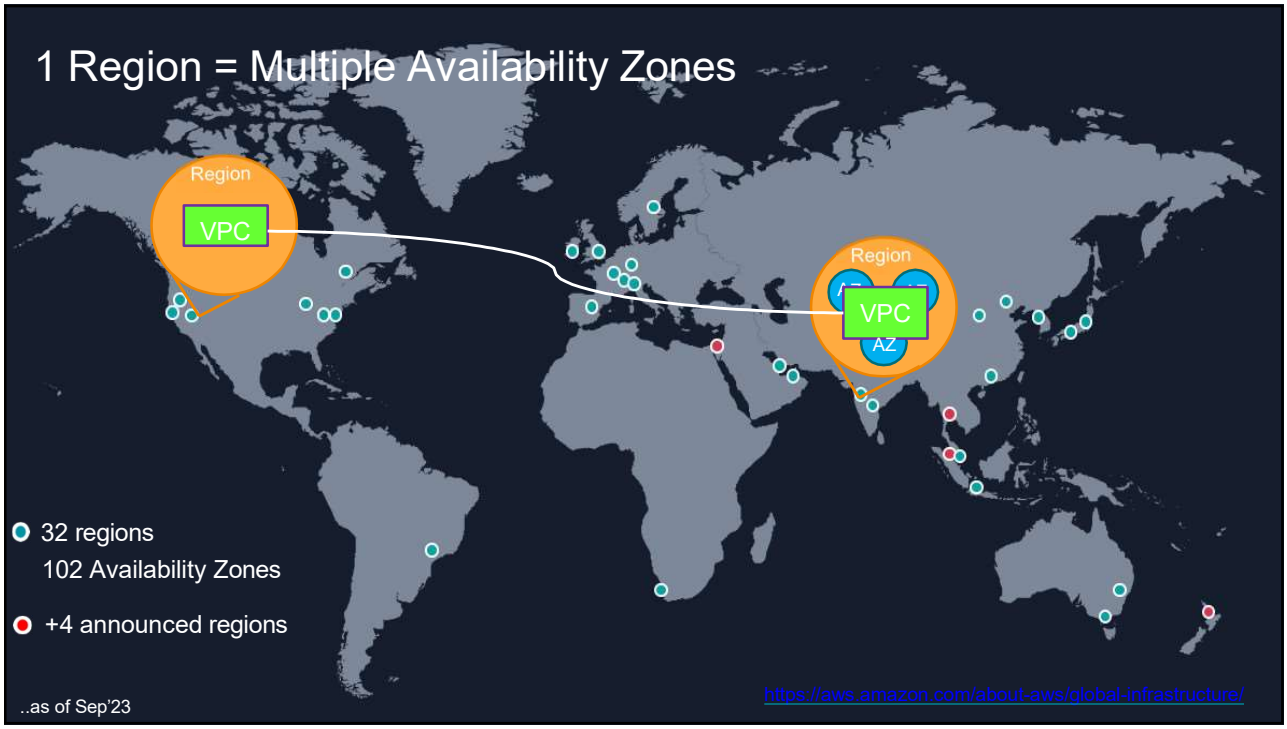
Amazon VPC

Moving from traditional on-premises network to virtual network in the Cloud

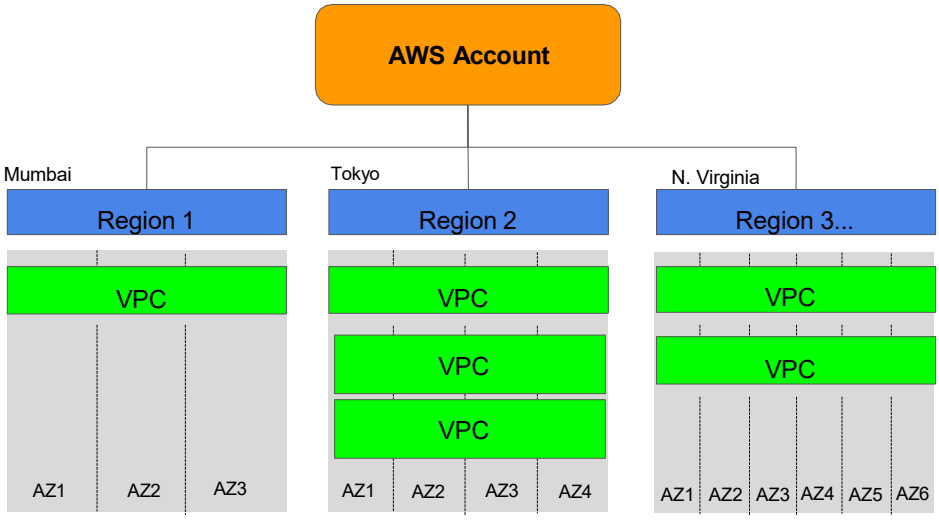


Traditional IT network vs VPC

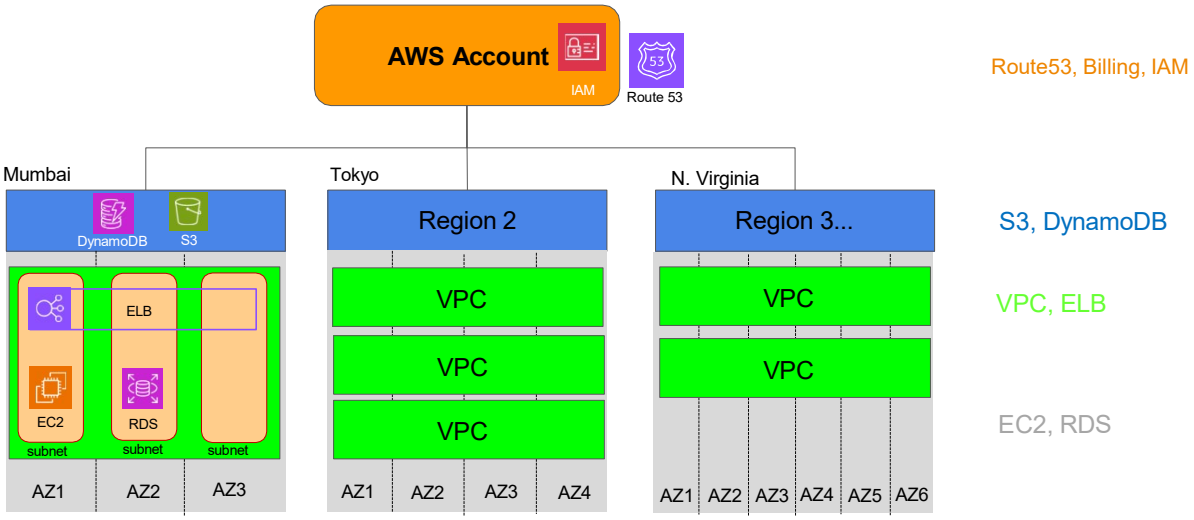




VPC Scope with respect to AWS account, region and AZs

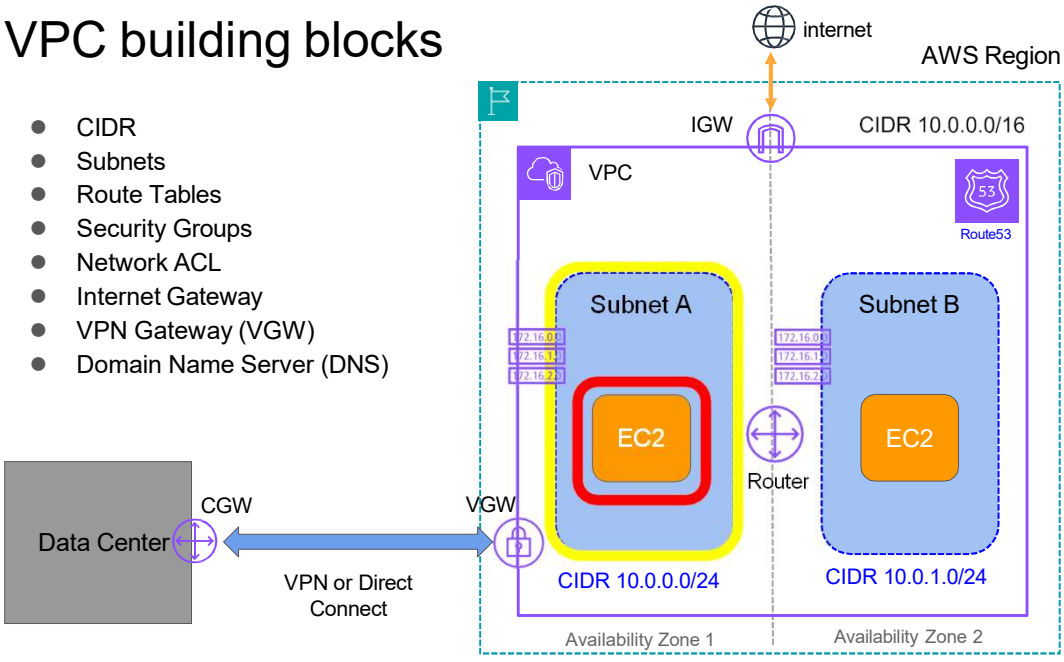


AWS services scope with respect to VPC

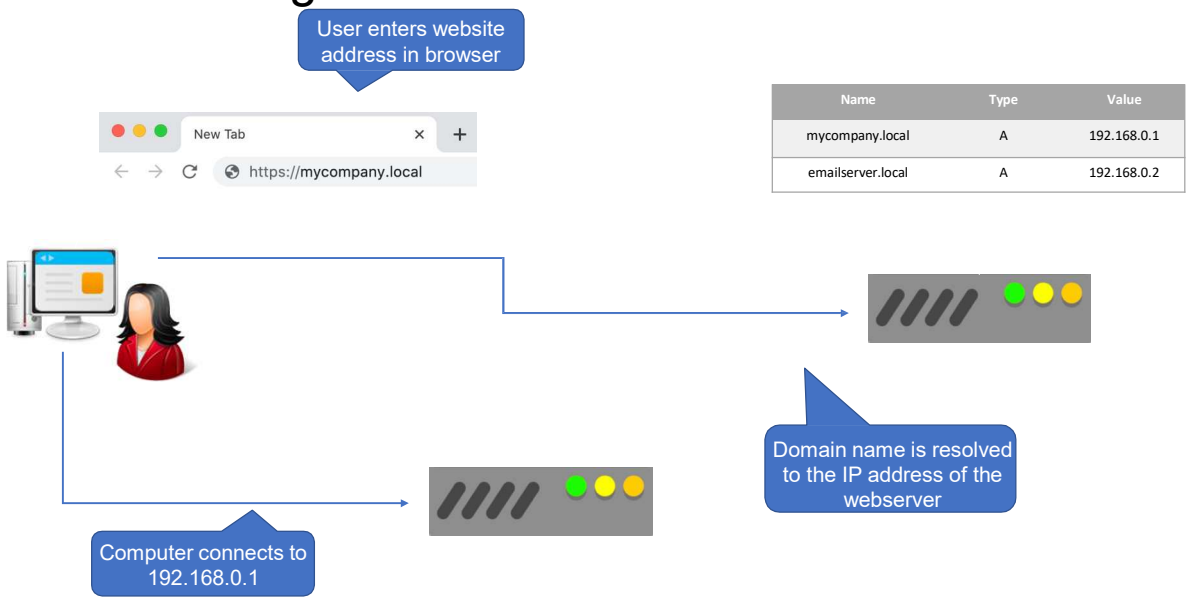


VPC building blocks

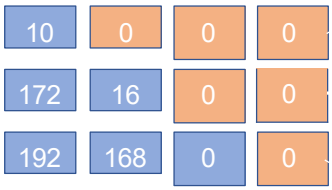
- CIDR
- Subnets
- Route Tables
- Security Groups
- Network ACL
- Internet Gateway
- VPN Gateway (VGW)
- Domain Name Server (DNS)



IP Addressing Basics



IP Addressing Basics



Private IP Addresses

private usage

RFC 1918

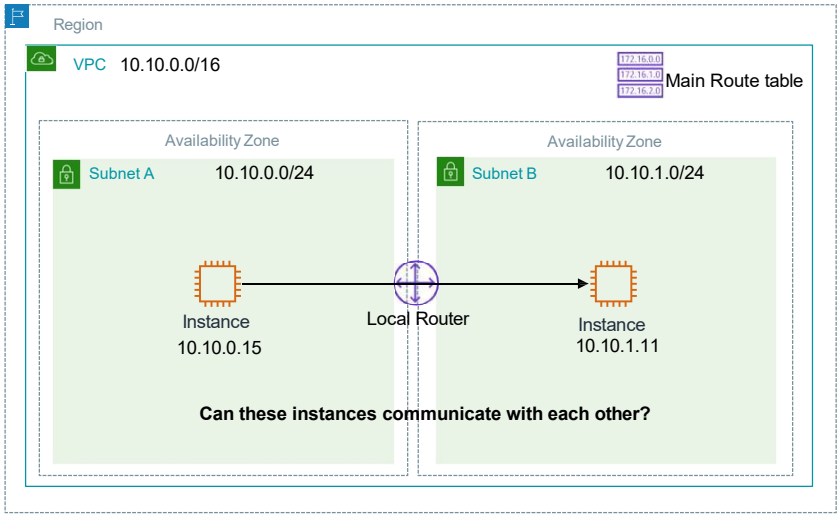
CIDR	First Address	Last Address
10.0.0.0/8	10.0.0.0	10.255.255.255
172.16.0.0/12	172.16.0.0	172.31.255.255
192.168.0.0/16	192.168.0.0	192.168.255.255

NOT

VPC Addressing

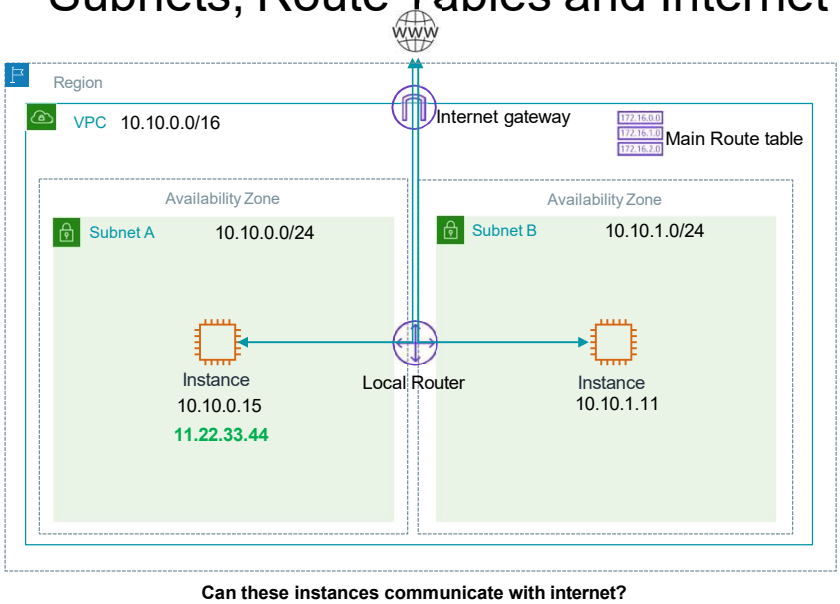
- **AWS VPC CIDR (IPv4)**
  - VPC prefix between /16 (65536 IPs) and /28 (16 IPs)
  - RFC 1918 IP ranges for Private network and corresponding AWS recommended ranges
    - 10.0.0.0/8 => 10.0.0.0 – 10.255.255.255 **=>** AWS CIDR 10.X.0.0/16
    - 172.16.0.0/12 => 172.16.0.0 - 172.31.255.255 **=>** AWS CIDR 172.16.0.0/16 to 172.31.0.0/16
    - 192.168.0.0/16 => 192.168.0.0 - 192.168.255.255 **=>** AWS CIDR 192.168.0.0/16
  - Subnet CIDR prefix between /16 to /28 (same as VPC CIDR)
- **AWS VPC CIDR (IPv6)**
  - VPC CIDR with prefix /56 (2^72 IPs)
  - IPv6 CIDR is allocated by AWS
  - Subnet CIDR prefix /64
  - IPv6 IP addresses are globally unique and publicly routable

Subnets, Route Tables and Internet Gateway



Destination	Target
10.10.0.0/16	Local

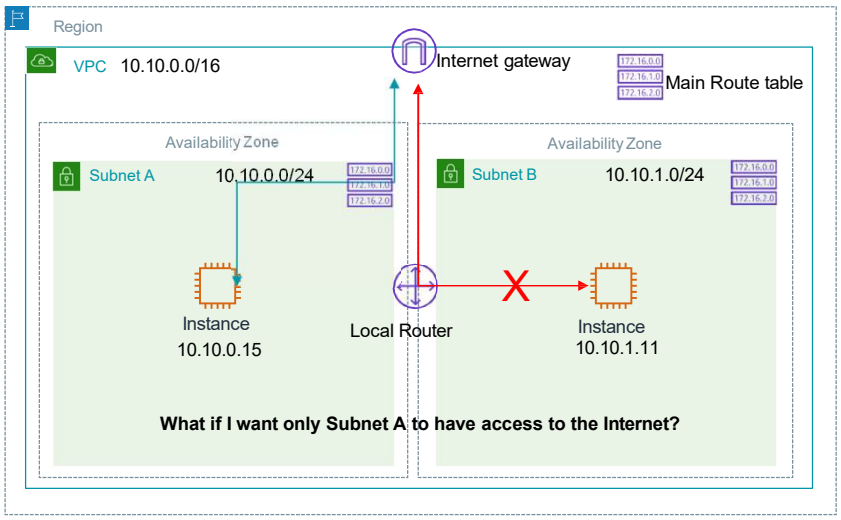
Subnets, Route Tables and Internet Gateway



Destination	Target
10.10.0.0/16	Local
0.0.0.0/0	Internet Gateway

Can these instances communicate with internet?

Subnets, Route Tables and Internet Gateway



Subnet A Route Table

Destination	Target
10.10.0.0/16	Local
0.0.0.0/0	Internet Gateway

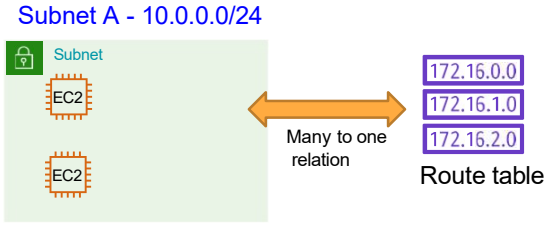
Subnet B Route Table

Destination	Target
10.10.0.0/16	Local

Now subnets are **not** following the Main Route Table

Route Table

- Contains rules to route the traffic in/out of Subnets
- Main route table at VPC level
- Custom route table at Subnet level
- Each route table contains default immutable local route for VPC
- If no custom route table is defined, then new subnets are associated with Main route table
- We can also modify the main route table



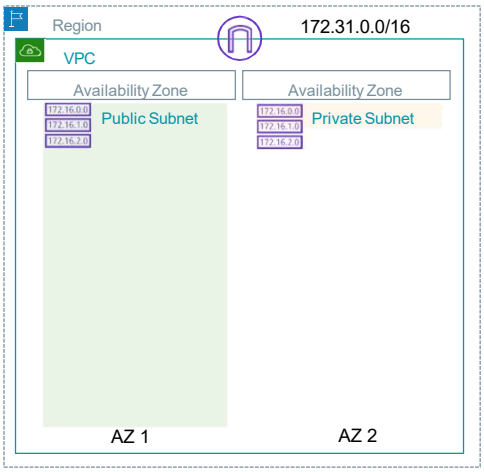
Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-xxxxx

Subnets

- Public Subnet
  - Has route for Internet
  - Instances with Public IP can communicate to internet
  - Ex: NAT, Web servers, Load balancer

Destination	Target
172.31.0.0/16	local
0.0.0.0/0	igw-xxx

1 Subnet => 1 AZ

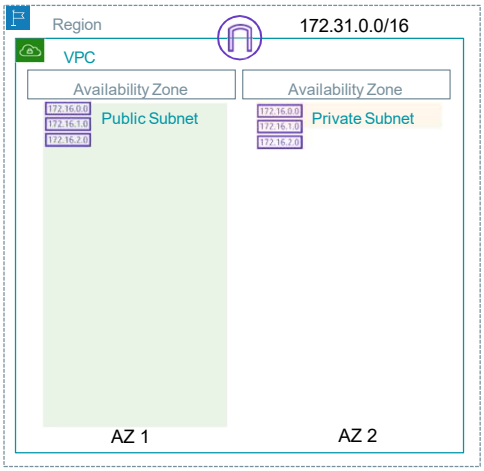


Subnets

- Private Subnet
  - No route to Internet
  - Instances receive private IPs
  - Typically uses NAT for instances to have internet access
  - Ex: Database, App server

Destination	Target
172.31.0.0/16	local

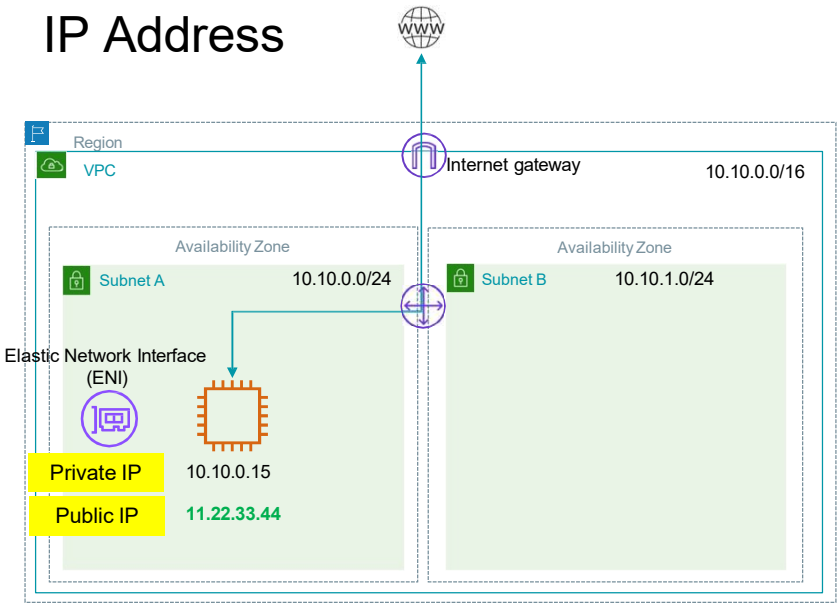
1 Subnet => 1 AZ



Subnets – Good to know

- AWS reserves 5 IPs address (first 4 and last 1 IP address) in each Subnet
- These 5 IPs are not available for use and cannot be assigned to an instance
- Example: if CIDR block 10.0.0.0/24, reserved IPs are:
  - 10.0.0.0: Network address
  - 10.0.0.1: Reserved by AWS for the VPC router
  - 10.0.0.2: Reserved by AWS for mapping to Amazon-provided DNS
  - 10.0.0.3: Reserved by AWS for future use
  - 10.0.0.255: Network broadcast address. AWS does not support broadcast in a VPC, therefore this address is reserved

IP Address



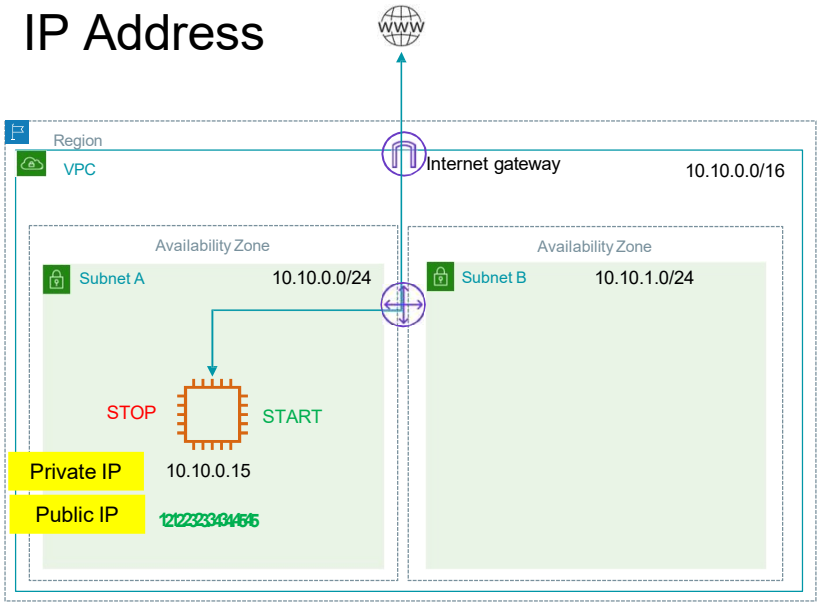
Private IP

- Private IP is assigned from the subnet range

Public IP

- Public IP is assigned from the Amazon's pool of Public IPs

IP Address



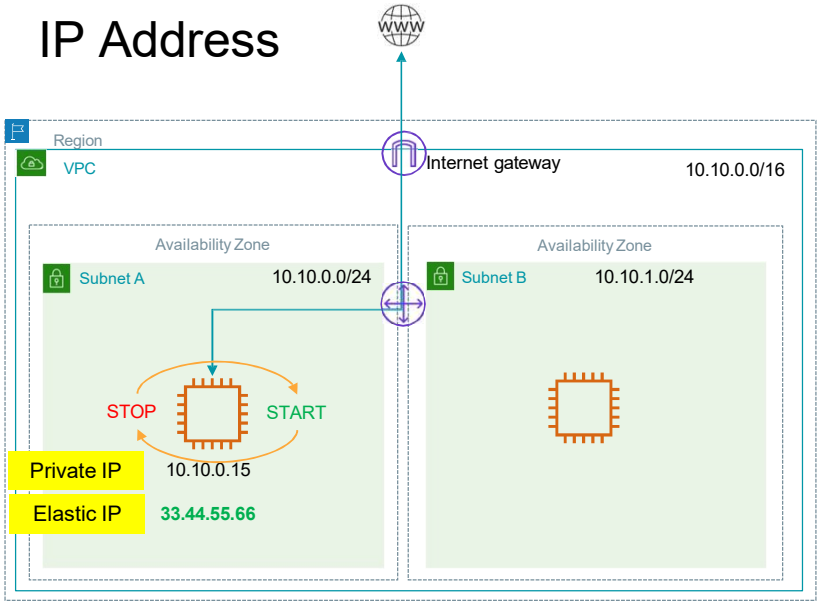
Private IP

- Private IP is assigned from the subnet range

Public IP

- Public IP is assigned from the Amazon's pool of Public IPs
- Public IP will change when you stop and start the instance

IP Address



Elastic IP

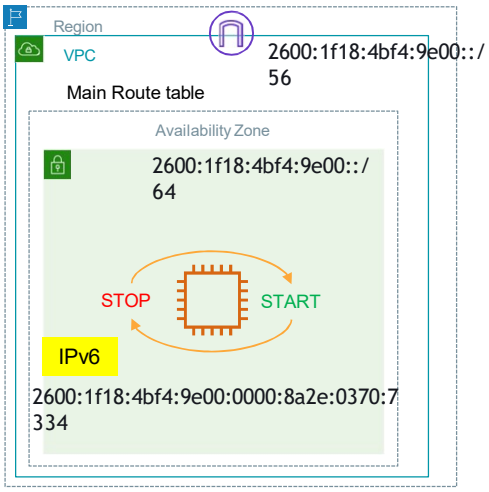
- Elastic IP is allocated to your AWS account
- Remains allocated until released
- Does not change on instance stop/start
- Can be remapped to another instance

Private, Public and Elastic IP (EIP)

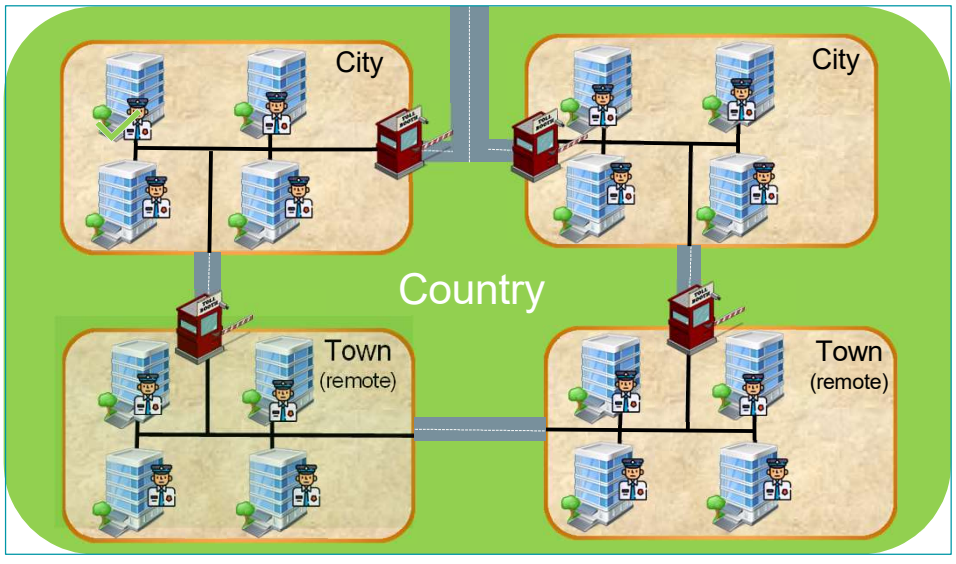
Feature	Private	Public	Elastic
Communication	Communication within VPC	Can communicate over internet	Can communicate over internet
Address range	Gets IP address from subnet range. Ex: 10.200.0.1	Gets IP address from Amazon Pool within region	Gets IP address from Amazon Pool within region
Instance stop/start behaviour	Once assigned cannot be changed	Changes over instance stop and start (not on instance OS reboot)	Do not change over instance stop and start.
Releasing IP	Released when instance is terminated	Released to pool when instance is stopped or terminated.	Not released. Remains in your account. (Billed)
Automatic Assignment	Receives private IP on launch of EC2 instance	Receives public IP on launch of EC2 instance if "Public IP addressing attribute" is set to true for subnet.	Have to explicitly allocate and attach EIP to EC2 instance. Can be reattached to other EC2.
Examples	Application servers, databases	Web servers, Load Balancers,	Web servers, Load Balancers, Websites

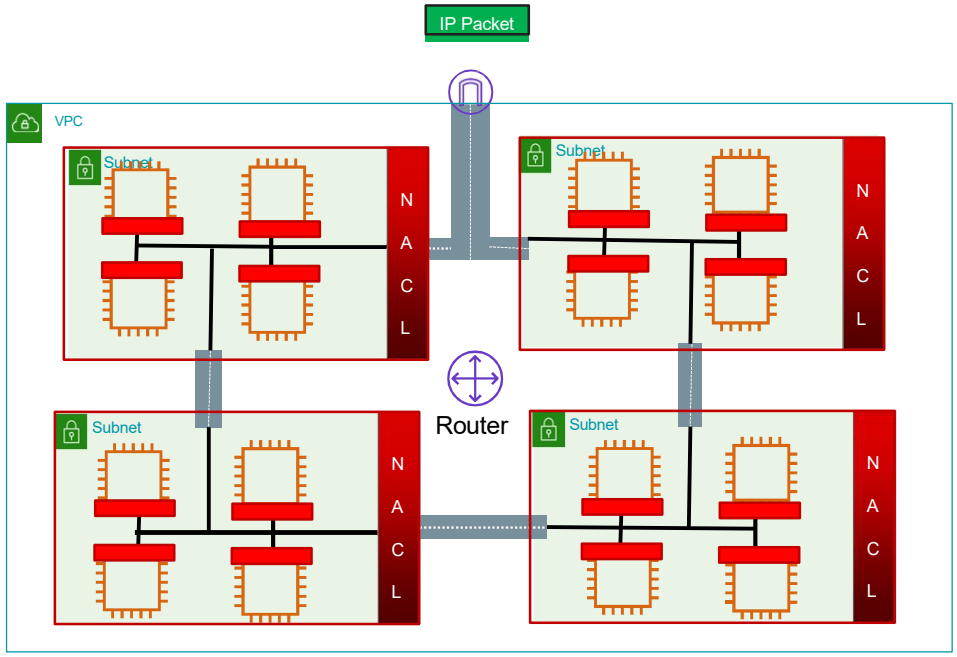
IPv6 Addresses

- AWS VPC also supports IPv6 addresses
- IPv6 address is 128 bits in size with 8 blocks of 16 bits each
  - Example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
- VPC CIDR with prefix /56 (2^72 IPs) and Subnet CIDR prefix /64
- IPv6 addresses are public and globally unique, and allows resources to communicate with the internet
- VPC can operate in dual-stack mode where VPC resources can communicate over IPv4, or IPv6, or both
- IPv6 address persists when you stop and start your instance, and is released when you terminate your instance



Routing vs Firewall

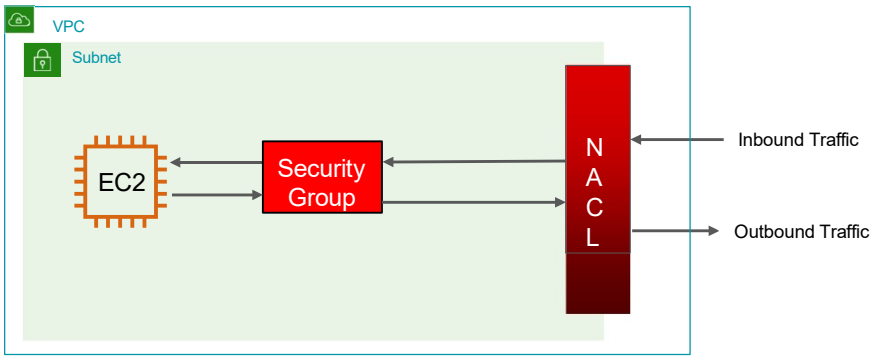




VPC Firewall - Security group

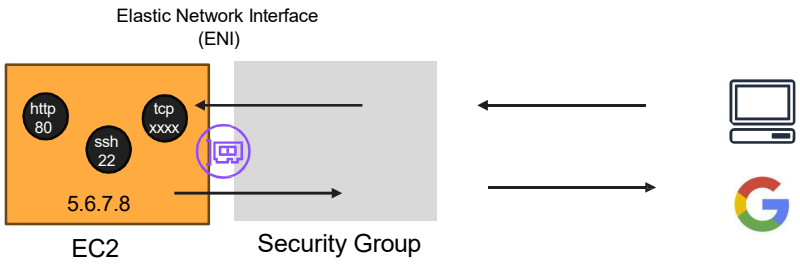
Firewalls inside VPC

- Security Groups
- Network Access Control List (NACL)

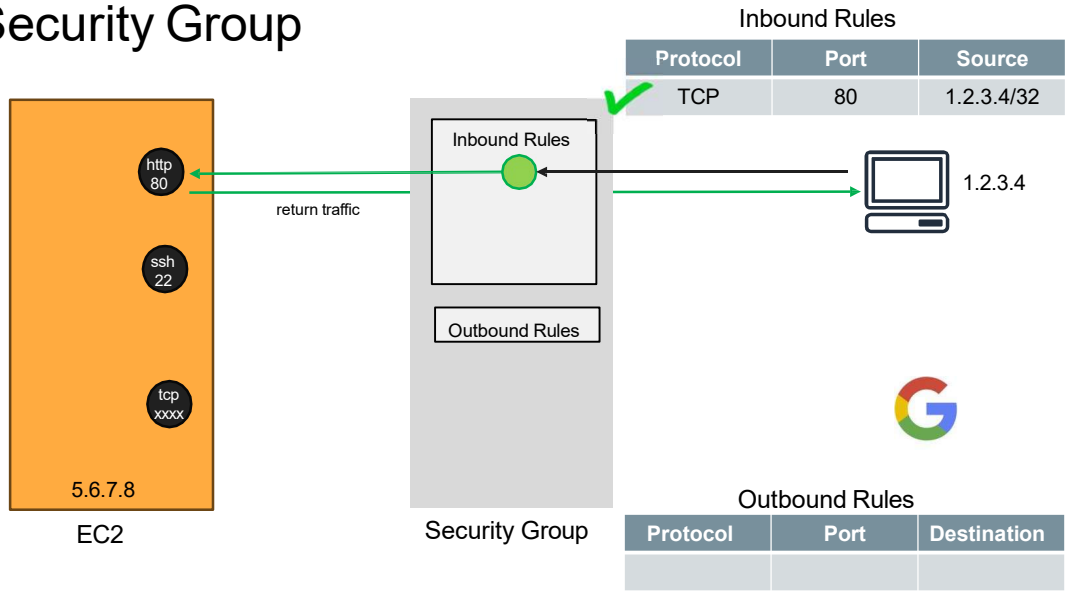


Security Group

- Security Groups are most basic, native and important firewall for EC2 instances (ENIs)
- Security group has Inbound and Outbound rules
- Security group has only ALLOW rules. Does not support DENY/Block rules.
- Default Security group in each VPC
- Authorises traffic for both IPv4 and IPv6 traffic
- Security groups are stateful
- You can reference another Security group as a source

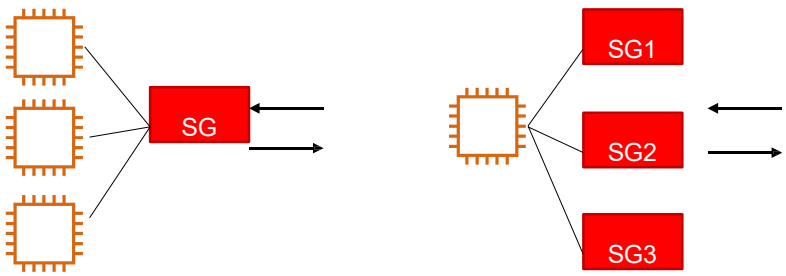


Security Group



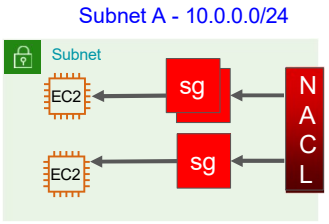
Security Groups - Summary

- Single Security Group can be attached to multiple instances
- Single Instance can have multiple Security groups
- All inbound traffic is blocked by default
- All outbound traffic is authorised by default
- Authorises traffic for both IPv4 and IPv6 traffic
- Security groups are stateful
- You can reference another Security group as a source



Network Access Control List (NACL)

- Works at Subnet level – Hence automatically applied to all instances
- Contains both Allow and Deny rules. Rules are numbered.
- Rules are evaluated in the order of rule number (1 to 32766)
- Stateless – We need to explicitly open ports for return traffic
- Default NACL allows all inbound and outbound traffic
- **NACL are a great way of blocking a specific IP at the subnet level**



Network ACL inbound rules

#Rule	Type	Protocol	Port	Source	Allow/Deny
100	All IPv4 traffic	All	All	180.151.138.43/32	DENY
101	HTTPS	TCP	443	0.0.0.0/0	ALLOW
*	All IPv4 traffic	All	All	0.0.0.0/0	DENY

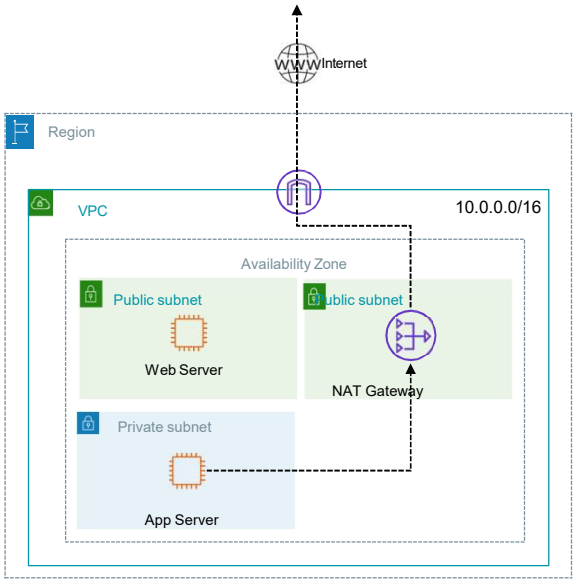
Security Groups vs Network ACL

Security Group	Network ACL
Operates at EC2 instance or ENI level	Operates at Subnet level
Supports only Allow rules	Supports both Allow and Deny rules
Stateful – Return traffic is allowed	Stateless – Return traffic needs to be authorized in Outbound rules
All rules are evaluated before making a decision	Rules are evaluated in the order (lower to higher) and first matching rule is applied

Refer VPC quota: https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html

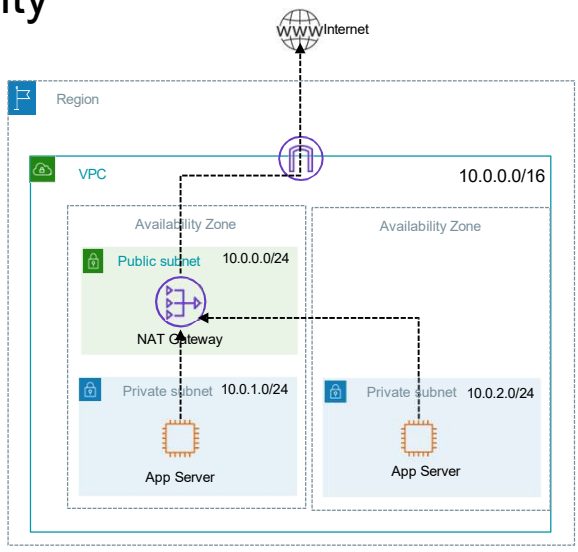
NAT Gateway

- AWS managed NAT, higher bandwidth, better availability, no admin
- Pay by the hour for usage and bandwidth
- 5 Gbps of bandwidth with automatic scaling up to 100 Gbps
- No security groups
- NACL at subnet level applies to NAT Gateway.
- Supported protocols: TCP, UDP, and ICMP
- Uses ports 1024–65535 for outbound connection
- For outbound internet access, NAT Gateway should be created in Public Subnet so that it can communicate with the Internet
- NAT Gateway should be allocated Elastic IP



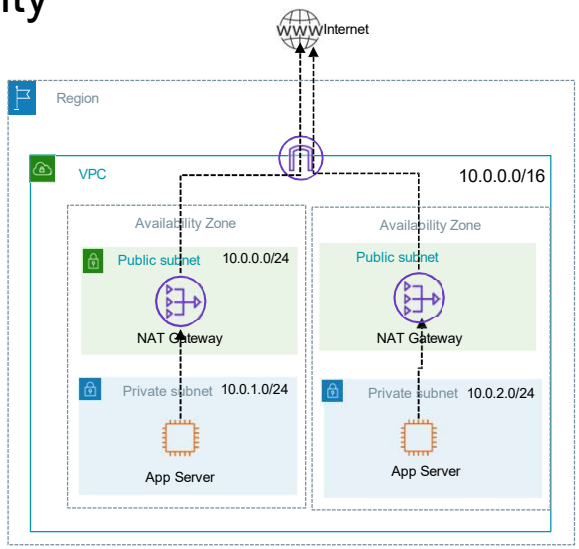
NAT Gateway High availability

- NAT Gateways are highly available within a single AZ
- For HA across multiple AZs, multiple NAT gateways can be launched



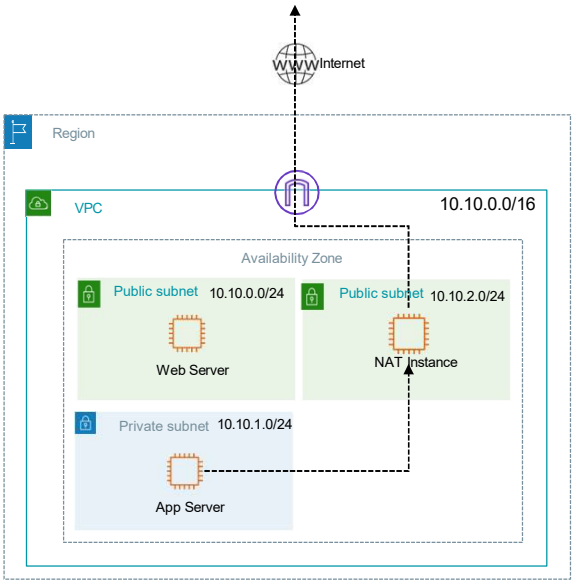
NAT Gateway High availability

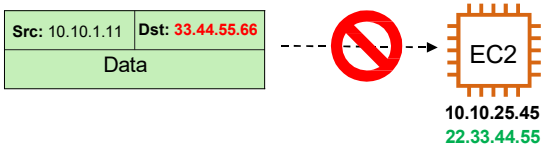
- NAT Gateways are highly available within a single AZ
- For HA across multiple AZs, multiple NAT gateways can be launched



NAT Instance

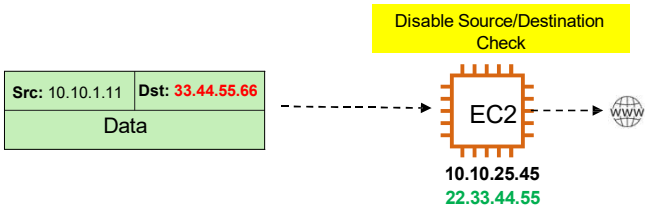
- Must be in a Public Subnet
- Must have Public or Elastic IP
- Should be launched using AWS provided NAT AMIs
- **Disable Source/Destination Check**

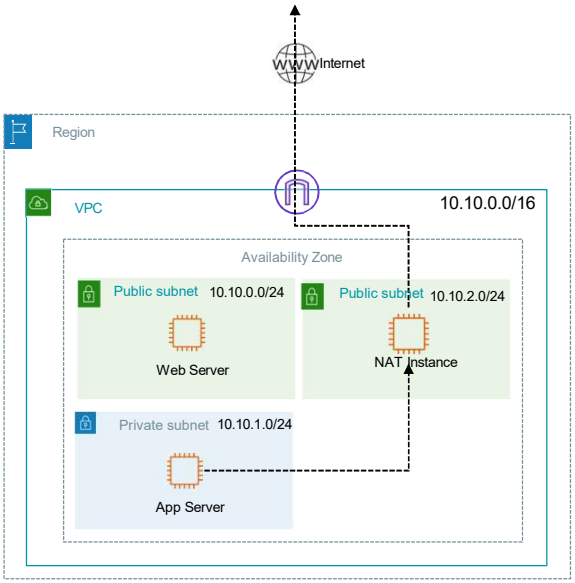




NAT Instance

- Must be in a Public Subnet
- Must have Public or Elastic IP
- Should be launched using AWS provided NAT AMIs
- **Disable Source/Destination Check**



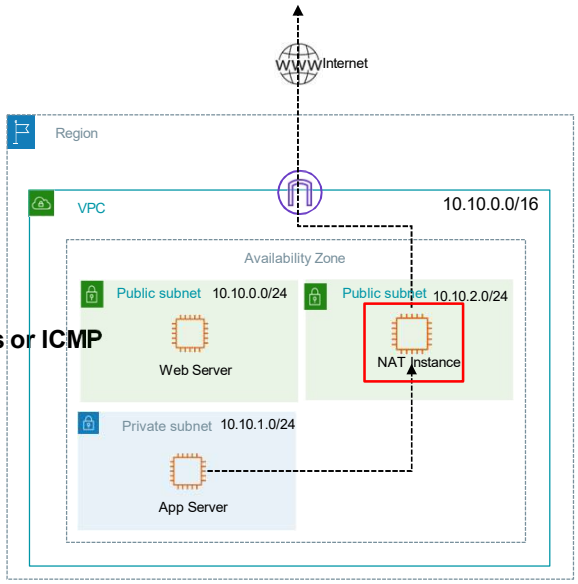


NAT Instance

- Must be in a Public Subnet
- Must have Public or Elastic IP
- Should be launched using AWS provided NAT AMIs
- Disable Source/Destination Check
- **Security group inbound rules to allow http/s or ICMP (ping) traffic from Private subnets**

Protocol	Port	Source
HTTP	80	10.10.1.0/24
HTTPS	443	10.10.1.0/24
All ICMP - IPv4	All	10.10.1.0/24

Outbound Rules

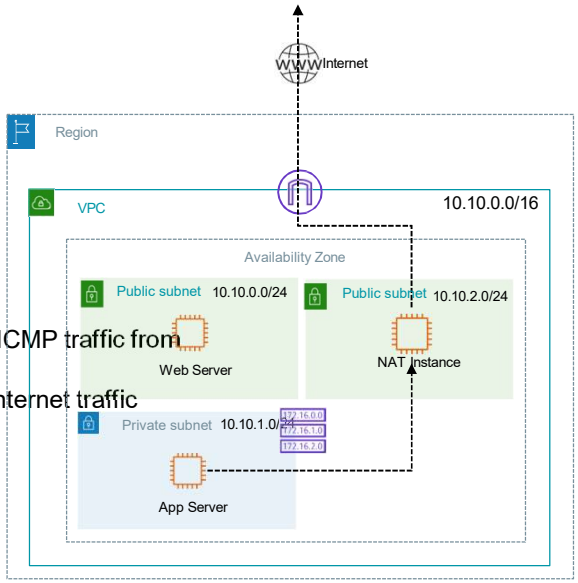


NAT Instance

- Must be in a Public Subnet
- Must have Public or Elastic IP
- Should be launched using AWS provided NAT AMIs
- Disable Source/Destination Check
- Security group inbound rules to allow http/s or ICMP traffic from Private subnet
- Update Private subnet route tables and route internet traffic through NAT EC2 instance

Private Subnet Route Table

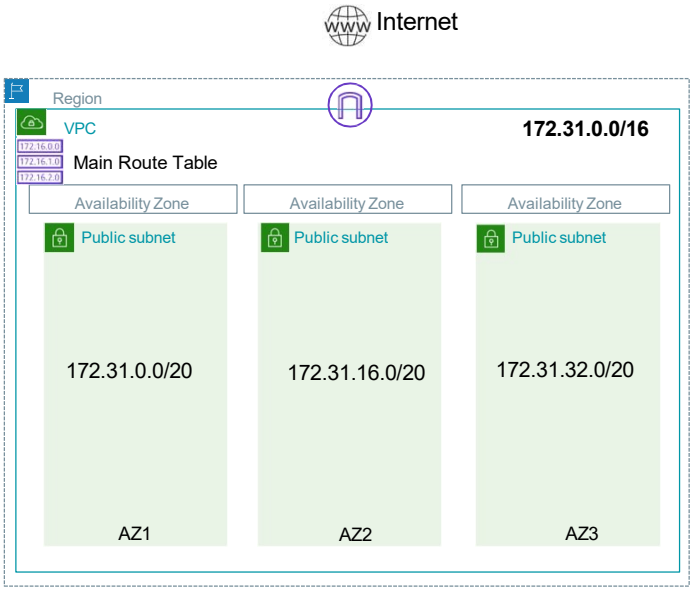
Destination	Target
10.10.0.0/16	Local
0.0.0.0/0	**Nat Instance**



Default VPC

- AWS Creates Default VPC in each AWS region
- Creates VPC with CIDR - 172.31.0.0/16
- Creates Subnets in every AZ with CIDR /20
- Creates Internet Gateway
- Main route table with route to Internet which make all subnets public
- If deleted, you can recreate default VPC

Destination	Target
172.31.0.0/16	local
0.0.0.0/0	igw-xxxxxx

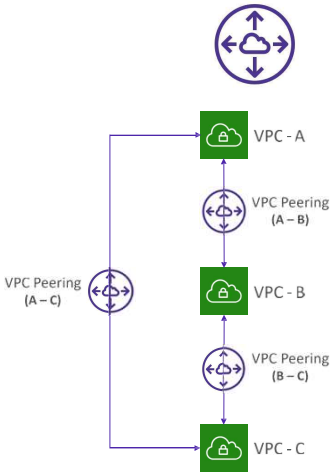


VPC connectivity options

- VPC Endpoints & PrivateLink
- VPC Peering connection
- Transit Gateway
- Site-2-Site VPN
- Client VPN
- Direct Connect

VPC Peering

- Privately connect two VPCs using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDRs
- VPC Peering connection is NOT transitive (must be established for each VPC that need to communicate with one another)
- You must update route tables in each VPC's subnets to ensure EC2 instances can communicate with each other



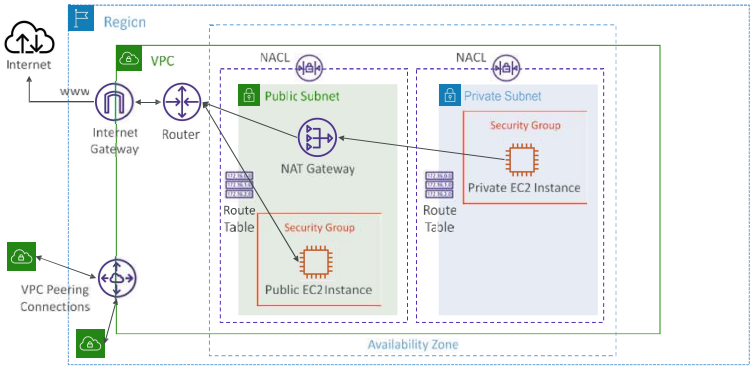
VPC Peering – Good to know

- You can create VPC Peering connection between VPCs in **different AWS accounts/regions**
- You can reference a security group in a peered VPC (works cross accounts – same region)

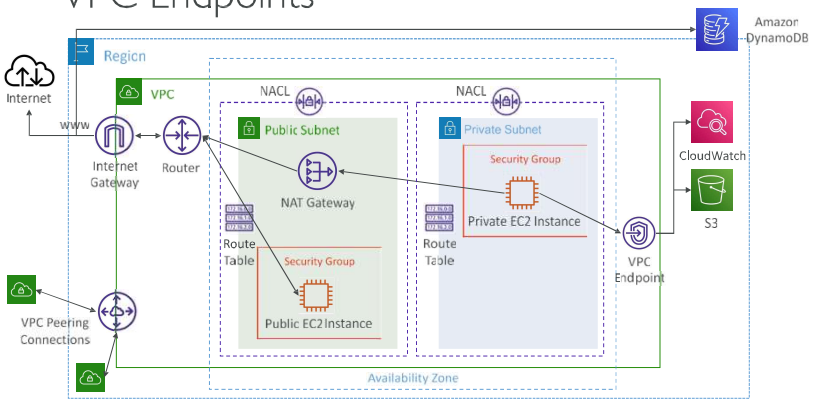
Type	Protocol	Port range	Source
HTTP	TCP	80	sg-04991f9af3473b939 / default
HTTP	TCP	80	[redacted] / sg-027ad1f7865d4be76

Account ID

VPC Peering

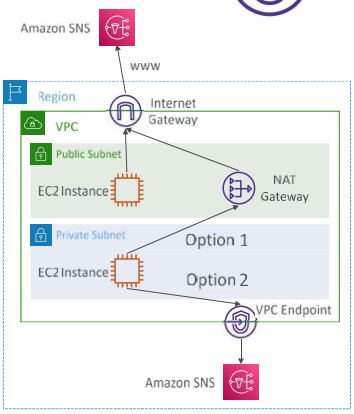


VPC Endpoints



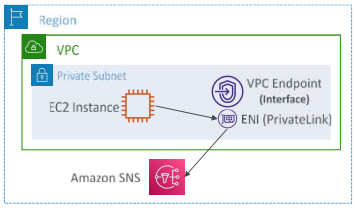
VPC Endpoints (AWS PrivateLink)

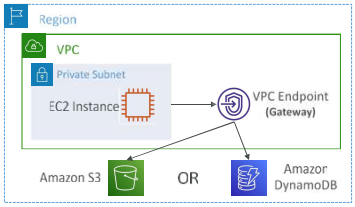
- Every AWS service is publicly exposed (public URL)
- VPC Endpoints (powered by AWS PrivateLink) allows you to connect to AWS services using a **private network** instead of using the public Internet
- They're redundant and scale horizontally
- They remove the need of IGW, NATGW, … to access AWS Services
- In case of issues:
  - Check DNS Setting Resolution in your VPC
  - Check Route Tables



Types of Endpoints

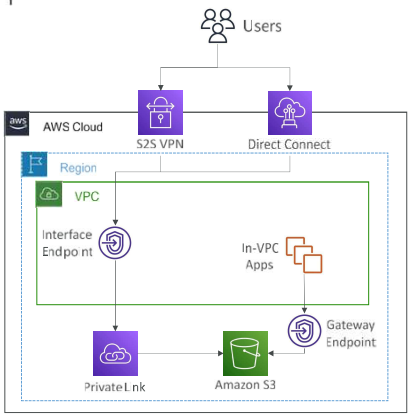
- Interface Endpoints (powered by PrivateLink)
  - Provisions an ENI (private IP address) as an entry point (must attach a Security Group)
  - Supports most AWS services
  - $ per hour + $ per GB of data processed
- Gateway Endpoints
  - Provisions a gateway and must be used as a target in a route table (does not use security groups)
  - Supports both S3 and DynamoDB
  - Free





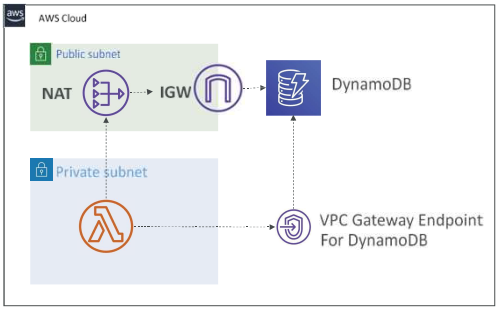
Gateway or Interface Endpoint for S3?

- Gateway is most likely going to be preferred all the time at the exam
- Cost: free for Gateway, $ for interface endpoint
- Interface Endpoint is preferred access is required from on-premises (Site to Site VPN or Direct Connect), a different VPC or a different region



Lambda in VPC accessing DynamoDB

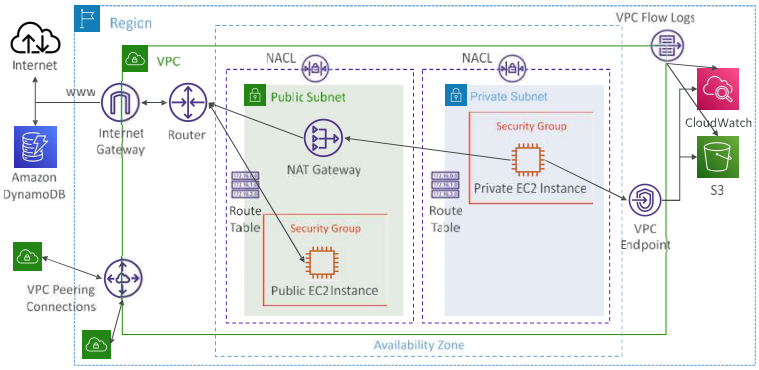
- DynamoDB is a public service from AWS
- Option 1: Access from the public internet
  - Because Lambda is in a VPC, it needs a NAT Gateway in a public subnet and an internet gateway
- Option 2 (better & free): Access from the private VPC network
  - Deploy a VPC Gateway endpoint for DynamoDB
  - Change the Route Tables



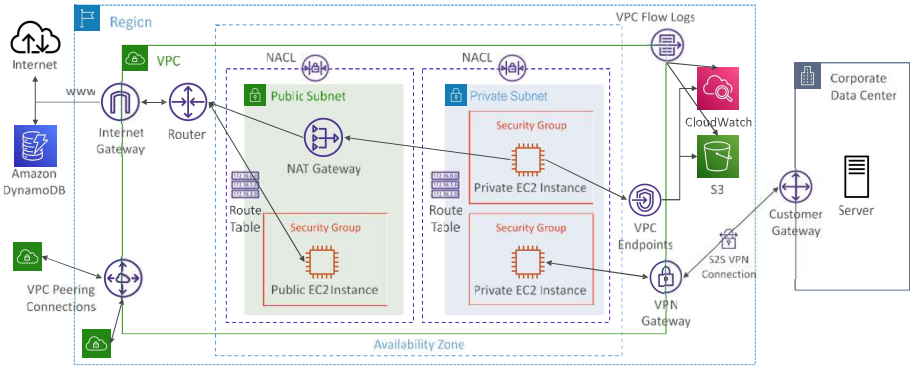
VPC Flow Logs

- Capture information about IP traffic going into your interfaces:
  - VPC Flow Logs
  - Subnet Flow Logs
  - Elastic Network Interface (ENI) Flow Logs
- Helps to monitor & troubleshoot connectivity issues
- Flow logs data can go to S3, CloudWatch Logs, and Kinesis Data Firehose
- Captures network information from AWS managed interfaces too: ELB, RDS, ElastiCache, Redshift, WorkSpaces, NATGW, Transit Gateway…

VPC Flow Logs



AWS Site-to-Site VPN

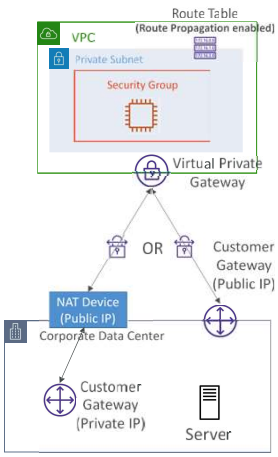


AWS Site-to-Site VPN

- Virtual Private Gateway (VGW)
  - VPN concentrator on the AWS side of the VPN connection
  - VGW is created and attached to the VPC from which you want to create the Site-to-Site VPN connection
  - Possibility to customize the ASN (Autonomous System Number)
- Customer Gateway (CGW)
  - Software application or physical device on customer side of the VPN connection
  - https://docs.aws.amazon.com/vpn/latest/s2svpn/your-cgw.html#DevicesTested

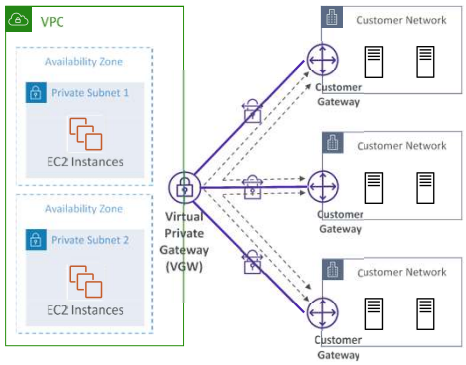
Site-to-Site VPN Connections

- Customer Gateway Device (On-premises)
  - What IP address to use?
    - Public Internet-routable IP address for your Customer Gateway device
    - If it's behind a NAT device that's enabled for NAT traversal (NAT-T), use the public IP address of the NAT device
- Important step: enable Route Propagation for the Virtual Private Gateway in the route table that is associated with your subnets
- If you need to ping your EC2 instances from on-premises, make sure you add the ICMP protocol on the inbound of your security groups

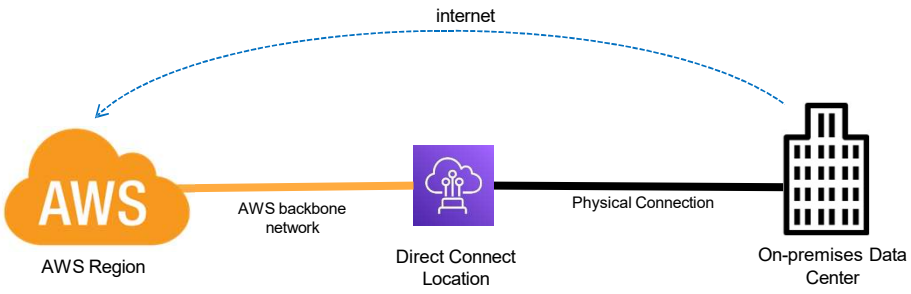


AWS VPN CloudHub

- Provide secure communication between multiple sites, if you have multiple VPN connections
- Low-cost hub-and-spoke model for primary or secondary network connectivity between different locations (VPN only)
- It's a VPN connection so it goes over the public Internet
- To set it up, connect multiple VPN connections on the same VGW, setup dynamic routing and configure route tables



Direct Connect (DX)

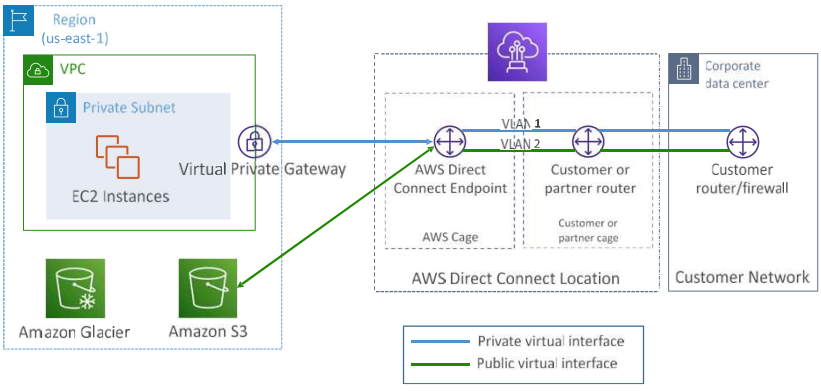


- Provides dedicated and consistent network
- Provides network bandwidth from 50 Mbps up to 100 Gbps over a single connection
- Low data transfer cost

Direct Connect (DX)

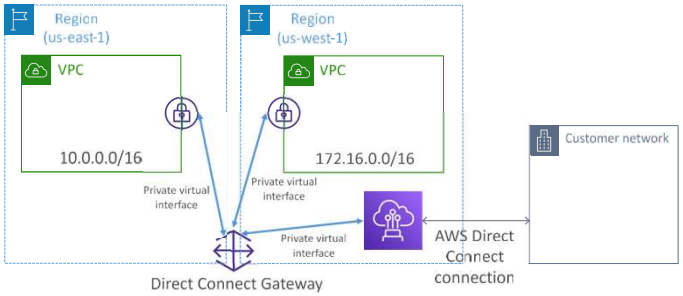
- Provides a dedicated private connection from a remote network to your VPC
- Dedicated connection must be setup between your DC and AWS Direct Connect locations
- You need to setup a Virtual Private Gateway on your VPC
- Access public resources (S3) and private (EC2) on same connection
- Use Cases:
  - Increase bandwidth throughput - working with large data sets – lower cost
  - More consistent network experience - applications using real-time data feeds
  - Hybrid Environments (on prem + cloud)
- Supports both IPv4 and IPv6

Direct Connect Diagram



Direct Connect Gateway

- If you want to setup a Direct Connect to one or more VPC in many different regions (same account), you must use a Direct Connect Gateway

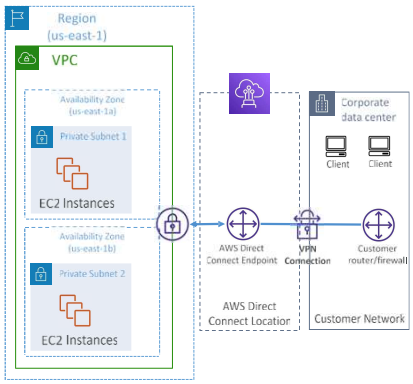


Direct Connect – Connection Types

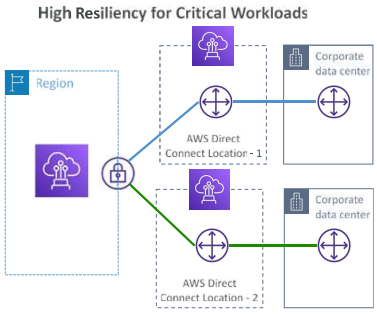
- **Dedicated Connections**: 1Gbps, 10 Gbps and 100 Gbps capacity
  - Physical ethernet port dedicated to a customer
  - Request made to AWS first, then completed by AWS Direct Connect Partners
- **Hosted Connections**: 50Mbps, 500 Mbps, to 10 Gbps
  - Connection requests are made via AWS Direct Connect Partners
  - Capacity can be added or removed on demand
  - 1, 2, 5, 10 Gbps available at select AWS Direct Connect Partners
- Lead times are often longer than 1 month to establish a new connection

Direct Connect – Encryption

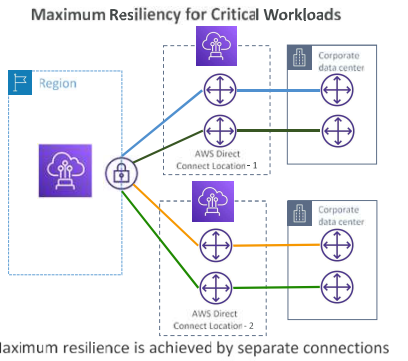
- Data in transit is not encrypted but is private
- AWS Direct Connect + VPN provides an IPsec-encrypted private connection
- Good for an extra level of security, but slightly more complex to put in place



Direct Connect - Resiliency

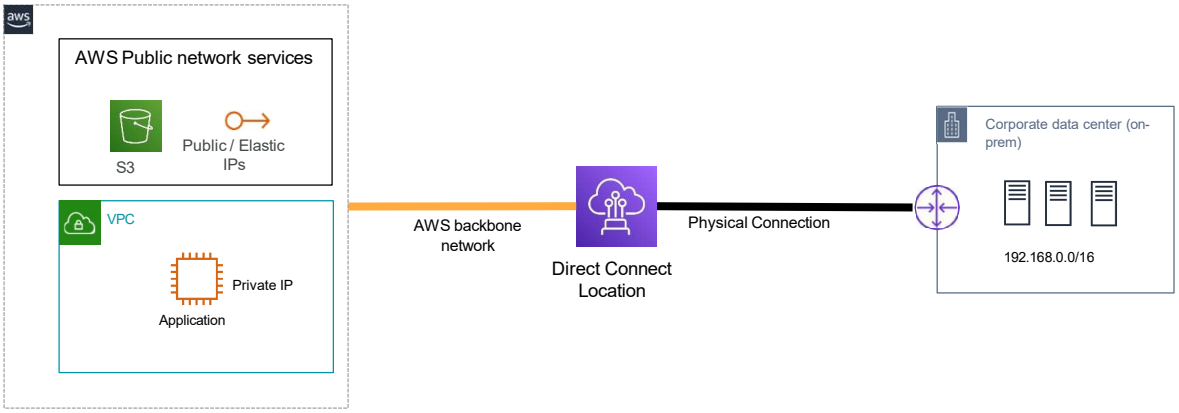


One connection at multiple locations



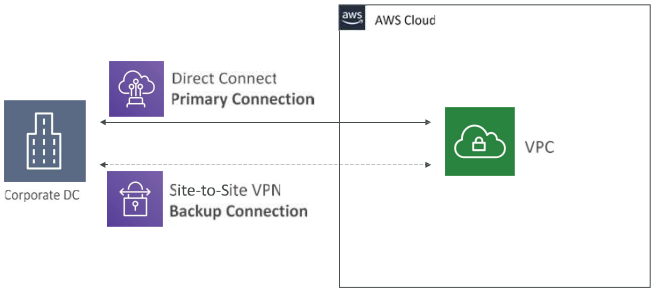
Maximum resilience is achieved by separate connections terminating on separate devices in more than one location.

Direct Connect



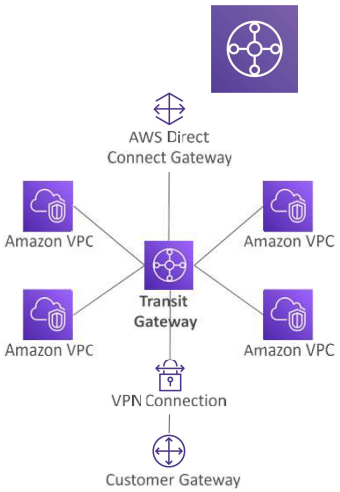
Site-to-Site VPN connection as a backup

- In case Direct Connect fails, you can set up a backup Direct Connect connection (expensive), or a Site-to-Site VPN connection

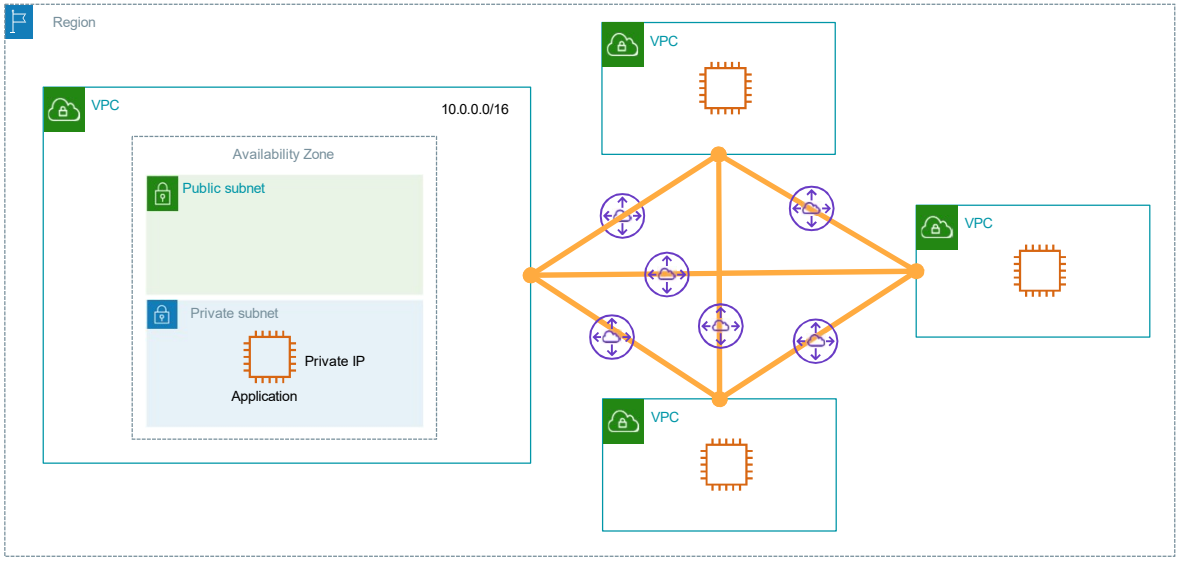


Transit Gateway

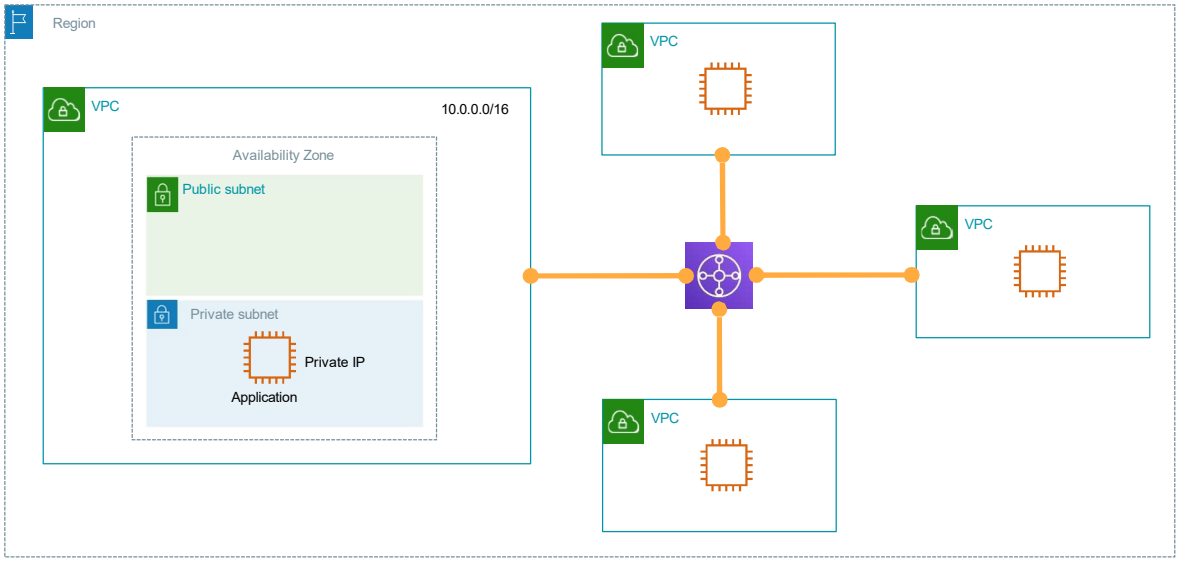
- For having transitive peering between thousands of VPC and on-premises, hub-and-spoke (star) connection
- Regional resource, can work cross-region
- Share cross-account using Resource Access Manager (RAM)
- You can peer Transit Gateways across regions
- Route Tables: limit which VPC can talk with other VPC
- Works with Direct Connect Gateway, VPN connections
- Supports IP Multicast (not supported by any other AWS service)



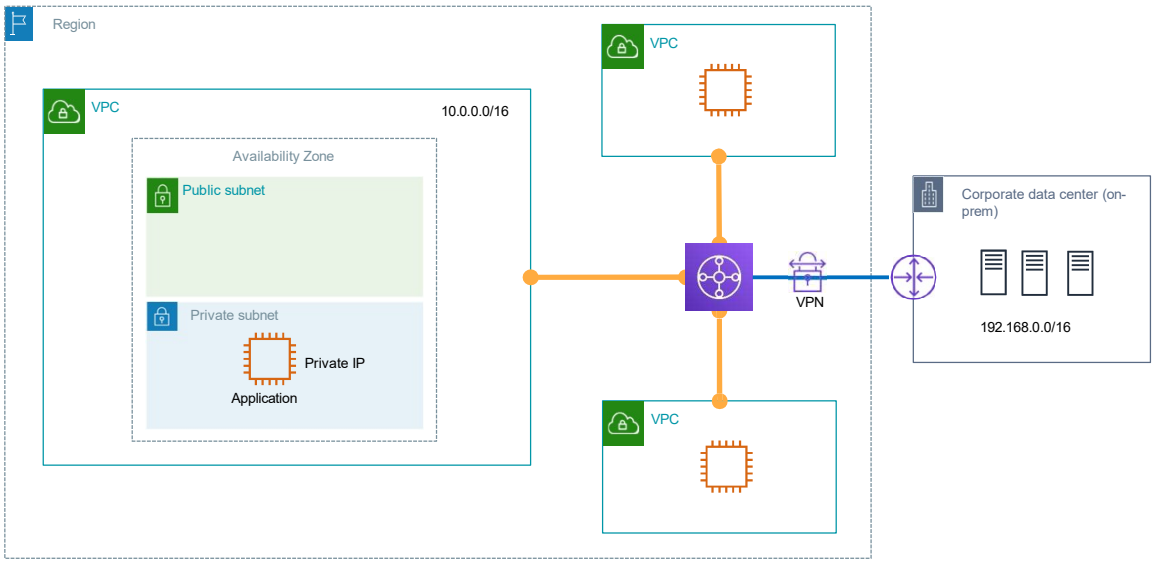
Transit Gateway



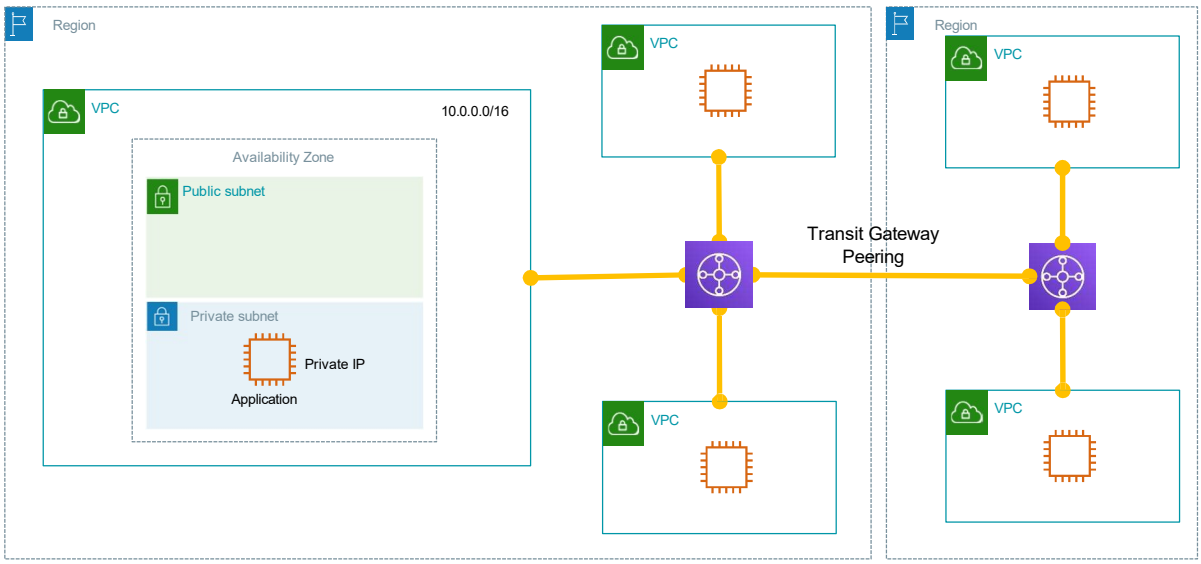
Transit Gateway



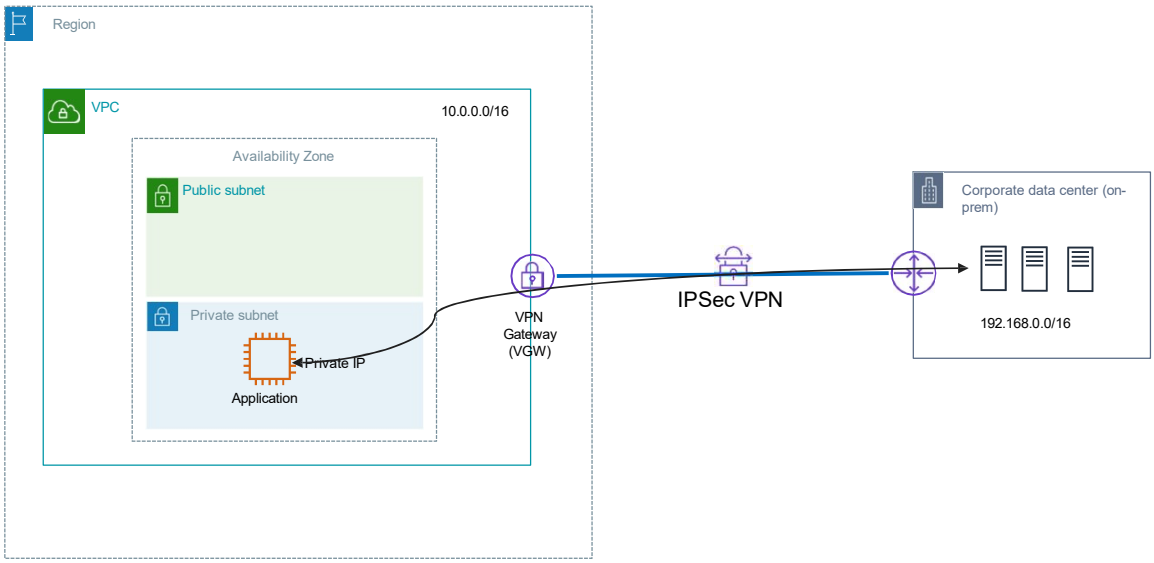
Transit Gateway with VPN attachment



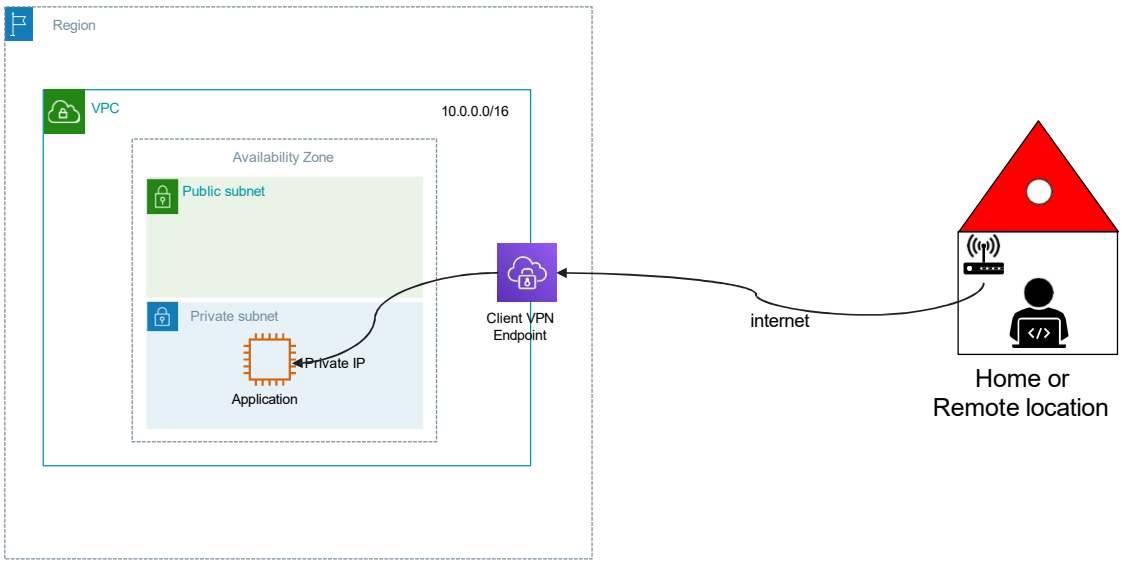
Transit Gateway peering

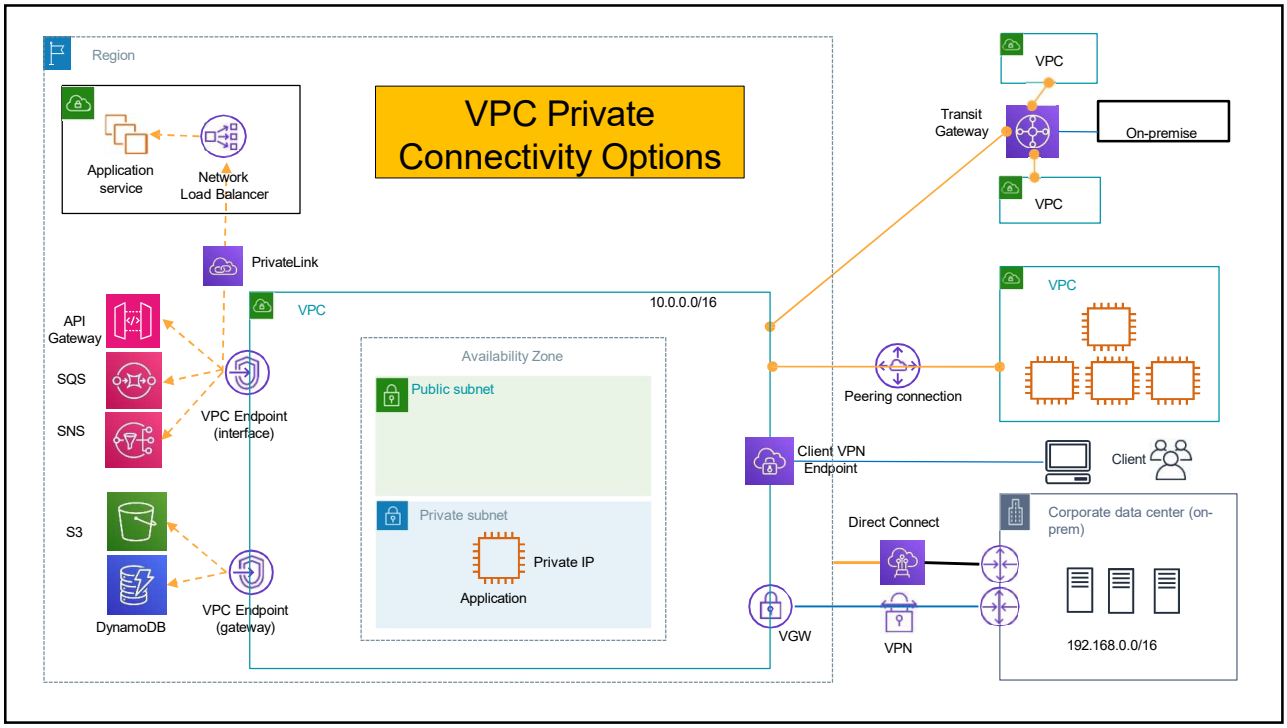


Site to Site VPN



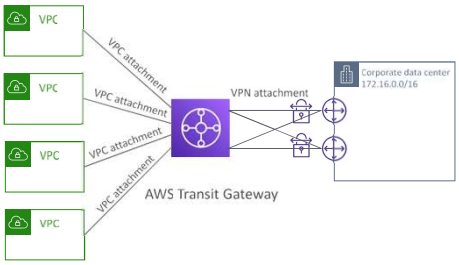
Client to Site VPN



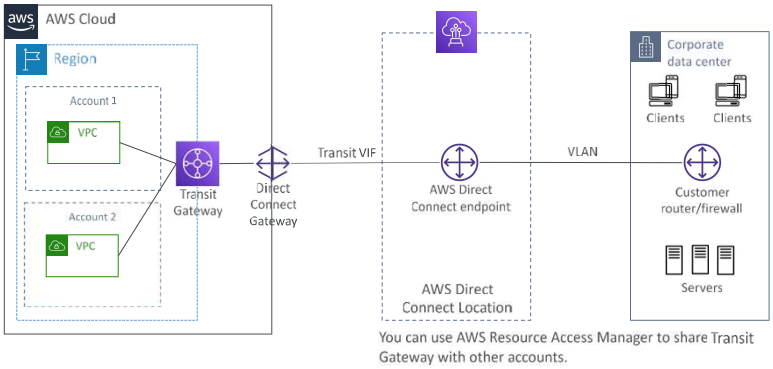


Transit Gateway: Site-to-Site VPN ECMP

- ECMP = Equal-cost multi-path routing
- Routing strategy to allow to forward a packet over multiple best path
- Use case: create multiple Site-to-Site VPN connections to increase the bandwidth of your connection to AWS



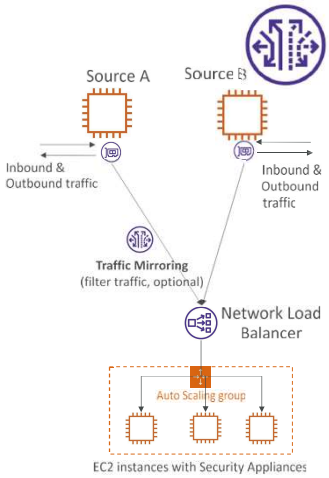
Transit Gateway – Share Direct Connect between multiple accounts



You can use AWS Resource Access Manager to share Transit Gateway with other accounts.

VPC – Traffic Mirroring

- Allows you to capture and inspect network traffic in your VPC
- Route the traffic to security appliances that you manage
- Capture the traffic
  - From (Source) – ENIs
  - To (Targets) – an ENI or a Network Load Balancer
- Capture all packets or capture the packets of your interest (optionally, truncate packets)
- Source and Target can be in the same VPC or different VPCs (VPC Peering)
- Use cases: content inspection, threat monitoring, troubleshooting, …



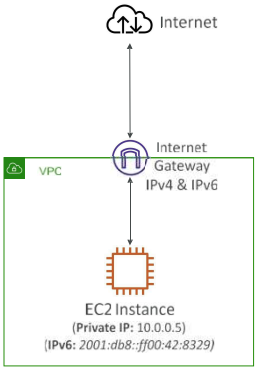
What is IPv6?

- IPv4 designed to provide 4.3 Billion addresses (they'll be exhausted soon)

- IPv6 is the successor of IPv4
- IPv6 is designed to provide 3.4×10^{38} unique IP addresses
- Every IPv6 address in AWS is public and Internet-routable (no private range)
- Format ➔ x.x.x.x.x.x.x.x (x is hexadecimal, range can be from 0000 to ffff)
- Examples:
  - 2001:db8:3333:4444:5555:6666:7777:8888
  - 2001:db8:3333:4444:cccc:dddd:eeee:ffff
  - :: ➔ all 8 segments are zero
  - 2001:db8:: ➔ the last 6 segments are zero
  - ::1234:5678 ➔ the first 6 segments are zero
  - 2001:db8::1234:5678 ➔ the middle 4 segments are zero

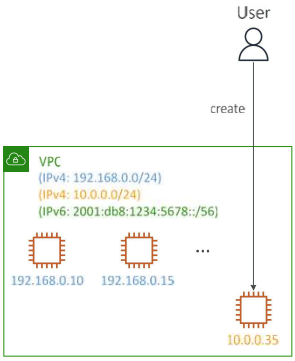
IPv6 in VPC

- IPv4 cannot be disabled for your VPC and subnets
- You can enable IPv6 (they're public IP addresses) to operate in dual-stack mode
- Your EC2 instances will get at least a private internal IPv4 and a public IPv6
- They can communicate using either IPv4 or IPv6 to the internet through an Internet Gateway



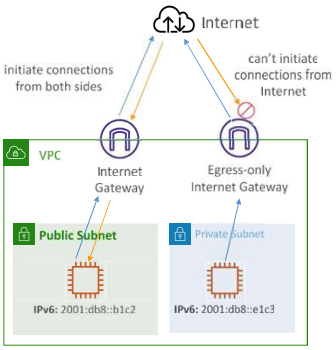
IPv4 Troubleshooting

- IPv4 cannot be disabled for your VPC and subnets
- So, if you cannot launch an EC2 instance in your subnet
  - It's not because it cannot acquire an IPv6 (the space is very large)
  - It's because there are no available IPv4 in your subnet
- Solution: create a new IPv4 CIDR in your subnet



Egress-only Internet Gateway

- Used for IPv6 only
- (similar to a NAT Gateway but for IPv6)
- Allows instances in your VPC outbound connections over IPv6 while preventing the internet to initiate an IPv6 connection to your instances
- You must update the Route Tables



VPC Section Summary (1/3)

- CIDR – IP Range
- VPC – Virtual Private Cloud => we define a list of IPv4 & IPv6 CIDR
- Subnets – tied to an AZ, we define a CIDR
- Internet Gateway – at the VPC level, provide IPv4 & IPv6 Internet Access
- Route Tables – must be edited to add routes from subnets to the IGW, VPC Peering Connections, VPC Endpoints, …
- Bastion Host – public EC2 instance to SSH into, that has SSH connectivity to EC2 instances in private subnets
- NAT Instances – gives Internet access to EC2 instances in private subnets. Old, must be setup in a public subnet, disable Source / Destination check flag
- NAT Gateway – managed by AWS, provides scalable Internet access to private EC2 instances, when the target is an IPv4 address

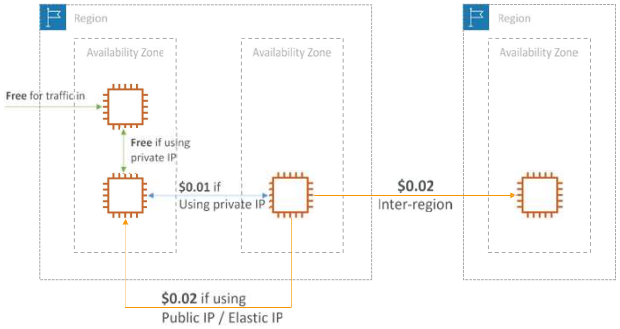
VPC Section Summary (2/3)

- NACL – stateless, subnet rules for inbound and outbound, don't forget Ephemeral Ports
- Security Groups – stateful, operate at the EC2 instance level
- VPC Peering – connect two VPCs with non overlapping CIDR, non-transitive
- VPC Endpoints – provide private access to AWS Services (S3, DynamoDB, CloudFormation, SSM) within a VPC
- VPC Flow Logs – can be setup at the VPC / Subnet / ENI Level, for ACCEPT and REJECT traffic, helps identifying attacks, analyze using Athena or CloudWatch Logs Insights
- Site-to-Site VPN – setup a Customer Gateway on DC, a Virtual Private Gateway on VPC, and site-to-site VPN over public Internet
- AWS VPN CloudHub – hub-and-spoke VPN model to connect your sites

VPC Section Summary (3/3)

- Direct Connect – setup a Virtual Private Gateway on VPC, and establish a direct private connection to an AWS Direct Connect Location
- Direct Connect Gateway – setup a Direct Connect to many VPCs in different AWS regions
- AWS PrivateLink / VPC Endpoint Services:
  - Connect services privately from your service VPC to customers VPC
  - Doesn't need VPC Peering, public Internet, NAT Gateway, Route Tables
  - Must be used with Network Load Balancer & ENI
- ClassicLink – connect EC2-Classic EC2 instances privately to your VPC
- Transit Gateway – transitive peering connections for VPC, VPN & DX
- Traffic Mirroring – copy network traffic from ENIs for further analysis
- Egress-only Internet Gateway – like a NAT Gateway, but for IPv6 targets

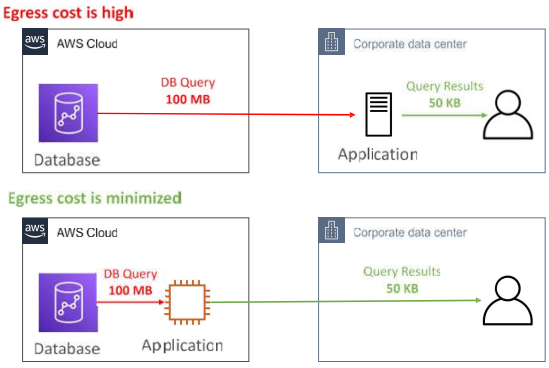
Networking Costs in AWS per GB - Simplified



- Use Private IP instead of Public IP for good savings and better network performance
- Use same AZ for maximum savings (at the cost of high availability)

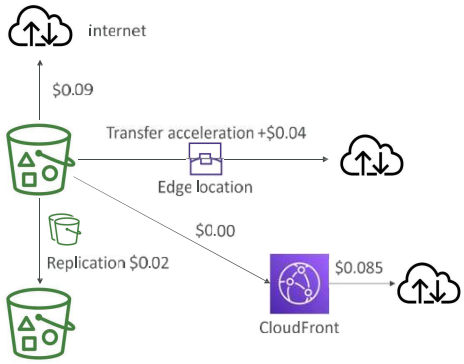
Minimizing egress traffic network cost

- Egress traffic: outbound traffic (from AWS to outside)
- Ingress traffic: inbound traffic - from outside to AWS (typically free)
- Try to keep as much internet traffic within AWS to minimize costs
- Direct Connect location that are co-located in the same AWS Region result in lower cost for egress network



S3 Data Transfer Pricing – Analysis for USA

- S3 ingress: free
- S3 to Internet: $0.09 per GB
- S3 Transfer Acceleration:
  - Faster transfer times (50 to 500% better)
  - Additional cost on top of Data Transfer Pricing: +$0.04 to $0.08 per GB
- S3 to CloudFront: $0.00 per GB
- CloudFront to Internet: $0.085 per GB (slightly cheaper than S3)
  - Caching capability (lower latency)
  - Reduce costs associated with S3 Requests Pricing (7x cheaper with CloudFront)
- S3 Cross Region Replication: $0.02 per GB



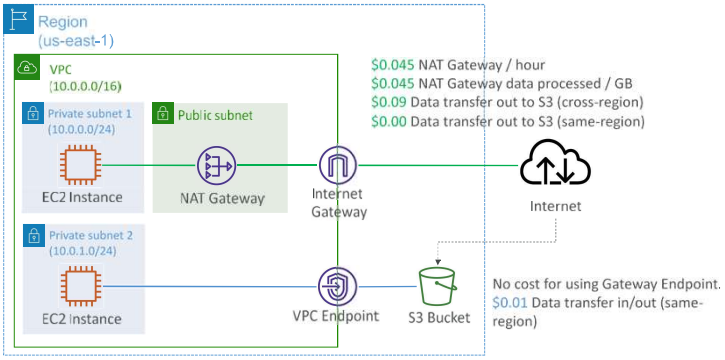
Pricing: NAT Gateway vs Gateway VPC Endpoint

Subnet 1 route table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

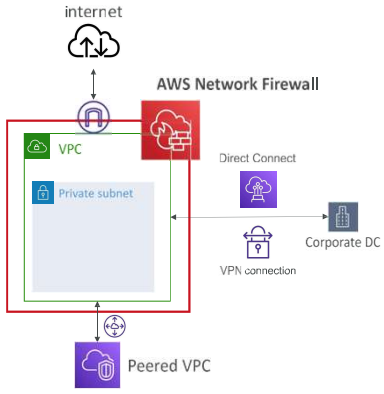
Subnet 2 route table

Destination	Target
10.0.0.0/16	Local
pl-id for Amazon S3	vpce-id

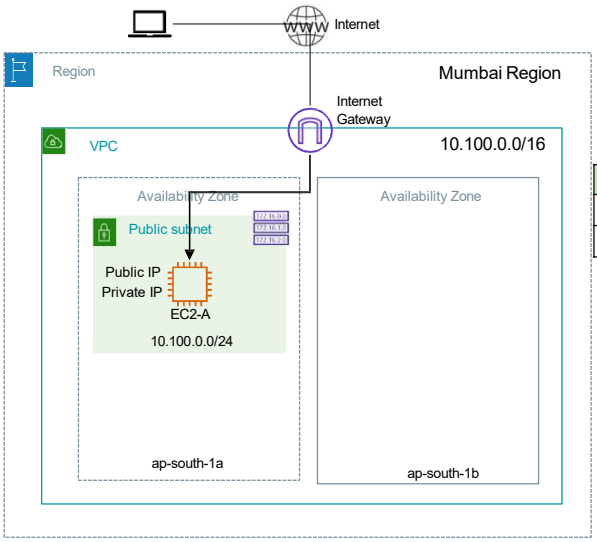


AWS Network Firewall

- Protect your entire Amazon VPC
- From Layer 3 to Layer 7 protection
- Any direction, you can inspect
  - VPC to VPC traffic
  - Outbound to internet
  - Inbound from internet
  - To / from Direct Connect & Site-to-Site VPN
- Internally, the AWS Network Firewall uses the AWS Gateway Load Balancer
- Rules can be centrally managed cross-account by AWS Firewall Manager to apply to many VPCs



Exercise – Public Subnet



Public Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	igw-xxxxxx

High level steps

1. Create a new VPC
2. Create an Internet Gateway & associate with your VPC
3. Create a Subnet in one of the availability zone. Enable Auto-assign Public IP for a subnet.
4. Create a Route table and add a route for destination (0.0.0.0/0) with target as an internet gateway
5. Associate route table with your subnet
6. Launch EC2 instance in your subnet.
7. Connect to EC2 instance over SSH using it's Public IP

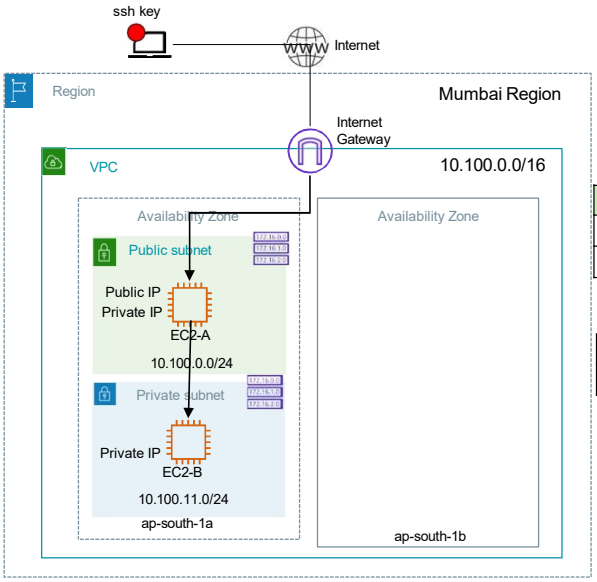
Steps

1. Create VPC
   a. AWS Console -> Go to VPC service -> Your VPCs -> Create VPC (Resources to create : VPC Only, Name tag: VPC-A, IPv4 CIDR block: Select IPv4 CIDR manual input (10.100.0.0/16) , Tenancy : Default - > Create VPC
2. Create Internet Gateway
   a. Internet Gateways -> Create internet gateway (Name tag: VPC-A-IGW) -> Create internet gateway
   b. Select Internet gateway -> Actions -> Attach to VPC -> Select your VPC (VPC-A) -> Attach Internet Gateway
3. Create Subnet
   a. Subnets -> Create subnet
   b. Select VPC ID: VPC-A
   c. Subnet 1 of 1 -> Subnet Name: VPC-A-Public, AZ: Select AZ 1, IPv4 CIDR block : 10.100.0.0/24) -> Create Subnet
   d. Select Subnet -> Actions -> Edit Subnet Settings - > Modify Auto-Assign IP Settings-> Enable -> Save
1. Create Route table
   a. Route Tables -> Create Route Table (Name: VPC-A-Public-RT, select VPC: VPC-A) -> Create route table
   b. Select Route table -> Routes -> Edit routes -> Add another route (Destination: 0.0.0.0/0, Target: Internet gateway -> igw-xxxxx) -> Save changes
5. Associate route table with the subnet
   a. Select Route table -> Subnet Associations -> Edit subnet associations -> Check the VPC-A-Public subnet -> Save associations

Steps

6. Launch EC2 instance in newly created Public Subnet
   a. Go to EC2 Service -> EC2 Dashboard -> Launch Instances
   b. Name: EC2-A
   c. Select Application and OS Images (Amazon Machine Image): Amazon Linux (default)
   d. Select instance type: t2.micro (default)
   e. Select key pair : *Your key-pair that you had created earlier in pre-requisites*
   f. Network settings -> Edit -> Select your VPC (VPC-A) and your public subnet
   g. Make sure Auto-Assign Public IP is enabled
   h. Firewall -> Create security group
      a. Name: EC2-A-SG
      b. Inbound Security group rule: Add rule (Type-> SSH, port Range-> 22, source type -> My IP
   i. Configure Storage -> 8GiB, gp3 (default)
   j. Launch Instance
7. Connect to EC2 instance with *Public IP* from your workstation using Putty or terminal with user *ec2-user*

Exercise – Private Subnet



Public Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	igw-xxxxxx

Private Subnet Route table

Destination	Target
10.100.0.0/16	Local

Continuing with earlier setup

1. Create a new subnet in Availability zone 1 (as shown)
2. Create a new route tables and associate with Private subnet. (route entries as shown)
3. Launch EC2 instance (EC2-B) in the Private subnet. Make sure Security Group for EC2-B allows SSH and ICMP (ping) from VPC CIDR.
4. Connect to EC2-A over SSH. Ping to EC2-B Private IP.
5. Bring/copy SSH key onto EC2-A and change file permissions to 400
6. From EC2-A terminal SSH into EC2-B using ssh key file
7. Once logged into EC2-B, try to ping google.com or wget google.com

Steps

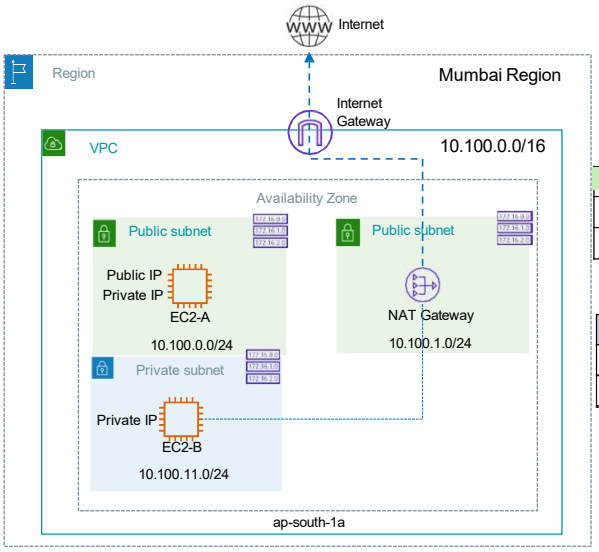
1. Create Private subnet
   a. Subnets -> Create subnet, Select VPC ID: VPC-A
   b. Subnet 1 of 1 -> Subnet Name: VPC-A-Private, AZ: Select AZ 1, IPv4 CIDR: 10.100.11.0/24) -> Create Subnet
2. Create a Route table for Private subnet
   a. Route Tables -> Create Route Table (Name: VPC-A-Private-RT, select VPC: VPC-A) -> Create route table
   b. Select Route table -> Subnet Associations -> Edit subnet associations -> Check the VPC-A-Private subnet -> Save associations
3. Launch EC2-B instance in the Private Subnet
   a. Go to EC2 Service -> EC2 Dashboard -> Launch Instances
   b. Name: EC2-B
   c. Select AMI: Amazon Linux (default)
   d. Select instance type: t2.micro (default)
   e. Select key pair : *Your key-pair that you had created earlier*
   f. Network settings -> Edit -> Select your VPC (VPC-A) and Private subnet (VPC-B-Private)
   g. Firewall -> Create security group
      a. Security Group Name: EC2-B-SG
      b. Inbound Security group rule: Add rule for SSH (port 22) for source type (Custom) Source as 10.100.0.0/16
      c. Add security group rule
      d. Inbound Security group rule: Add rule for ICMP IPv4 for source as 10.100.0.0/16 (Type: All ICMP –IPV4, Source type- Custom as 10.100.0.0/16
   h. Configure Storage -> 8GiB, gp3 (default)
   i. Launch Instance and wait for the instance to be in running state

Steps

4. From EC2-A instance, ping to EC2-B private IP using command:
   $ping 10.100.11.x
5. Create a key.pem on EC2-A using any editor. Paste .pem file content and save the file. Change file permissions to 400 using command:
   $chmod 400 key.pem
6. SSH to EC2-B using command:
   $ssh –i key.pem ec2-user@10.100.11.x
7. Try to access the internet using following commands:
   $ping google.com
   $wget https://google.com

   [Above commands should not work, why? There is no outbound internet connectivity to the Private subnet]

Exercise – NAT Gateway



Public Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	igw-xxxxxx

Private Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	Nat-gateway

Continuing from the earlier setup

1. Create a new Public subnet in Availability zone 1 (as shown)
2. Associate an existing Public route table with this subnet
3. Create a NAT Gateway in this Public subnet
4. Update the Private subnet route table and add route entry for destination 0.0.0.0/0 with target as nat gateway
5. While logged into EC2-B, try to access internet

Steps

1. Create a new Public subnet
    a. Subnets -> Create subnet
    b. Select VPC ID: VPC-A
    c. Subnet 1 of 1 -> Subnet Name: VPC-A-Public-NAT, AZ: Select AZ 1, IPv4 CIDR: 10.100.1.0/24)
    d. Save
2. Associate existing Public route table
    a. Select Public Route table (VPC-A-Public-RT) -> Subnet Associations -> Edit subnet associations -> Check the VPC-A-Public-NAT subnet -> Save associations
3. Create a nat gateway
    a. VPC console -> NAT Gateways -> Create NAT Gateway
    b. Name: VPC-A-NATGW
    c. Subnet: VPC-A-Public-NAT
    d. Connectivity Type: Public
    e. Elastic IP -> Allocate Elastic IP
    f. Create NAT gateway
    g. Wait until NAT gateway is available/ready.
4. Update Private subnet route table
    1. Route tables -> Select Private subnet (VPC-A-Private-RT) -> Routes
    2. Edit routes -> Add another route (Destination: 0.0.0.0/0, Target: nat gateway -> nat-xxxxx) -> Save changes

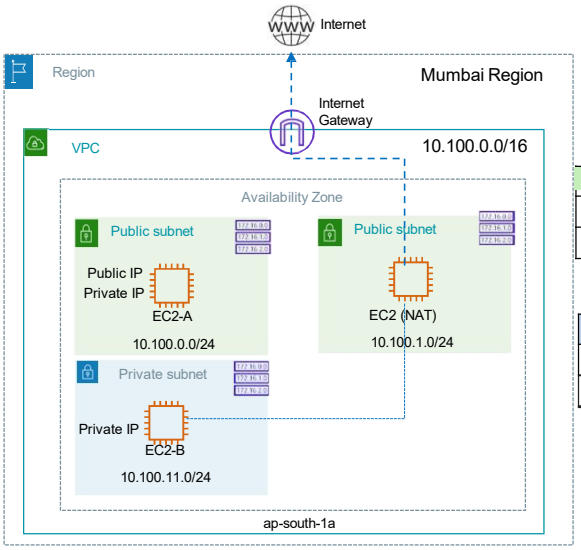
Steps

5. Log into EC2-B (SSH) and try to access internet using following commands:
    $ping google.com
    $wget https://google.com

[Above commands should work, why? There is now outbound internet connectivity via the NAT Gateway]

Note: If you don't plan to continue with the next exercises at this moment, then terminate ec2 instance(s), delete NAT gateway, release Elastic IP and optionally delete the VPC. Otherwise you can continue with this setup until you finish first 4 exercises and then terminate/delete everything.

Exercise – NAT Instance



Public Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	igw-xxxxxx

Private Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	**eni-xxxxxx**

Continuing from the earlier setup

1. Delete NAT gateway and release Elastic IP & remove route entry from Private route table
2. Launch EC2 instance in the NAT Public subnet using nat AMI (amzn-ami-vpc-nat-xxxx). Make sure EC2 instance gets Public IP and Security group to allow port 80 & 443 inbound traffic for VPC CIDR source.
3. Disable source/destination check for NAT EC2 instance
4. Update the Private subnet route table and modify route entry for destination 0.0.0.0/0 with target as NAT EC2 ENI
5. While logged into EC2-B, try to access internet

Steps

1. Delete NAT Gateway and release EIP
   a. NAT gateways -> Select your NAT gateway -> Actions -> Delete NAT Gateway
   b. Wait for NAT gateway to be deleted
   c. Elastic Ips -> Select your Elastic IP -> Actions -> Release Elastic IP addresses
2. Update private subnet route table
   a. Go to EC2 Service -> EC2 Dashboard -> Launch Instances
   b. Name: EC2-NAT
   c. Select AMI: amzn-ami-vpc-nat-xxxxx
   d. Select instance type: t2.micro (default)
   e. Select key pair : *Your key-pair that you had created earlier*
   f. Network settings -> Edit -> Select your VPC (VPC-A) and Public subnet (Public-Subnet-NAT)
   g. Make sure Auto-Assign Public IP is enabled
   h. Firewall -> Create security group
      a. Name: EC2-NAT-SG
      b. Inbound Security group rule: Add rule for HTTP (80) for source as 10.100.0.0/16
      c. Inbound Security group rule: Add rule for HTTPS (443) for source as 10.100.0.0/16
      d. Inbound Security group rule: Add rule for ICMP IPv4 for source as 10.100.0.0/16
   i. Configure Storage -> 8GiB, gp3 (default)
   j. Launch Instance and wait for the instance to be in running state

Steps

3. Disable source/destination check for EC2 NAT instance
   a. Select EC2-NAT instance -> Actions -> Networking -> Change source/destination check
   b. enable Stop -> Save
4. Update Private subnet route table
   a. Route tables -> Select Private subnet (VPC-A-Private-RT) -> Routes
   b. If you are continuing from the previous exercise, then you should see a Blackhole route because you have deleted NAT Gateway but did not remove this route. In that case, Edit routes and remove this rule -> Save.
   c. Edit routes -> Add new route (Destination: 0.0.0.0/0, Target: EC2 NAT instance -> Save. This should add EC2 instance ENI as target for this route (eni-xxxxxx)
5. Log into EC2-B (SSH) and try to access internet using following commands:

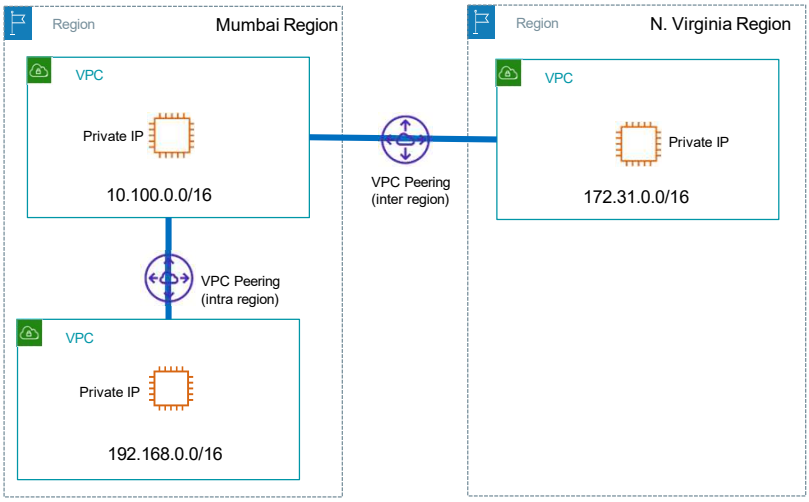
   $ping google.com
   $wget https://google.com

   [Above commands should work, why? There is now outbound internet connectivity via the NAT EC2 instance]

Clean-up

a. If you are **not** continuing with next exercises
   a. Terminate all three EC2 instances
   b. Delete VPC-A (there is no adding cost even if you have VPC-A and subnets in your account)
b. If you want to continue with the next exercise
   a. Terminate EC2-NAT instance
   b. Update VPC-A-Private-RT and remove the route you added for 0.0.0.0/0 via the EC2 instance.
   c. Continue with next exercise

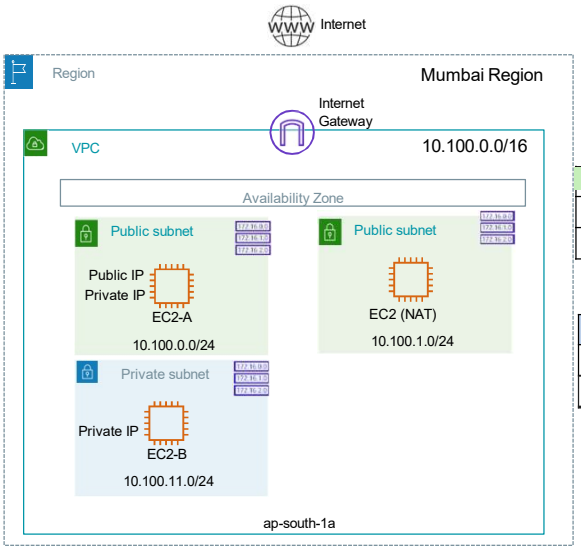
Exercise – VPC Peering



Exercise – VPC Peering

If you are continuing from the earlier setup

1. Delete EC2 NAT instance and release Elastic IP & remove route entry from Private route table



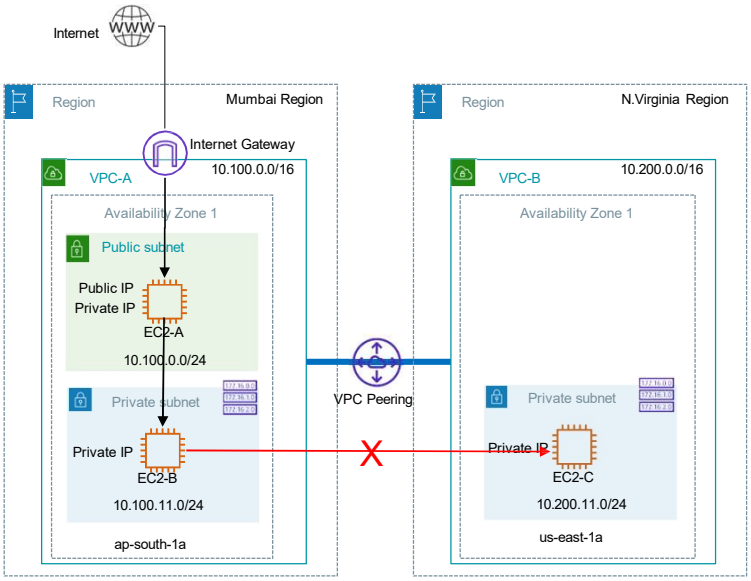
Public Subnet Route table

Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	igw-xxxxxx

Private Subnet Route table

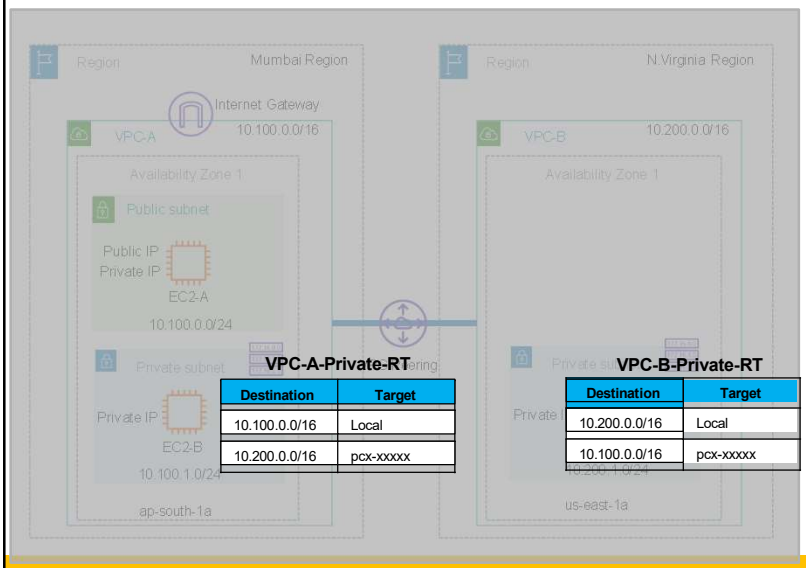
Destination	Target
10.100.0.0/16	Local
0.0.0.0/0	**eni-xxxxxx**

Exercise – VPC Peering



1. Create 2 VPCs in different AWS regions as shown. Create Internet gateway and associate with VPC-A. VPC-B do not need internet gateway.
2. Create Public and Private subnets as shown in respective VPCs
3. Launch EC2 instances in respective subnets. Only EC2-A should have Public IP. Have appropriate security groups to allow SSH from EC2-A to EC2-B and SSH and ping from EC2-B to EC2-C.
4. Login to EC2-A -> EC2-B and try to ping EC2-C. At this moment from EC2-B you should not be able to reach to EC2-C.
5. Create VPC peering between both the VPCs.

Exercise – VPC Peering



VPC-A-Private-RT

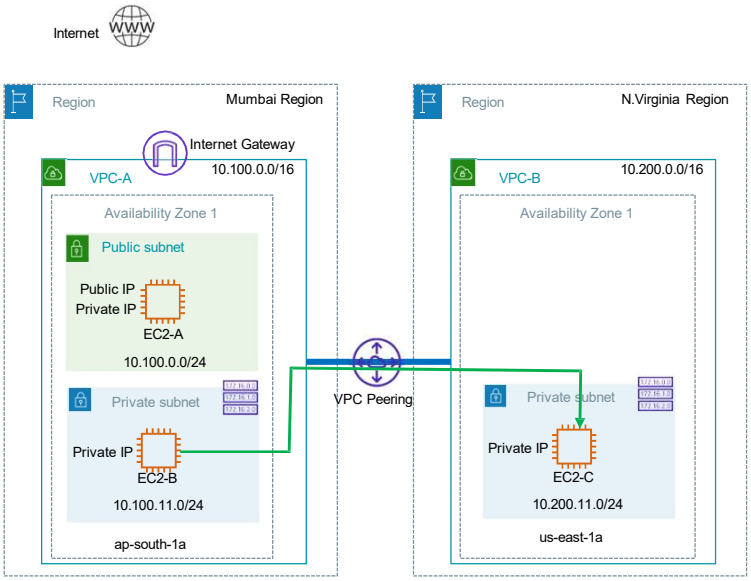
Destination	Target
10.100.0.0/16	Local
10.200.0.0/16	pcx-xxxxx

VPC-B-Private-RT

Destination	Target
10.200.0.0/16	Local
10.100.0.0/16	pcx-xxxxx

1. Create 2 VPCs in different AWS regions as shown. Create Internet gateway and associate with VPC-A. VPC-B do not need internet gateway.
2. Create Public and Private subnets as shown in respective VPCs
3. Launch EC2 instances in respective subnets. Only EC2-A should have Public IP. Have appropriate security groups to allow SSH from EC2-A to EC2-B and SSH and ping from EC2-B to EC2-C.
4. Login to EC2-A -> EC2-B and try to ping EC2-C. At this moment from EC2-B you should not be able to reach to EC2-C.
5. Create VPC peering between both the VPCs.
6. Modify both VPC Private subnet route tables and add routes for the destination VPC with VPC peering as target

Exercise – VPC Peering



1. Create 2 VPCs in different AWS regions as shown. Create Internet gateway and associate with VPC-A. VPC-B do not need internet gateway.
2. Create Public and Private subnets as shown in respective VPCs
3. Launch EC2 instances in respective subnets. Only EC2-A should have Public IP. Have appropriate security groups to allow SSH from EC2-A to EC2-B and SSH and ping from EC2-B to EC2-C.
4. Login to EC2-A -> EC2-B and try to ping EC2-C. At this moment from EC2-B you should not be able to reach to EC2-C.
5. Create VPC peering between both the VPCs.
6. Modify both VPC Private subnet route tables and add routes for the destination VPC with VPC peering as target
7. Communication should be successful in both direction between EC2-B and EC2-C

Steps

1. Create VPC-A and VPC-B in different AWS regions with non-overlapping CIDRs.
   a. Create VPC-A in Mumbai (ap-south-1) region with CIDR 10.100.0.0/16. Create and associate Internet gateway to VPC-A.
   b. Create VPC-B in N.Virginia (us-east-1) region with CIDR 10.200.0.0/16. No internet gateway.
2. Create subnets
   a. In VPC-A, create 1 Public subnet and 1 Private subnet. Create route tables and associate with corresponding subnets.
   b. In VPC-B, create 1 Private subnet. Create a route table and associate with the subnet.
3. Launch EC2 instances and login
   a. Launch Public EC2 instance in VPC-A Public Subnet (with Public IP) and associate security group to allow SSH (22) from MyIP or anywhere.
   b. Launch Private EC2 instance in VPC-A Private Subnet. Associate security group to allow SSH (22) from VPC-A CIDR (10.100.0.0/16)
   c. Launch Private EC2 instance in VPC-B Private Subnet. Assoicate security group to allow SSH (22) and ping (ICMP-IPv4) from VPC-A CIDR (10.100.0.0/16)
4. Connect to EC2-B and try to reach EC2-C
   a. SSH into EC2-A from your workstation -> From EC2-A, SSH into EC2-B (for this you need to bring your SSH key to EC2-A. Refer pre-requisites or troubleshooting section on how to do it)
   b. From EC2-B terminal, try to ping EC2-C private IP.

[This should not work, why? There is no connectivity between VPC-A and VPC-B.

Steps

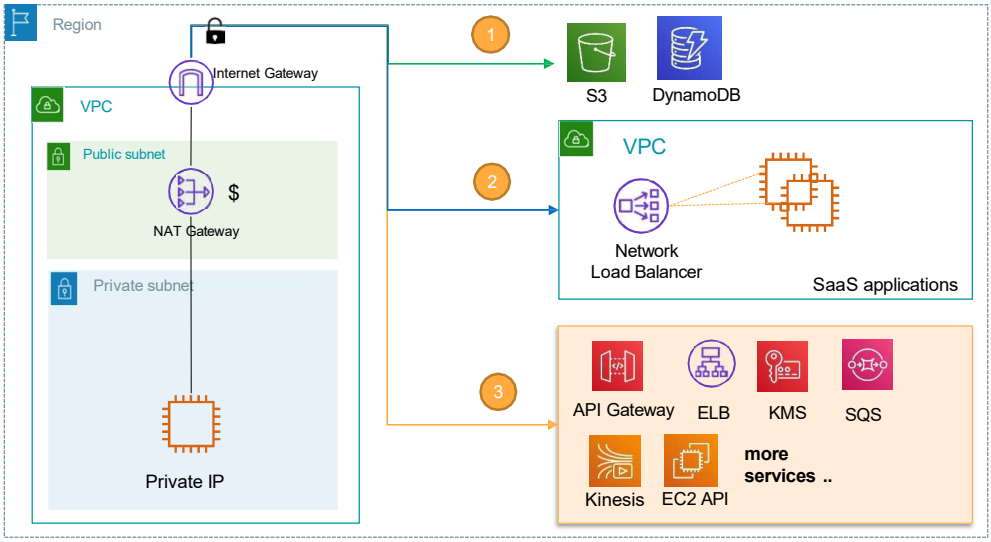
5. Create VPC Peering between both the VPCs
   a. Go to N.Virginia region VPC console -> Copy the VPC ID VPC-B (vpc-xxxxx)
   b. Go to Mumbai region VPC console -> Peering Connections -> Create peering connection
   c. Name: VPC-A-VPC-B-Peering
   d. Select local VPC: VPC-A
   e. Select another VPC to peer with: My Account
   f. Region: Another Region
   g. Select region: N. Virginia (us-east-1)
   h. VPC ID: *paste the VPC-B ID that you have copied.*
   i. Create peering connection
   j. Go to N.Virginia region VPC console -> Peering Connections -> Actions -> Accept request -> Accept
   k. From EC2-B terminal, try to ping EC2-C private IP again, does that work?
6. Update both Private subnet route tables and add route to reach to other VPC
   a. Mumbai region -> Select Private subnet (VPC-A-Private-RT) -> Routes -> Edit routes -> Add a route (Destination: 10.200.0.0/16, Target: Peering connection) -> Save changes
   b. N.Virginia region -> Select Private subnet (VPC-B-Private-RT) -> Routes -> Edit routes -> Add a route (Destination: 10.100.0.0/16, Target: Peering connection) -> Save changes
5. Log into EC2-B (SSH) and try to access EC2-C private IP using following commands:
   $ping 10.200.11.x

   [Above commands should work, why? There is now connectivity and route between both VPCs Private subents.

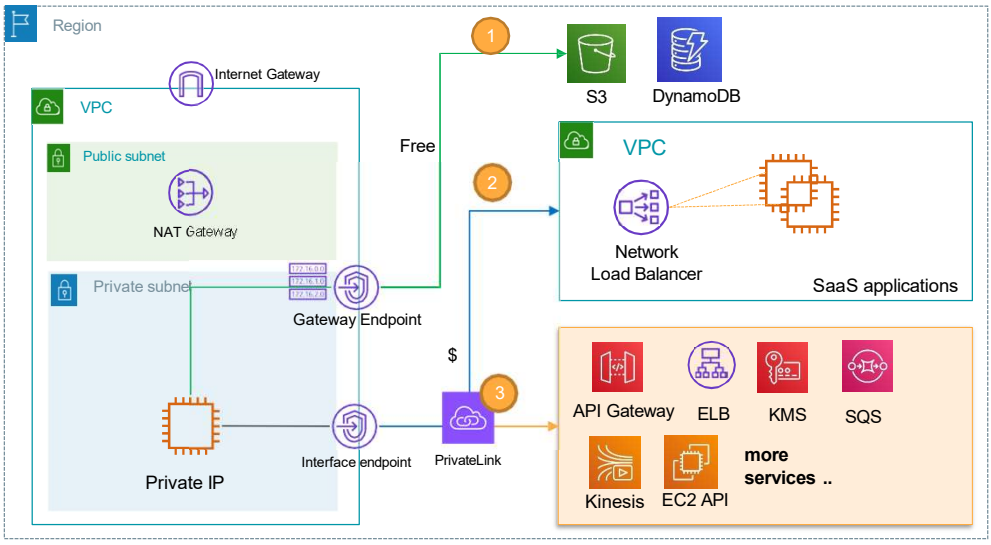
Clean-up

a. If you are **not** continuing with next exercises
   a. Terminate all three EC2 instances
   b. Delete VPC peering connection
   c. Delete VPC-B in N.Virginia region
   d. Delete VPC-A in Mumbai region (there is no adding cost even if you have VPC-A and subnets in your account)
b. If you want to continue with the next exercise
   a. Terminate EC2-C in VPC-B
   b. Delete VPC peering connection
   c. Delete VPC-B in N.Virginia region
   d. Continue with next exercise

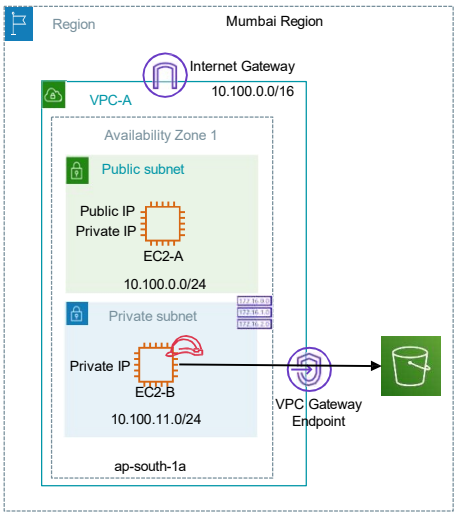
VPC Endpoints & PrivateLink



VPC Endpoints & PrivateLink



Exercise – VPC Gateway endpoint for S3



Private Subnet Route table

Destination	Target
10.100.0.0/16	Local
pl-xxxxxx	vpce-xxxxxx

1. Create a VPC, internet gateway, a public subnet and a private subnet as shown
2. Launch EC2 instances in respective subnets. Only EC2-A should have Public IP. Have appropriate security groups to allow SSH from EC2-A to EC2-B.
3. Create a S3 bucket in the same region, upload any sample file
4. Create an IAM role for EC2 instance and attach S3 read-only IAM policy. Associate role with EC2-B instance.
5. Login to EC2-B and try to download file from S3 using s3 CLI command. Does not work.
6. Create VPC endpoint for S3 and update Private subnet route table.
7. Try to download the same file from S3 again. Should work this time.

Steps

1. Create VPC and Subnets in any of the region
   a. Create VPC-A in Mumbai (ap-south-1) region with CIDR 10.100.0.0/16. Create and associate Internet gateway to VPC-A.
   b. Create 1 Public subnet and 1 Private subnet. Create route tables and associate with corresponding subnets.
2. Launch EC2 instances and login
   a. Launch Public EC2 instance in VPC-A Public Subnet (with Public IP) and associate security group to allow SSH (22) from MyIP or anywhere.
   b. Launch Private EC2 instance in VPC-A Private Subnet. Associate security group to allow SSH (22) from VPC-A CIDR (10.100.0.0/16)
3. Create S3 bucket and upload sample file
   a. S3 console -> Create Bucket
   b. Bucket Name: <unique bucket name>
   c. AWS Region: same region in which you are doing this exercise
   d. Create bucket
   e. Select same bucket -> Upload -> Add files -> Choose sample file from your local machine -> Upload
4. Create IAM role for EC2 to be able to download file from S3
   a. Go to IAM console -> Roles -> Create role
   b. Use case: EC2
   c. Click Next -> Permission policies -> Search for S3 and select "AmazonS3ReadOnlyAccess“ policy -> Next
   d. Role name: EC2_ROLE_FOR_S3_READONLY -> Create role

Steps

5. From EC2-B, try to download your file from S3
   a. SSH into EC2-A from your workstation
   b. From EC2-A, SSH into EC2-B (for this you need to bring your SSH key to EC2-A. Refer pre-requisites or troubleshooting section on how to do it)
   c. From EC2-B terminal, try to download file from S3. This command does not work as there is no connectivity to S3.
      $ aws s3 cp s3://bucket-name/filename /home/ec2-user/
6. Create VPC gateway endpoint for S3 and update route table
   a. VPC console -> Endpoints -> Create endpoint
   b. Name: my-s3-endpoint
   c. Services: search S3 and select “com.amazonaws.ap-south-1.s3” Type: Gateway
   d. VPC: VPC-A
   e. Route tables: Select Private subnet route table
   f. Create endpoint
   g. After endpoint is created successfully, route table should be updated with route for s3 prefix list with target as VPC endpoint.
7. Try to download your file from S3 again
   a. From EC2-B terminal, try to download file from S3.
      $ aws s3 cp s3://bucket-name/filename /home/ec2-user/

[Above commands should work, why? There is now connectivity and route between VPC and S3 through VPC endpoint.

Clean-up

a. If you are **not** continuing with next exercises
   a. Terminate both EC2 instances
   b. Delete VPC endpoint
   c. Delete VPC-A (there is no adding cost even if you have VPC-A and subnets in your account)
b. If you want to continue with the next exercise
   a. Delete VPC endpoint connection
   b. Update Private subnet route table and remove route for s3.