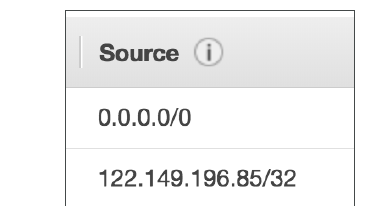
**CIDR (Classless Inter Domain Routing)**

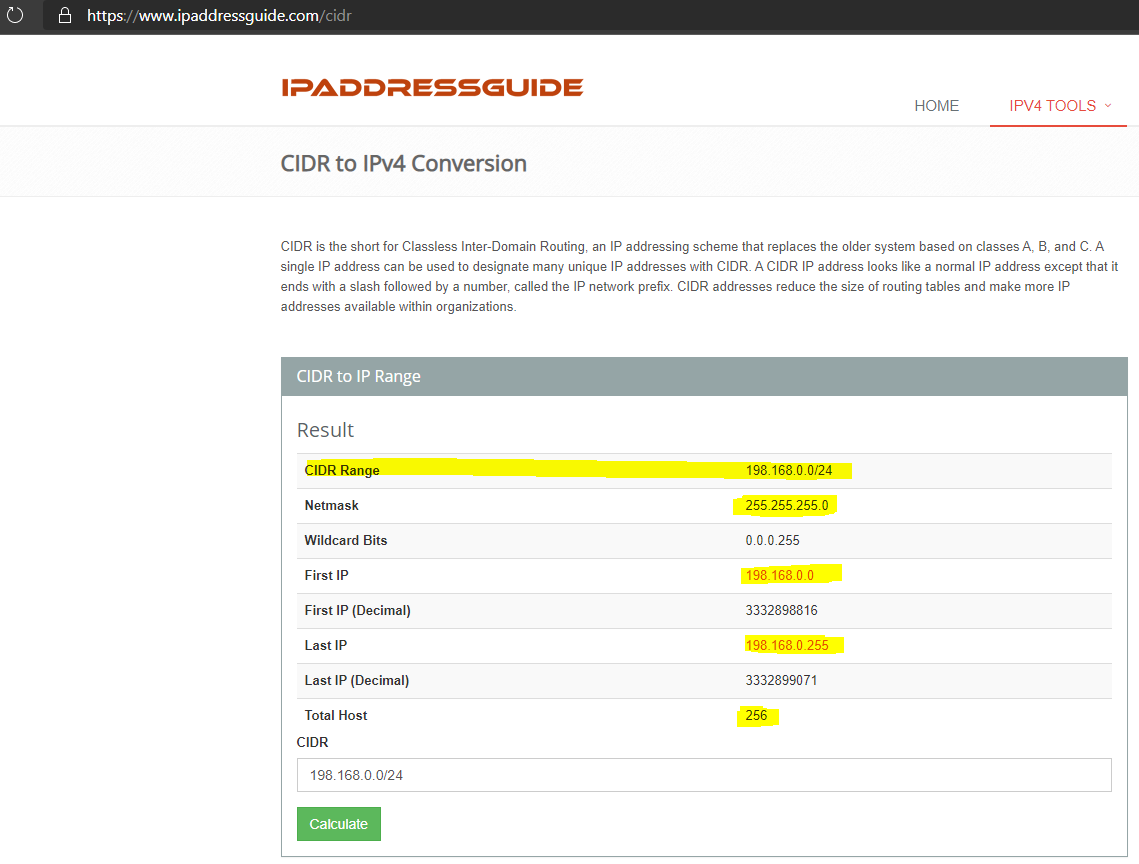
* Are used for security group rules or AWS network



* CIDR – defines IP address range
* A CIDR has two components:
  + The base IP (XX.XX.XX.XX)
    - The base IP represents an IP contained in the range
  + The Subnet Mask (/26)
    - The subnet masks define how many bits can change in IP.
    - The subnet mask can take two forms. Examples:
      * 255.255.255.0 (Subnet mask)-> less common (windows)
      * /24 (CIDR) ->more common (cloud)
    - The subnet masks basically allows part of the underlying IP to get additional next values from the base IP
      * /32 allows for 1 IP = 2^0 , but no IP can change
      * /31 allows for 2 IP = 2^1
      * /30 allows for 4 IP = 2^2
      * /16 allows for 65,536 IP = 2^16
      * /0 allows for all IPs = 2^32
      * Quick memo:
        + /32 – no IP number can change
        + /24 - last IP number can change
        + /16 – last IP two numbers can change
        + /8 – last IP three numbers can change
        + /0 – all IP numbers can change

**Little exercise**

* 192.168.0.0/24 = … ?
  + 192.168.0.0 – 192.168.0.255 (2^8=256 IP)
* 192.168.0.0/16 = … ?
  + 192.168.0.0 – 192.168.255.255 (65,536 IP)
* 134.56.78.123/32 = … ?
  + Just 134.56.78.123
* 0.0.0.0/0
  + All IP!
* When in doubt, use this website: <https://www.ipaddressguide.com/cidr>



**Private vs Public IP (IPv4) - Allowed ranges**

* The Internet Assigned Numbers Authority (IANA) established certain blocks of IPV4 addresses for the use of private (LAN) and public
* (Internet) addresses.
  + Private IP can only allow certain values
    - 10.0.0.0 – 10.255.255.255 (10.0.0.0/8) <= in big networks
    - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12) <= default AWS one
    - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16) <= example: home networks
  + All the rest of the IP on the internet are public IP

**DEFAULT VPC (virtual Private Cloud)**

* All new accounts have a default VPC
* New instances are launched into this default VPC ,if no subnet is specified
* Default VPC have internet connectivity and all instances have public IP
* We also get a public and a private DNS name

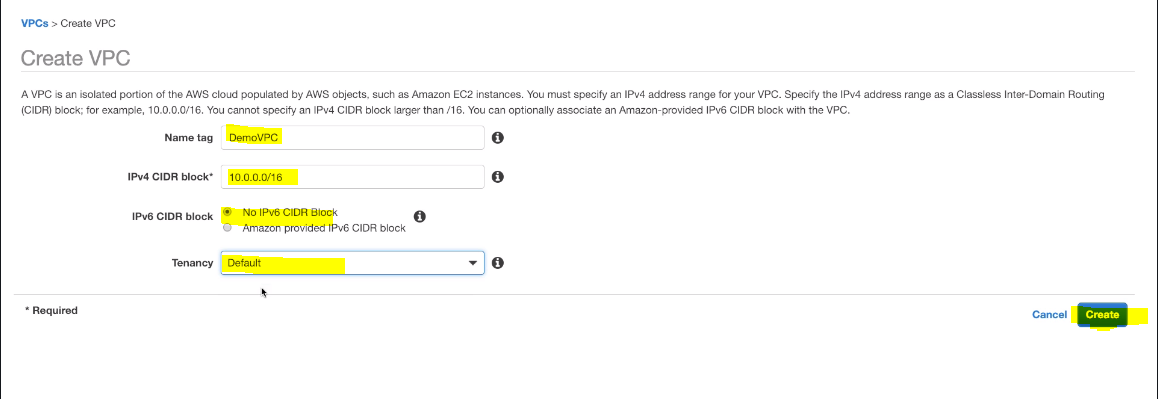


**Create VPC in AWS Hands on**

* You can have multiple VPCs in a region (max 5 per region – soft limit)
* Max CIDR per VPC is 5. For each CIDR:
  + Min size is /28 = 16 IP Addresses
  + Max size is /16 = 65536 IP Addresses
* Because VPC is private, only the Private IP ranges are allowed



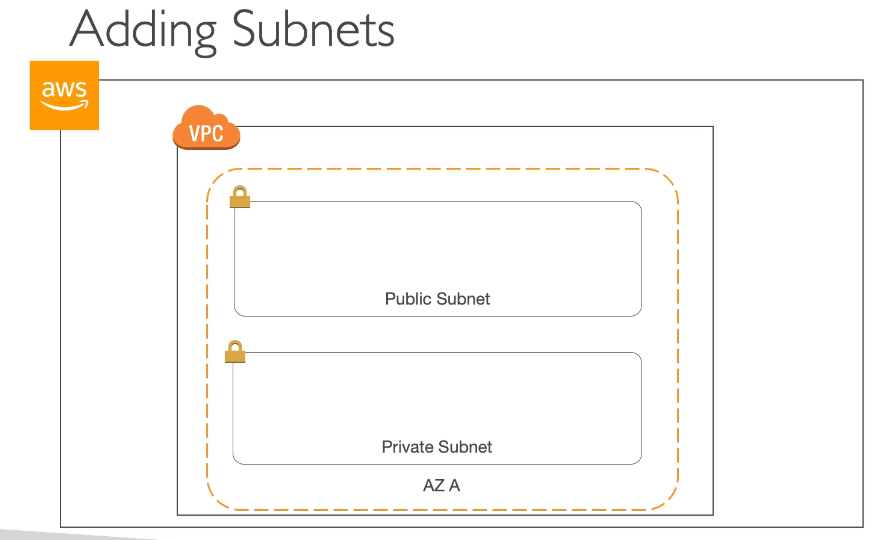
1. Go to Create VPC



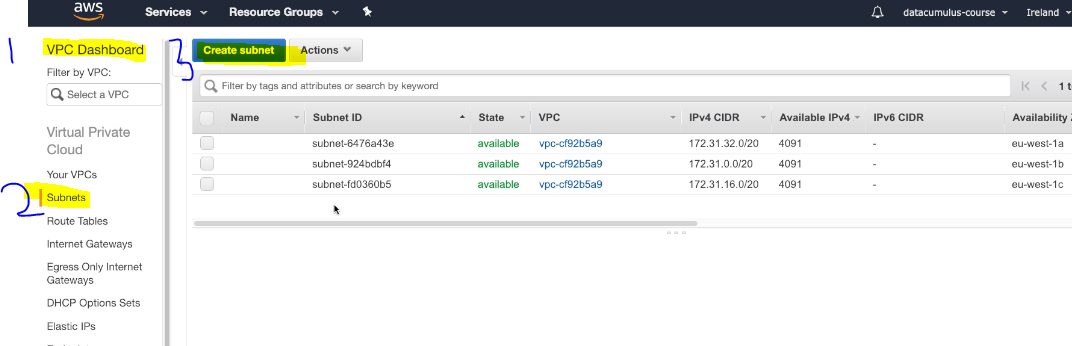
1. Above step only will create your VPC and same can be verified

**ADD SUBNET TO YOUR VPC**

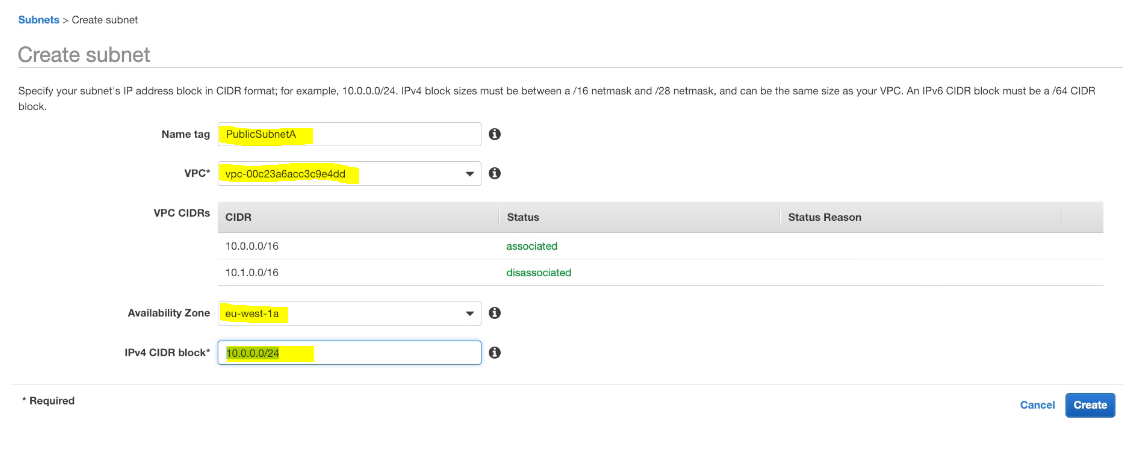
Let’s Create 2 subnets (public and private) per AZ ( in 2 AZ’s) = 4 Subnets



1. Go to create subnet under subnets

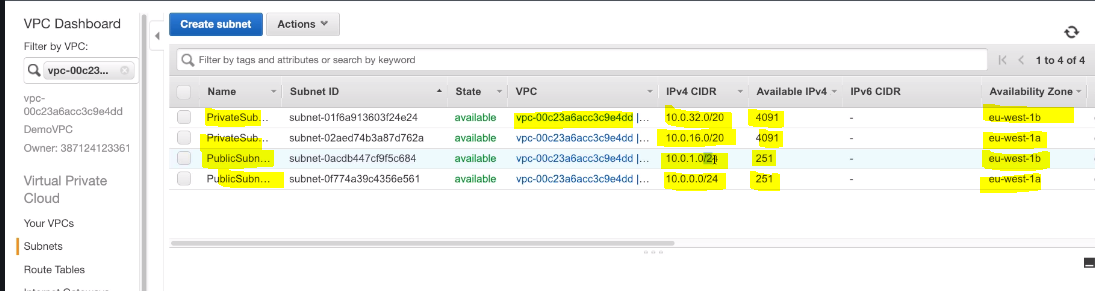


1. Choose VPC under which subnets needs to be created , Availability zone and CIDR



*\*\*Public subnet in AZ –>* ***eu-west-1a***

1. Similarly, create private subnet in AZ **-> eu-west-1a** and private/public in AZ -> **eu-west-1b**. Create Private IP with more CIDR and change Base IP of CIDR in each subnet. Hence, we would be having 4 subnets.



**Note:**

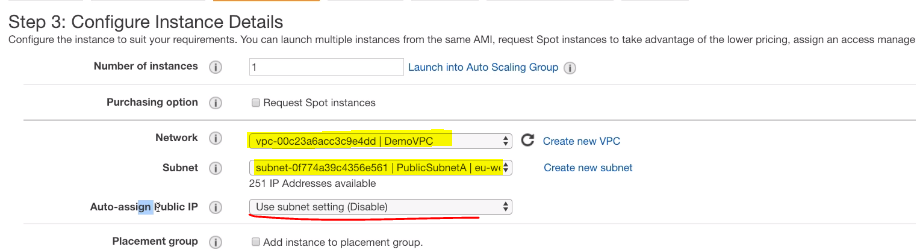
* We had specified CIDR 10.0.0.0/24 , so we should have got 2^32-24=2^8=256, but we have got 251 IP’s. Similarly, in private subnet we have got 4091 IP’s instead of 2^12=4096. Hence, in both case 5 IP’s are missing.
* Reason:
  + AWS reserves 5 IPs address (first 4 and last 1 IP address) in each Subnet
  + These 5 IPs are not available for use and cannot be assigned to an instance
  + Ex, if CIDR block 10.0.0.0/24, reserved IP 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 the address is reserved
  + Exam Tip:
    - If you need 29 IP addresses for EC2 instances, you can’t choose a Subnet of size /27 (32 IP) .You need at least 64 IP, Subnet size /26 (64-5 = 59 > 29, but 32-5 = 27 < 29)

**INTERNET GATEWAY AND ROUTE TABLE**

**What is Internet Gateway??**

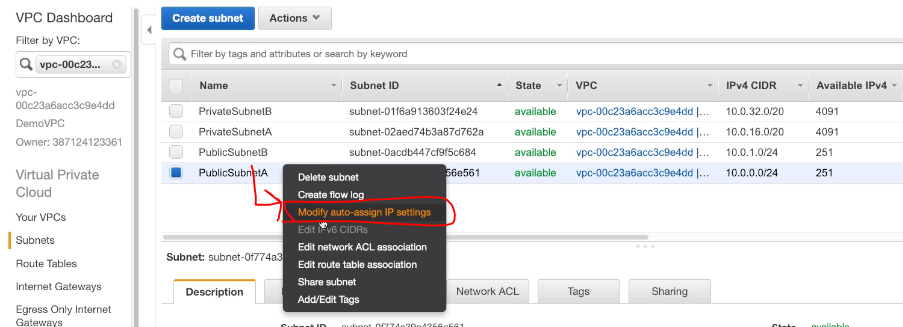
Let’s understand it with example:

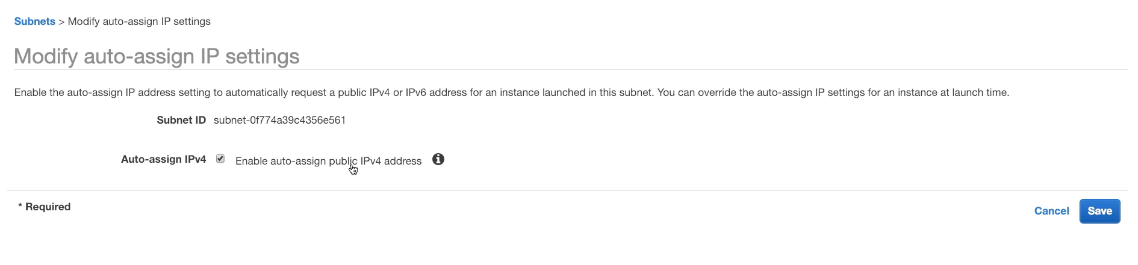
1. Create EC2 in Demo VPC we have created with Public Subnet.



\*\* Create EC2 instance as usual, but select your own VPC and subnet

1. It will show auto-assign IP is disables. Since we have not enabled auto-assign IP while creating subnets. So enable it by going into subnets

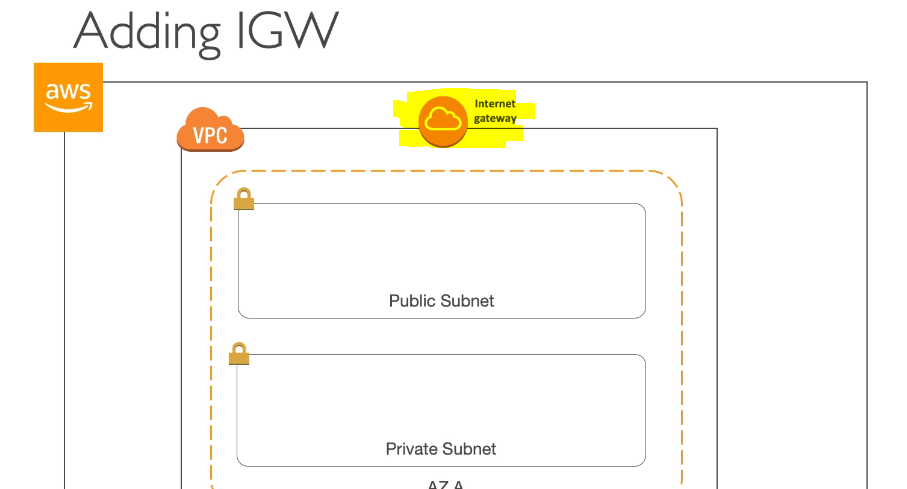




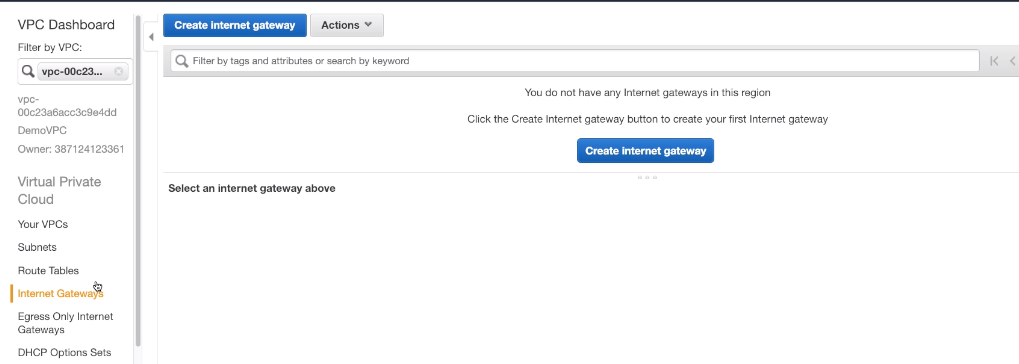
1. Now EC2 instance is created and while trying to access EC2 , we will get TIMEOUT error . Since, EC2 doesn’t have internet connection.

* Hence, **Internet Gateway (IGW)** allows resources of VPC to connect to internet.
* IGW scales horizontally and HA like other AWS services
* One VPC is connected to only one IGW and vice versa
* Internet Gateway at their own doesn’t allow internet access. Route tables must be also used

**Add Internet Gateway**



1. Create Internet gateway in VPC

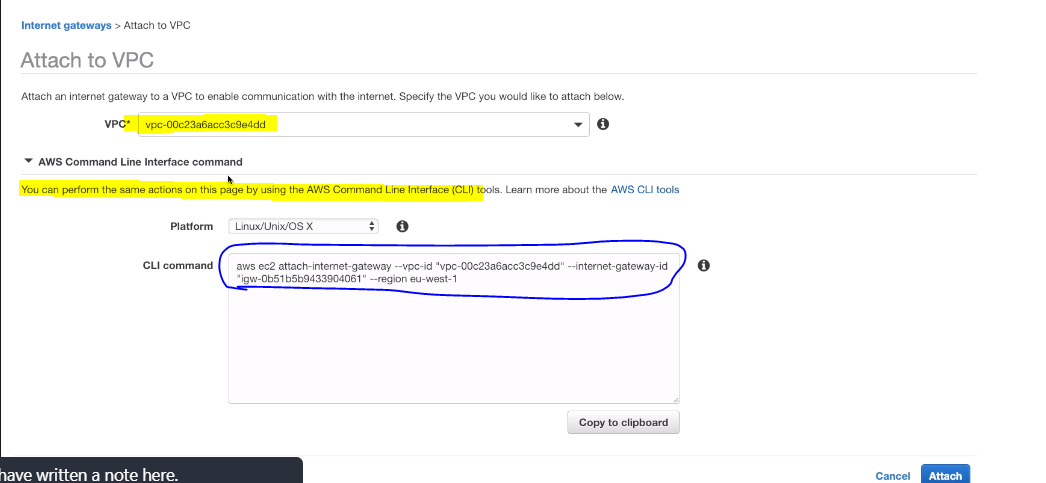




1. Attach/De-attach Internet Gateway to VPC

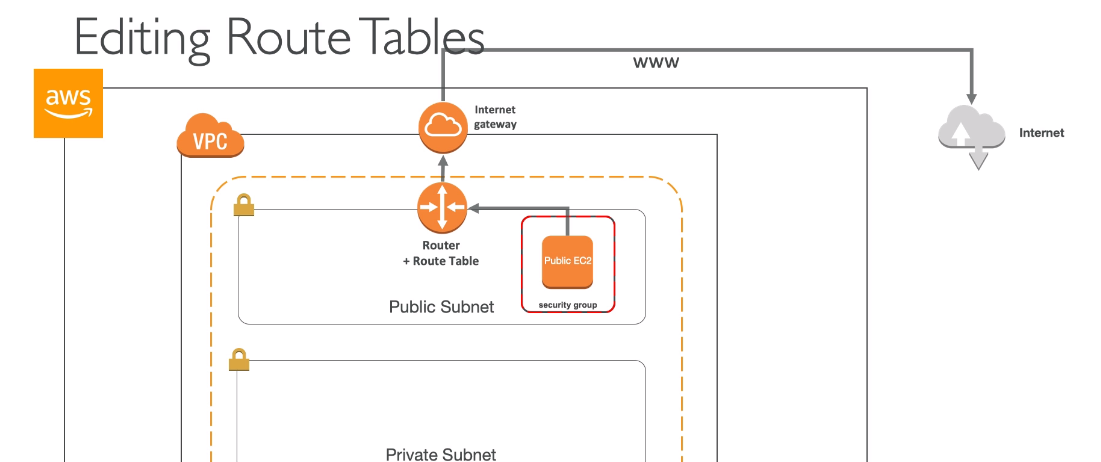


*\*\* One is default IGW attached to default VPC*

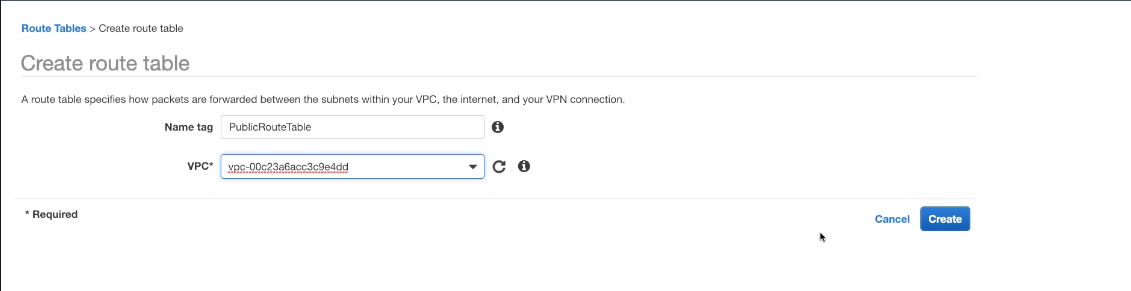


1. Still, you won’t be able to access the EC2 instance . Since , IGW connects to internet but Route table needs to be edited

**Add Route table**

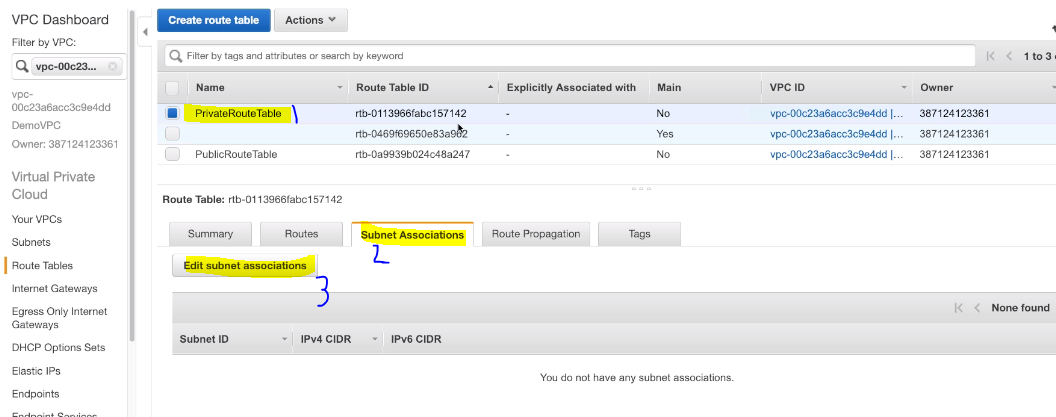


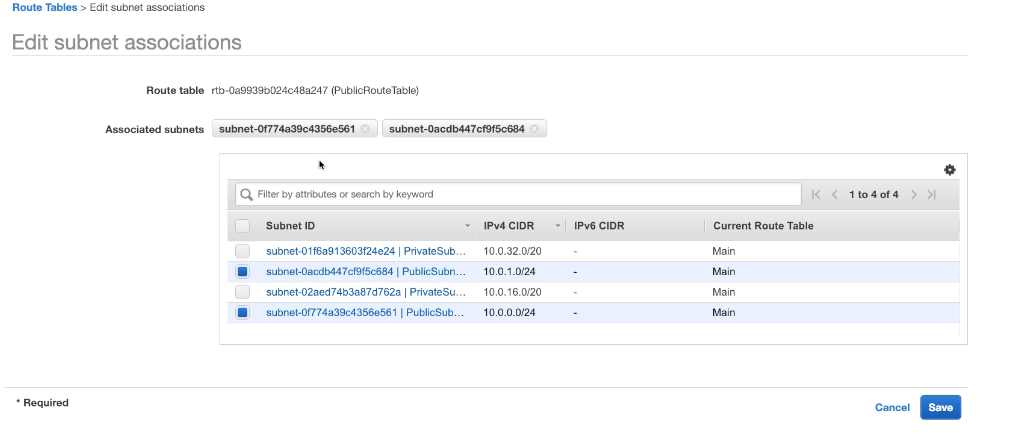
1. Got to Route Tables in VPC and click on create route table



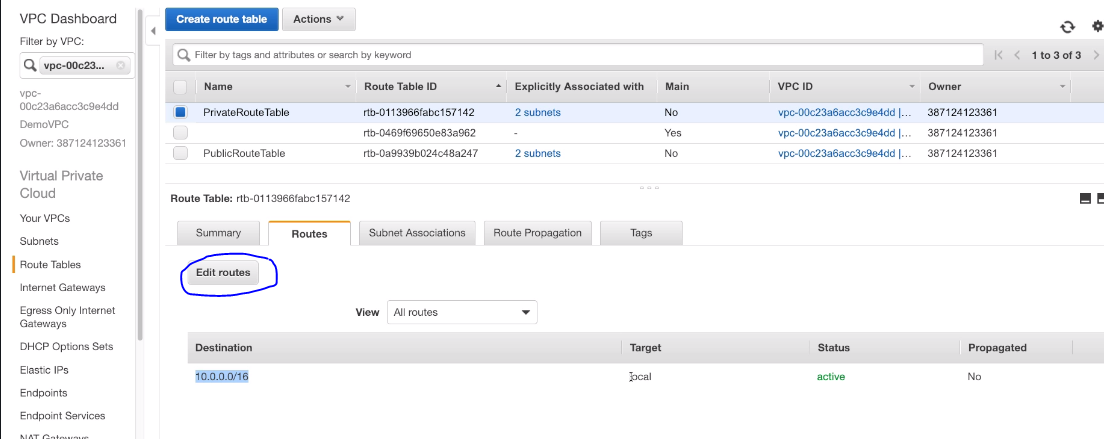
\*\* Create 2 route tables (Public and private)

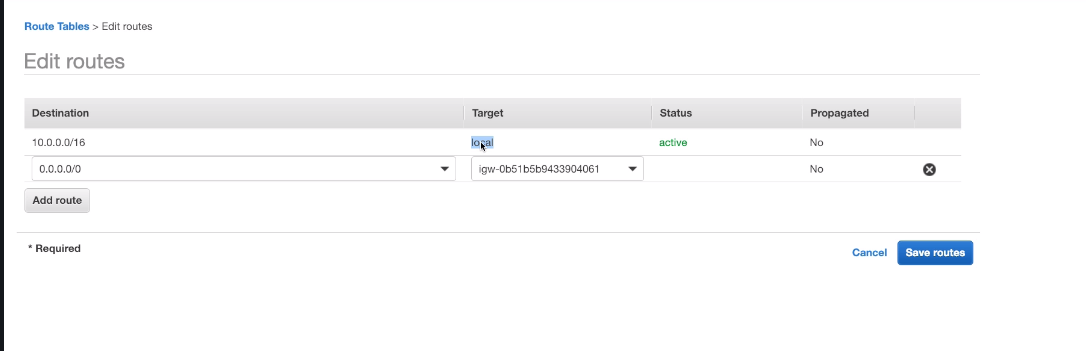
1. Add Subnets to route tables , private subnets to private route table and similarly public





1. Now, edit route table

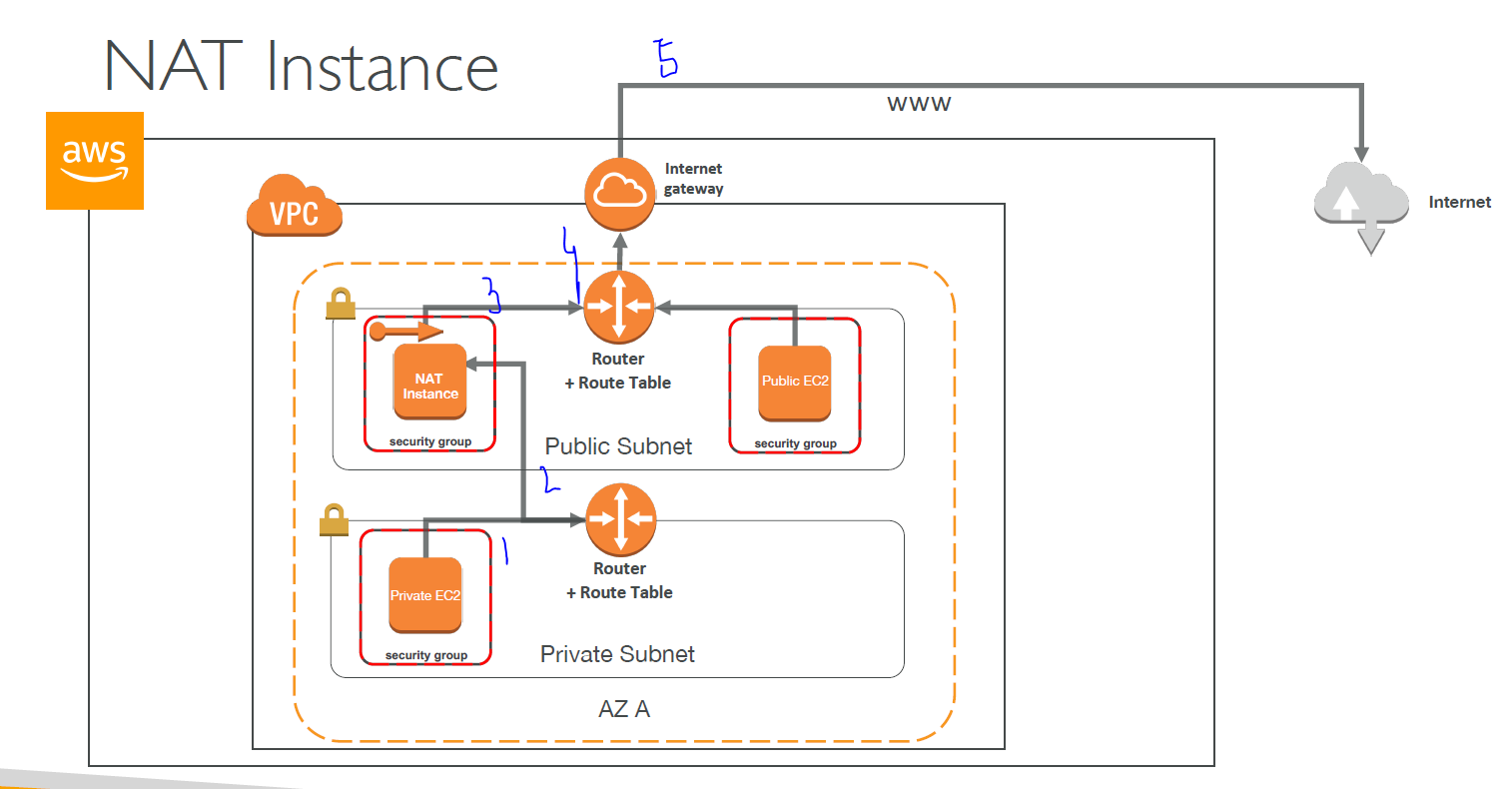




1. Now, EC2 instance will be accessible

**NAT (Network Address Translation) -** *Outdated but still comes in exam*

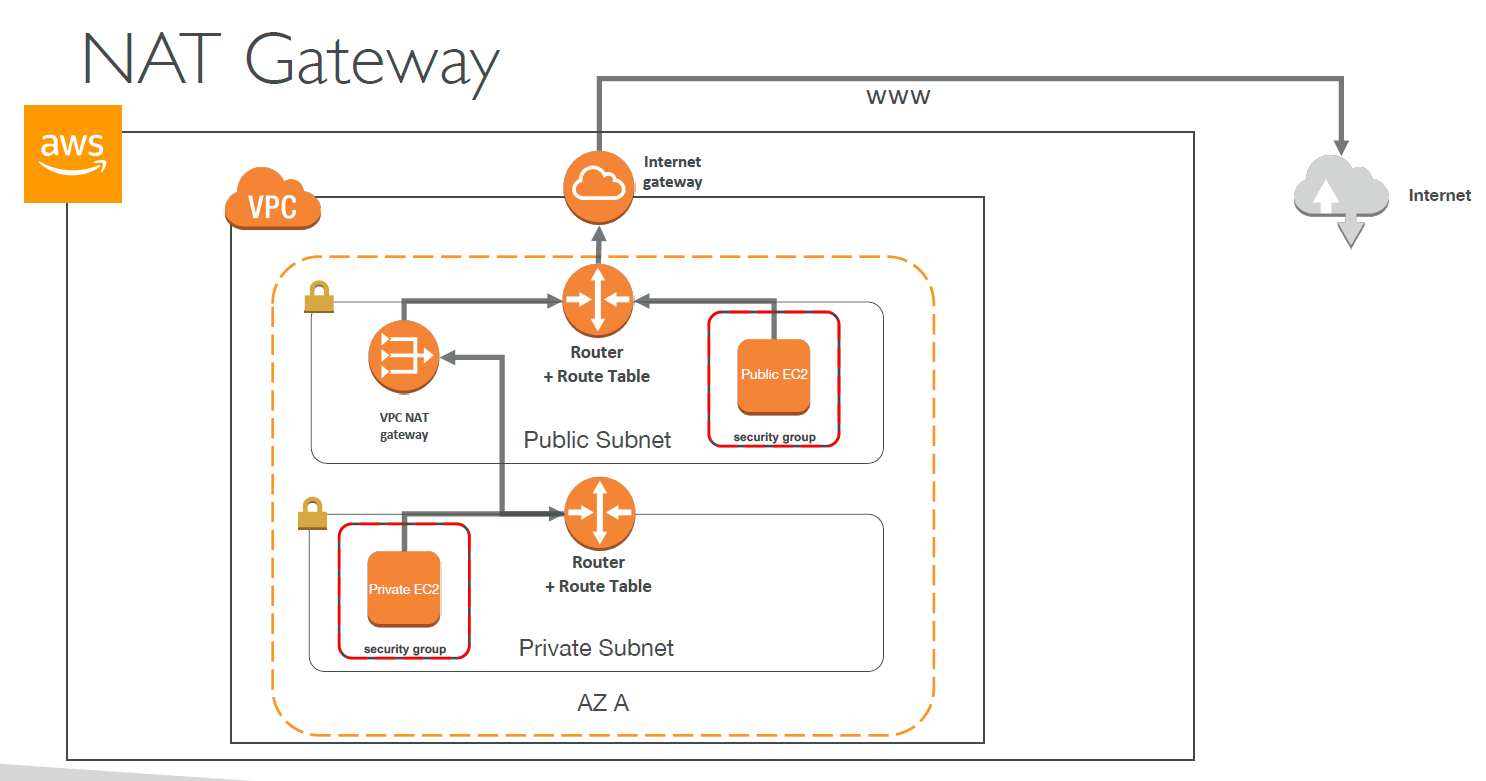
* Allows instances in the private subnets to connect to the internet
* Route table must be configured to route traffic from private subnets to NAT Instance



* Cons of NAT instance
  + Internet traffic bandwidth depends on EC2 instance performance
  + Must manage security groups & rules:
  + Inbound:
    - Allow HTTP / HTTPS Traffic coming from Private Subnets
    - Allow SSH from your home network (access is provided through Internet Gateway)
  + Outbound:
    - Allow HTTP / HTTPS traffic to the internet

**NAT GATEWAY**

* AWS managed NAT, higher bandwidth, better availability, no admin
* Pay by the hour for usage and bandwidth
* NAT is created in a specific AZ, uses an EIP
* Cannot be used by an instance in that subnet (only from other subnets)
* Requires an IGW (Private Subnet => NAT => IGW)
* 5 Gbps of bandwidth with automatic scaling up to 45 Gbps
* No security group to manage / required



**DNS Resolution in VPC**

* **enableDnsSupport**: (= DNS Resolution setting) ***– Right click on VPC – Edit DNS Resolution***
  + Default True
  + Helps decide if DNS resolution is supported for the VPC
  + If True, queries the AWS DNS server at 169.254.169.253
* **enableDnsHostname**: (= DNS Hostname setting) ***– Right click on VPC – Edit DNS hostname***
  + False by default for newly created VPC, True by default for Default VPC
  + Won’t do anything unless enableDnsSupport=true
  + If True, Assign public hostname to EC2 instance if it has a public
* If you use custom DNS domain names in a private zone in Route 53, you must set both these attributes to true

**SECURITY IN VPC – NETWORK ACL (NACL)**

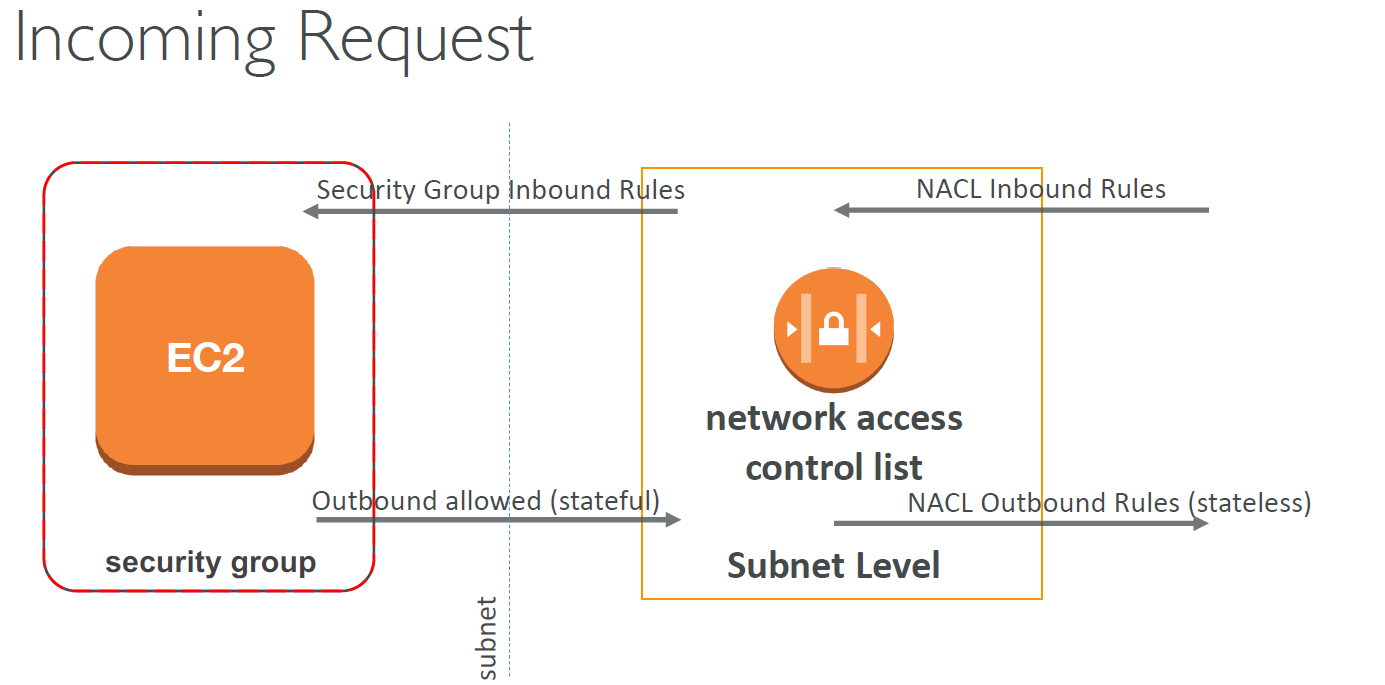
* Till now, we have seen EC2 instance has security group for security purpose and its present in subnet and security group only controls traffic in subnet. But NACL are used to control traffic from and to subnet.

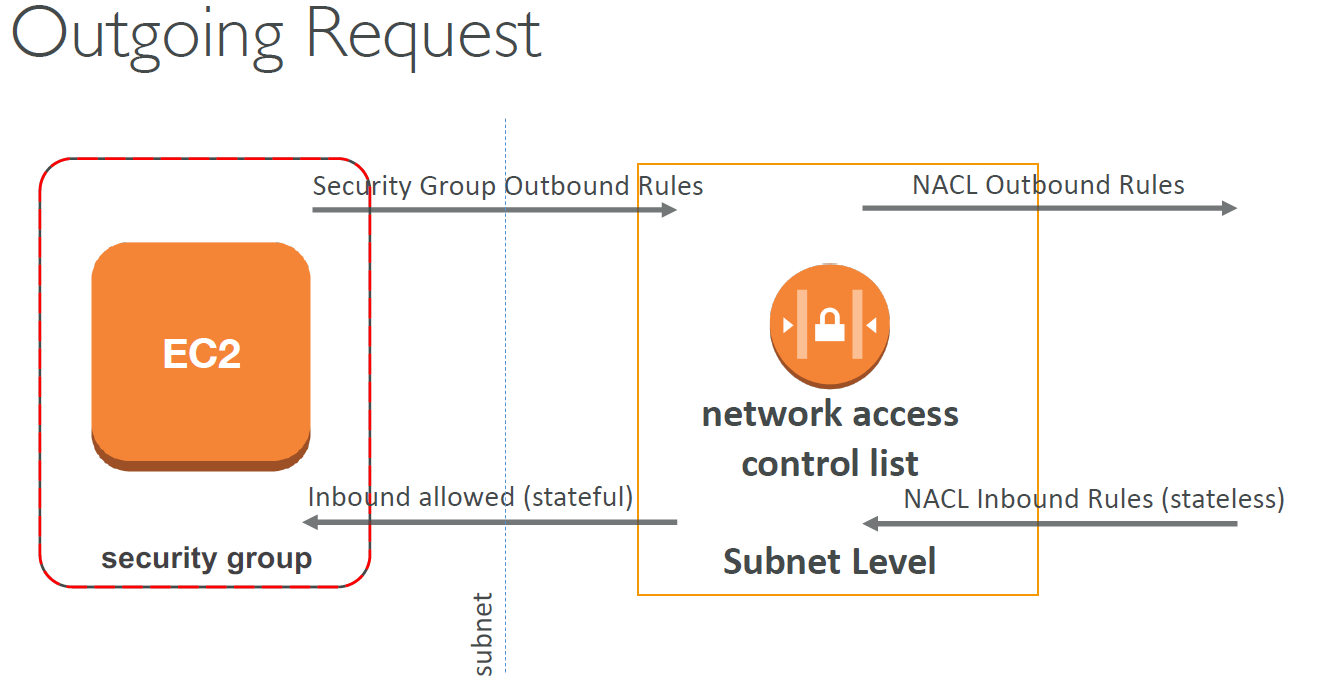
**NACL**

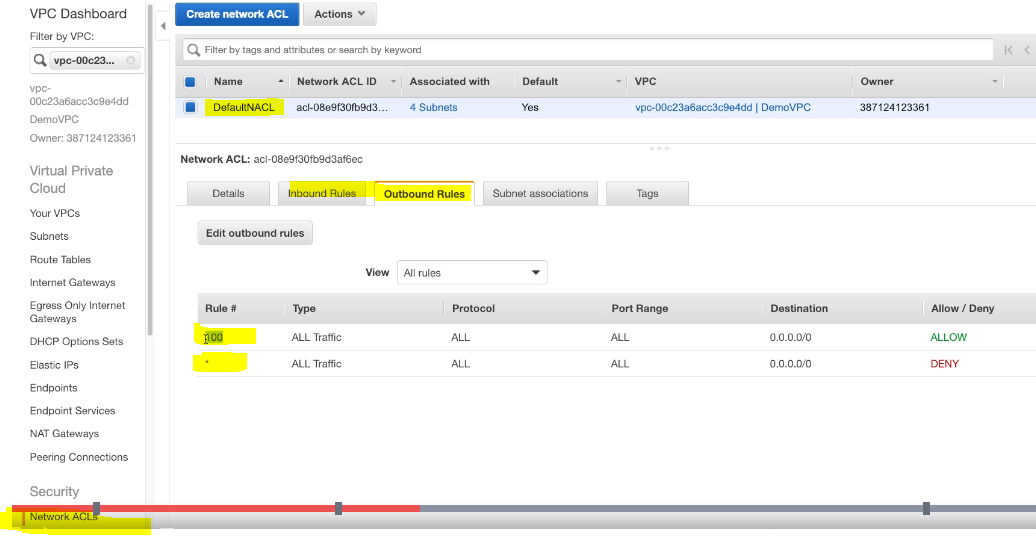
* NACL are like a firewall which control traffic from and to subnet
* Default NACL allows everything outbound and everything inbound. But Newly created NACL will deny everything
* One NACL per Subnet, new Subnets are assigned the Default NACL
* NACL are a great way of blocking a specific IP at the subnet level
* **Define NACL rules:**
  + Rules have a number (1-32766) and higher precedence with a lower number

E.g. If you define #100 ALLOW <IP> and #200 DENY <IP>, IP will be allowed

* + Last rule is an asterisk (\*) and denies a request in case of no rule match
  + AWS recommends adding rules by increment of 100

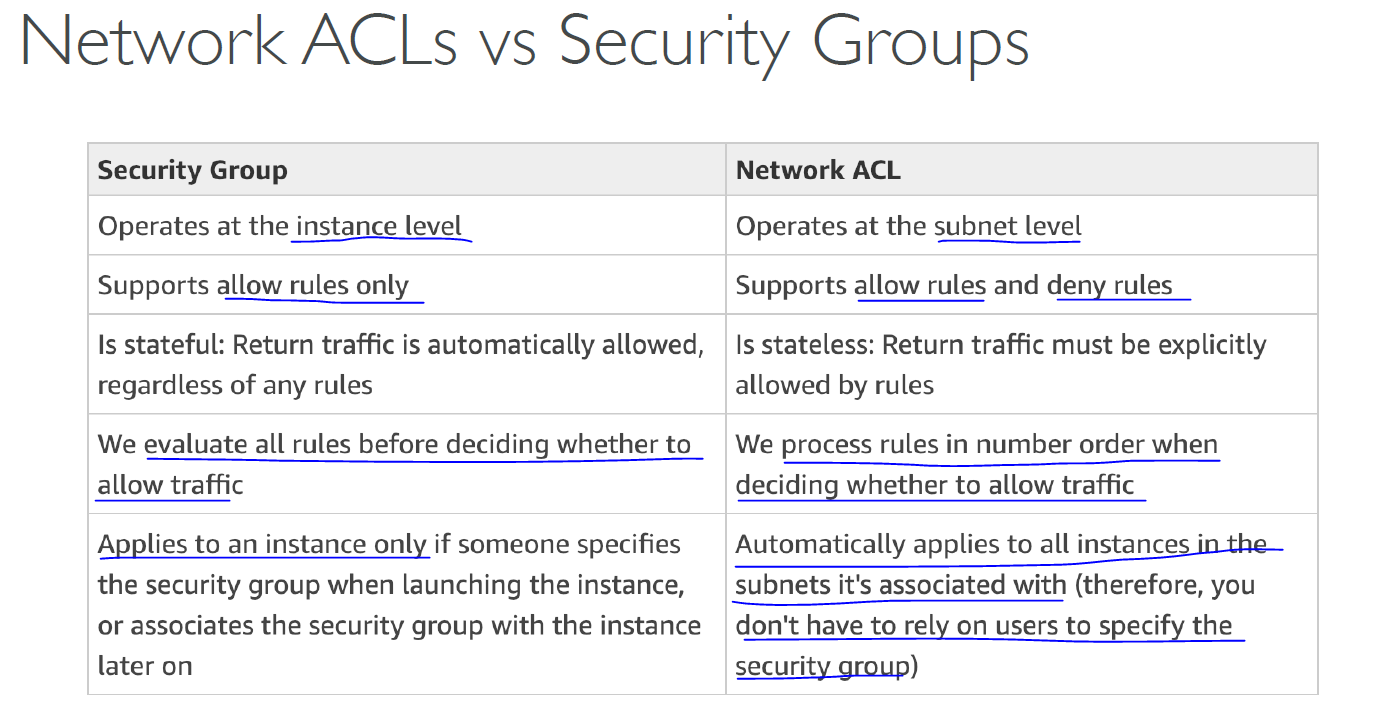






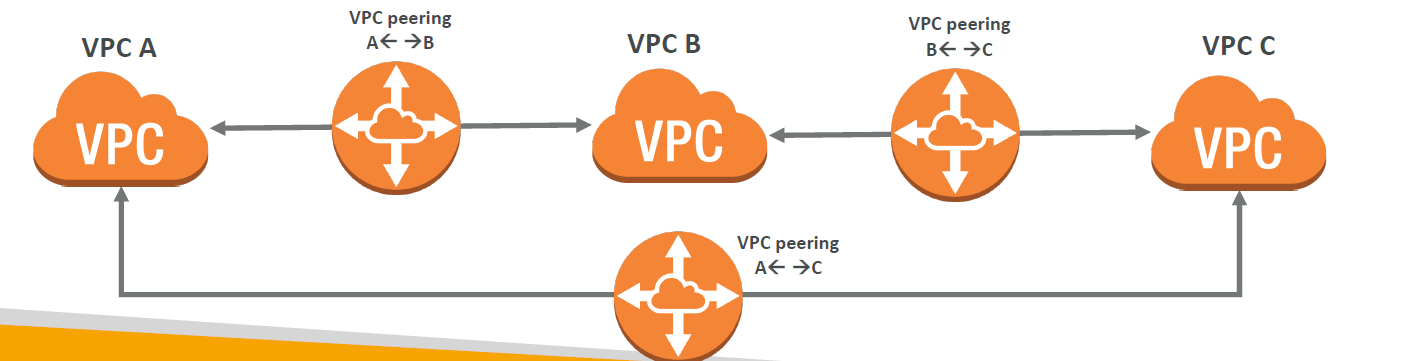
\*\* Inbound and outbound has same rule for default NACL and Rule #100 – allow all traffic and if rule#1 doesn’t match Rule \* will applicable and block all traffic

* Create or Edit above Default inbound and outbound rule and observe traffic on instance



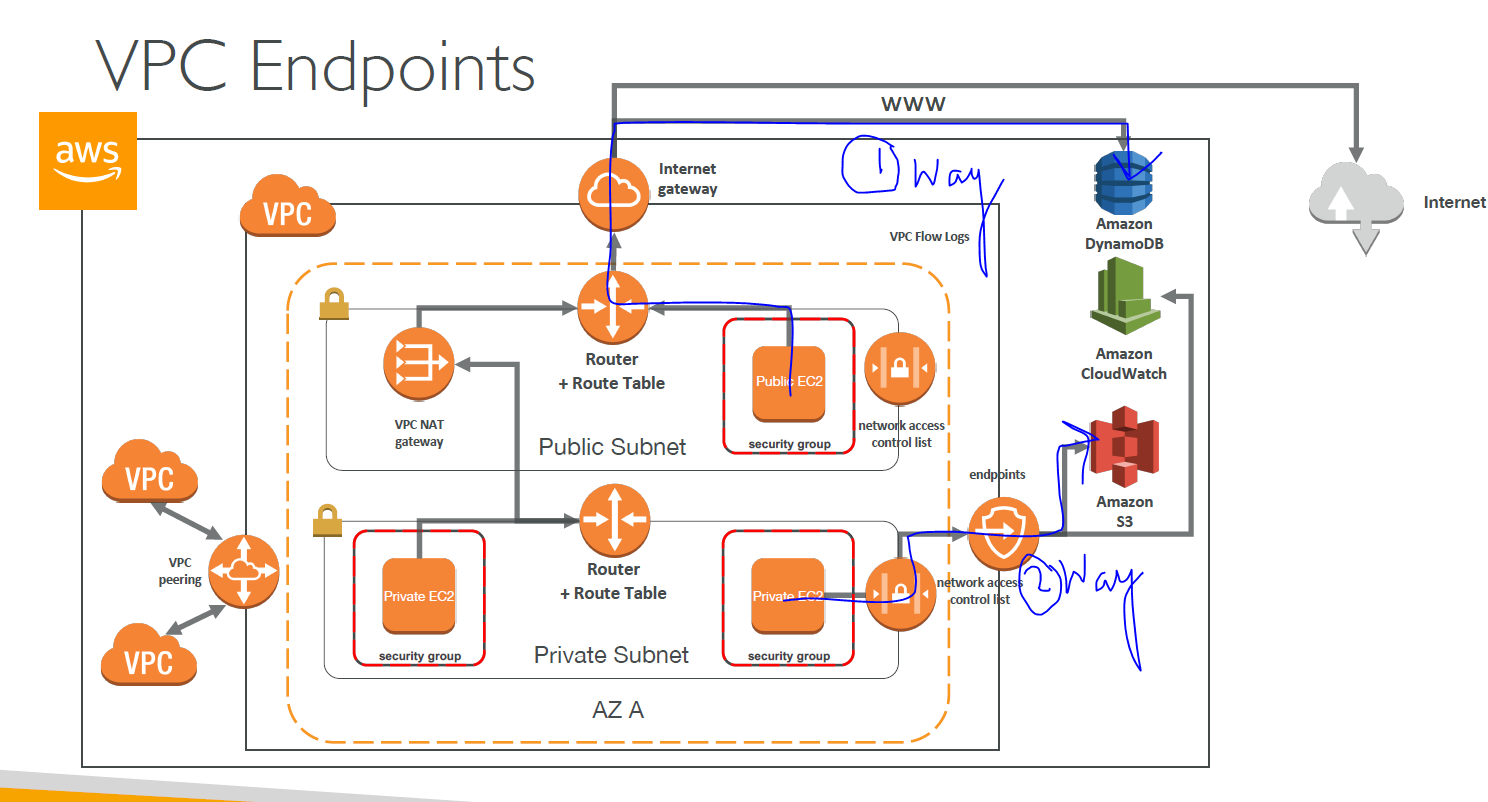
**VPC PEERING**

* Connect two VPC, privately using AWS’ network
* Make them behave as if they were in the same network
* Make sure different VPC’s must not have overlapping CIDR
* VPC Peering connection is not transitive (must be established for each VPC that need to communicate with one another)
* VPC peering can work inter-region, cross-account
* You must update route tables in each VPC’s subnets to ensure instances can communicate



**VPC Endpoints**

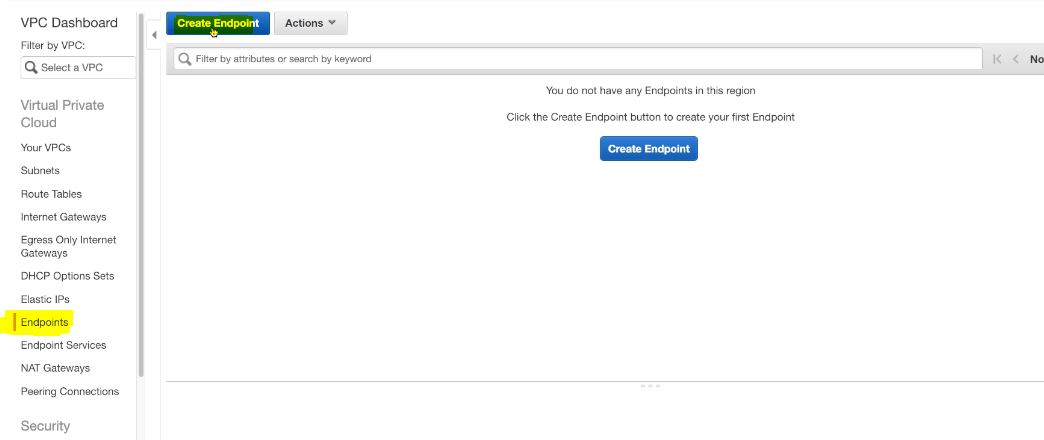
* Endpoints allow you to connect to AWS Services using a private network instead of the public www network
* They scale horizontally and are redundant
* They remove the need of IGW, NAT, etc… to access AWS Services
* **Interface**: provisions an ENI (private IP address) as an entry point (must attach security group) – most AWS services
* **Gateway**: provisions a target and must be used in a route table – S3 and DynamoDB services have gateway endpoint while other services have interface end point
* **In case of issues**:
  + Check DNS Setting Resolution in your VPC
  + Check Route Tables



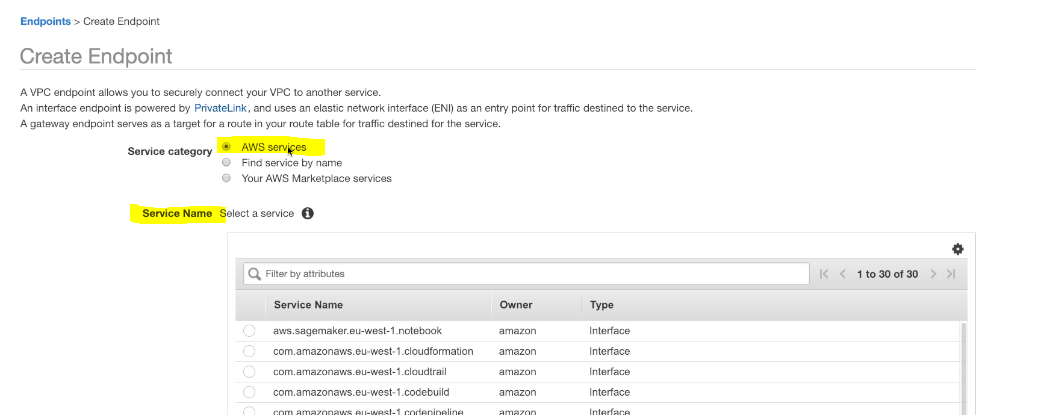
**Hands on**

* Earlier to access s3 from EC2, we have to create IAM role with S3 access and that role is attached to EC2 instance, But this approach will access S3 from EC2 via internet and same can be verified if you remove NAT gateway.Hence, we use VPC endpoints to avoid public internet.

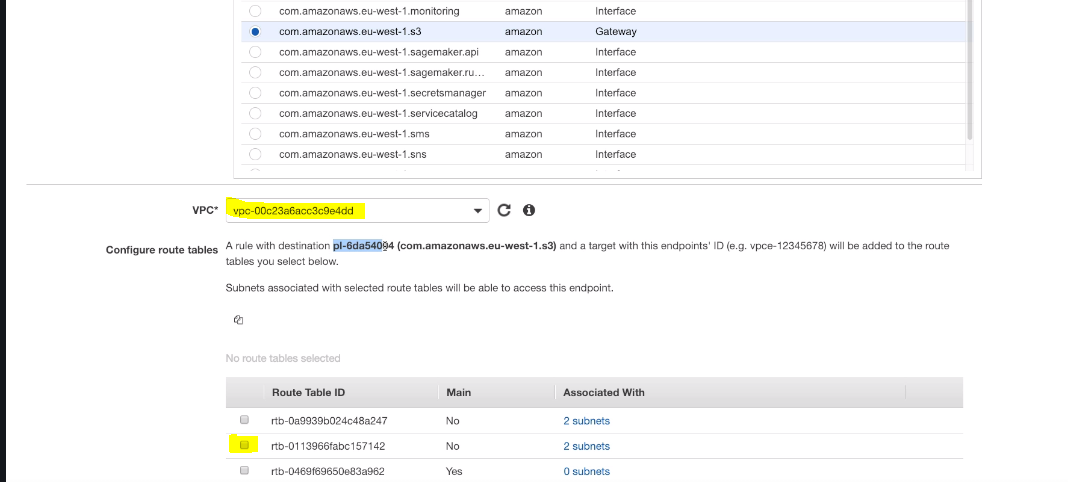
1. Go to Endpoints in VPC dashboard.



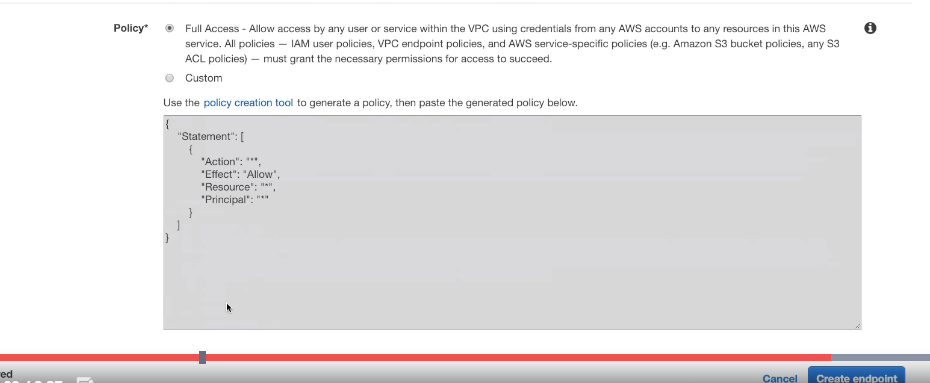
1. Select service category and Name



1. Select VPC for endpoint and choose Route table out of tables available in VPC

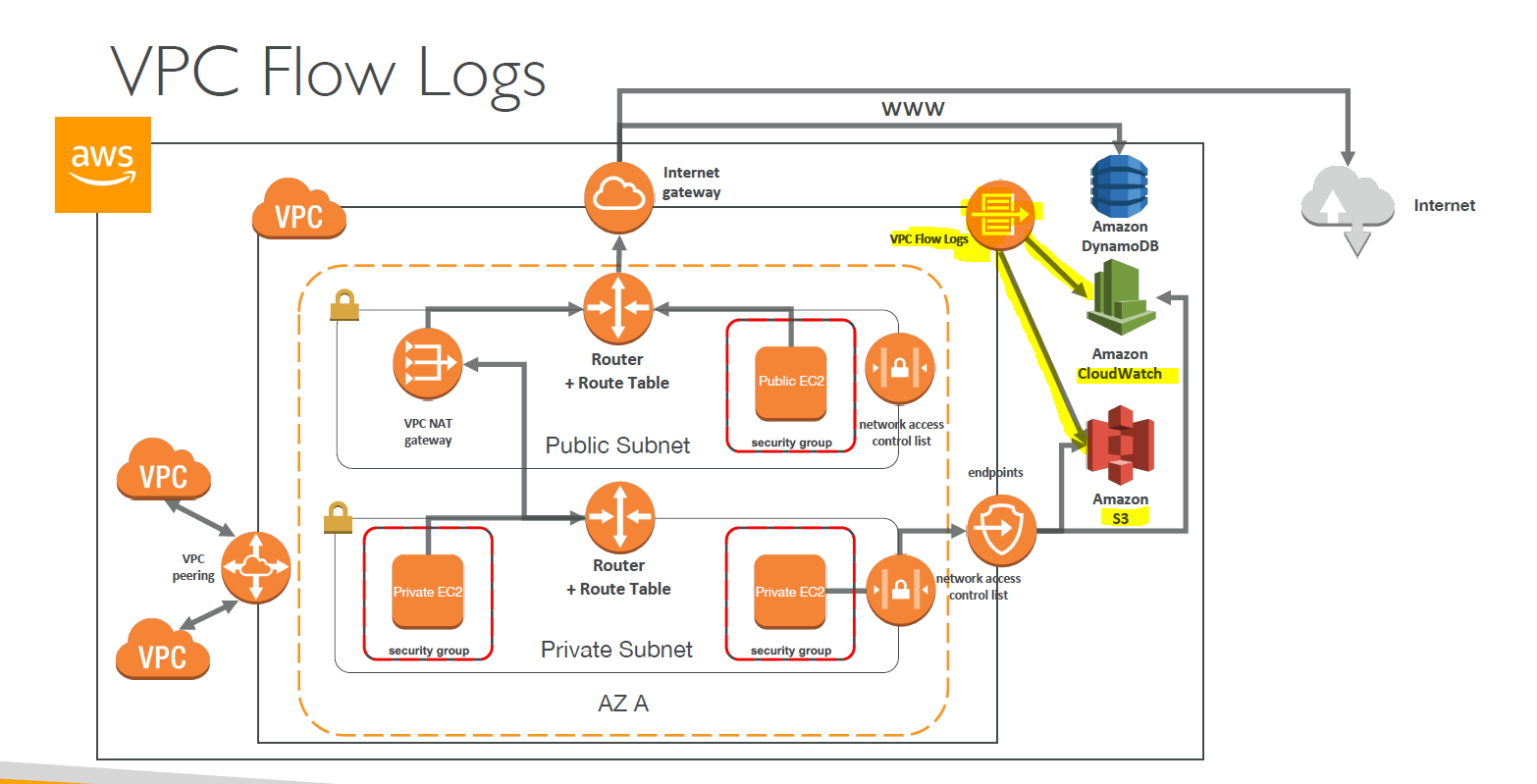


1. Choose Policy and click create endpoint



**FLOW LOGS**

* Capture information about IP traffic going into your interfaces:
  + VPC Flow Logs – includes all logs of network (subnet too)
  + Subnet Flow Logs
  + Elastic Network Interface Flow Logs
* Helps to monitor & troubleshoot connectivity issues
* Flow logs data can go to S3 / CloudWatch Logs
* Captures network information from AWS managed interfaces too: ELB, RDS, ElastiCache, Redshift, WorkSpaces



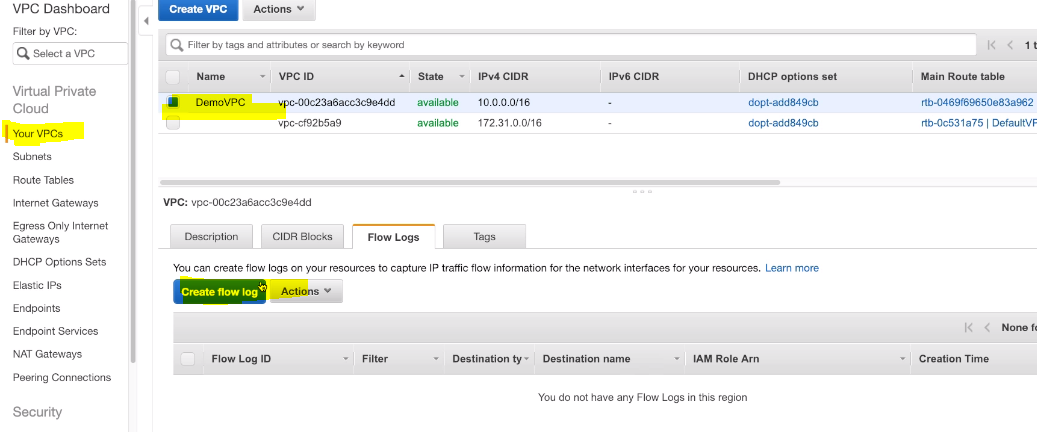
* **Flow Log Syntax**
  + <version> <account-id> <interface-id> <srcaddr> <dstaddr> <srcport>

<dstport> <protocol> <packets> <bytes> <start> <end> <action> <logstatus>

* + Can be used for analytics on usage patterns, or malicious behavior
  + Flow logs example: <https://docs.aws.amazon.com/vpc/latest/userguide/flowlogs.html#flow-log-records>
  + Query VPC flow logs using Athena on S3 or CloudWatch Logs Insights

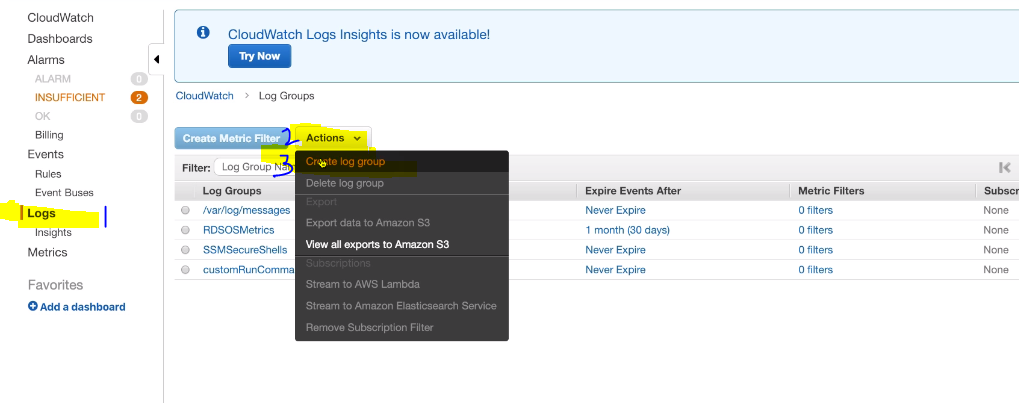
**Hands on**

1. Go to VPC in which you have to create flow logs

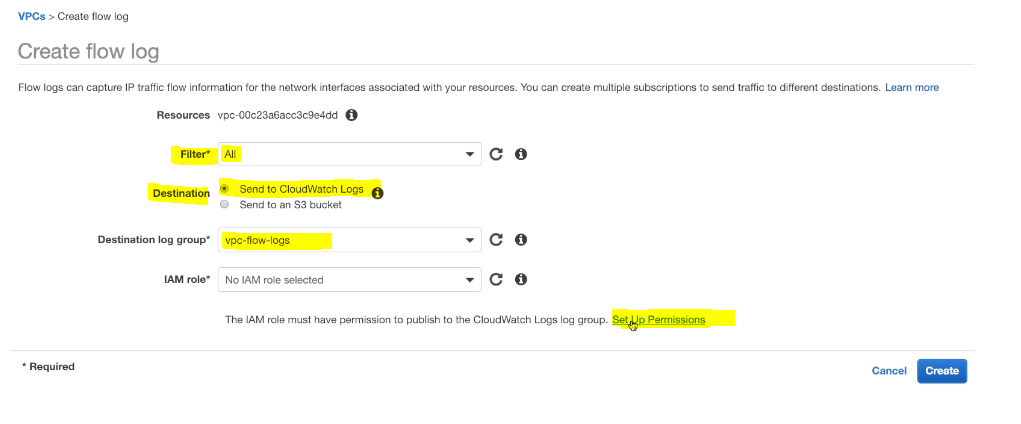


**Send logs to cloudwatch logs**

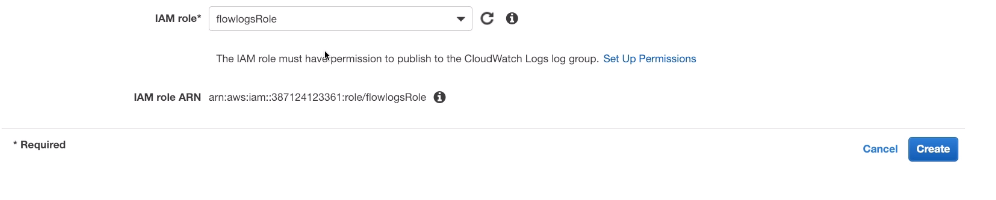
1. Create cloud watch logs group -> just give log group name , it will be created



1. Select Filer (Accept/Reject/All) , Destination , select log group that you have created in above step in cloud watch logs

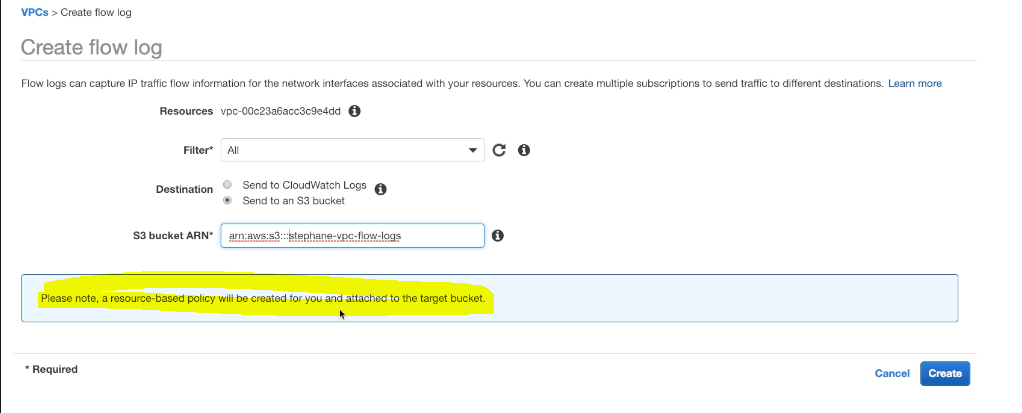


1. Click on Set Up Permissions, and AWS will automatically create IAM Role. Then, select from drop down create role and click create

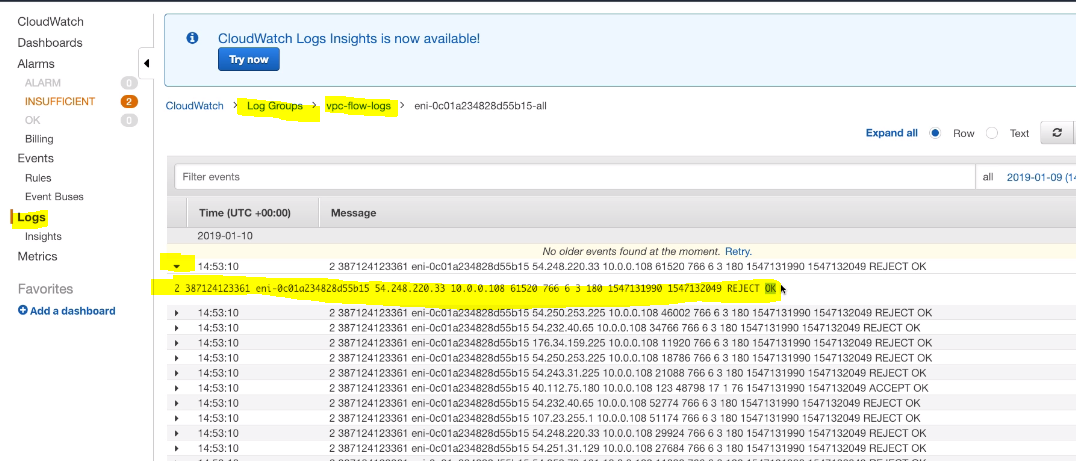


**Send logs to S3**

1. Create Bucket and add its ARN while creating Flow logs



* View Your logs in cloud watch or S3

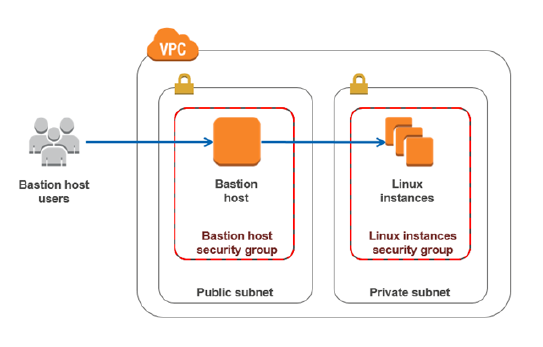


* **You can analyzed these logs in ATHENA**

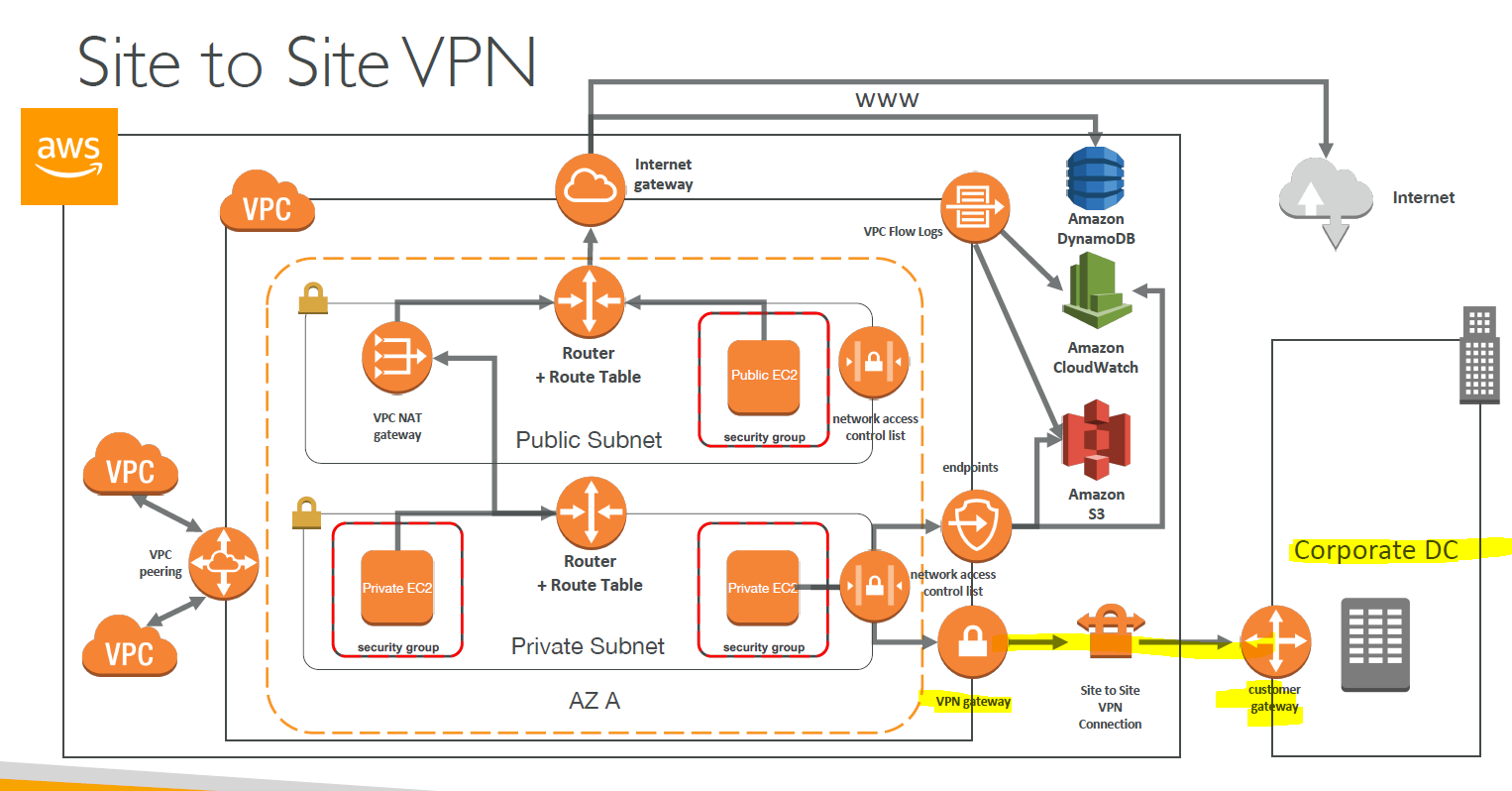
<https://docs.aws.amazon.com/athena/latest/ug/vpc-flow-logs.html>

**BASTION HOSTS**

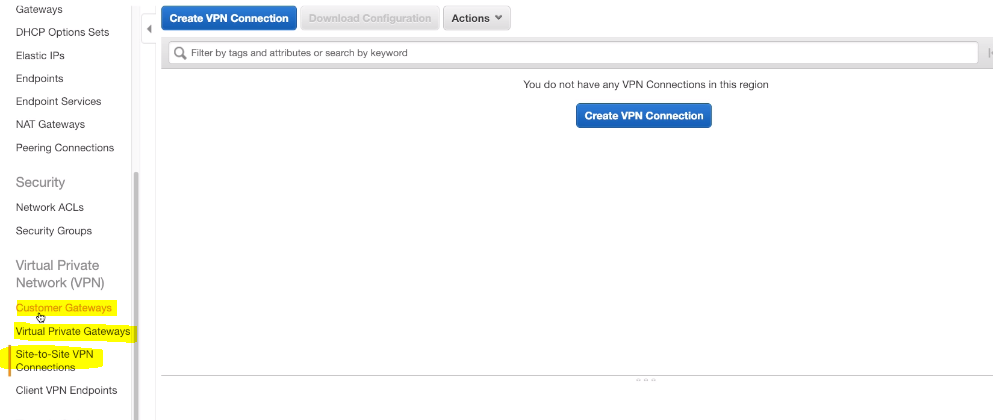
* We can use a Bastion Host to SSH into our private instances
* The bastion is in the public subnet which is then connected to all other private subnets
* Bastion Host security group must be tightened
* **Exam Tip**: Make sure the bastion host only has port 22 traffic from the IP you need, not from the security groups of your other instances



**VIRTUAL PRIVATE GATEWAY, CUSTOMER GATEWAY and SITE-TO-SITE VPN CONNECTION**

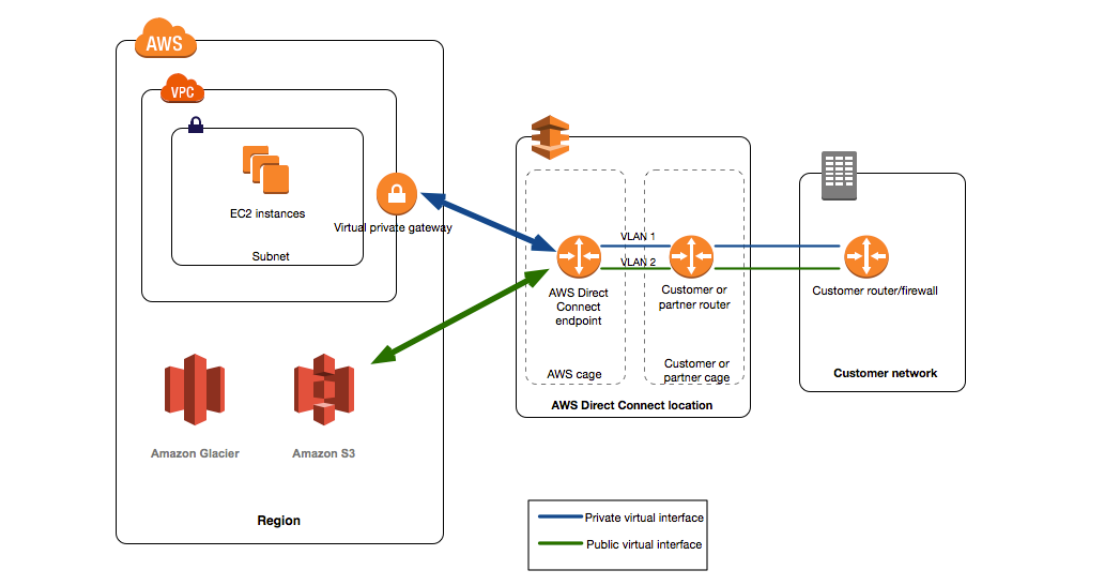


* **VPN** – virtual private network – encrypted connection that allows to connect to private network via internet.
* **VIRTUAL PRIVATE GATEWAY:**
  + VGW is created and attached to the VPC from which you want to create the Site-to Site VPN connection
* **CUSTOMER GATEWAY**
  + Software application or physical device on customer side of the VPN connection



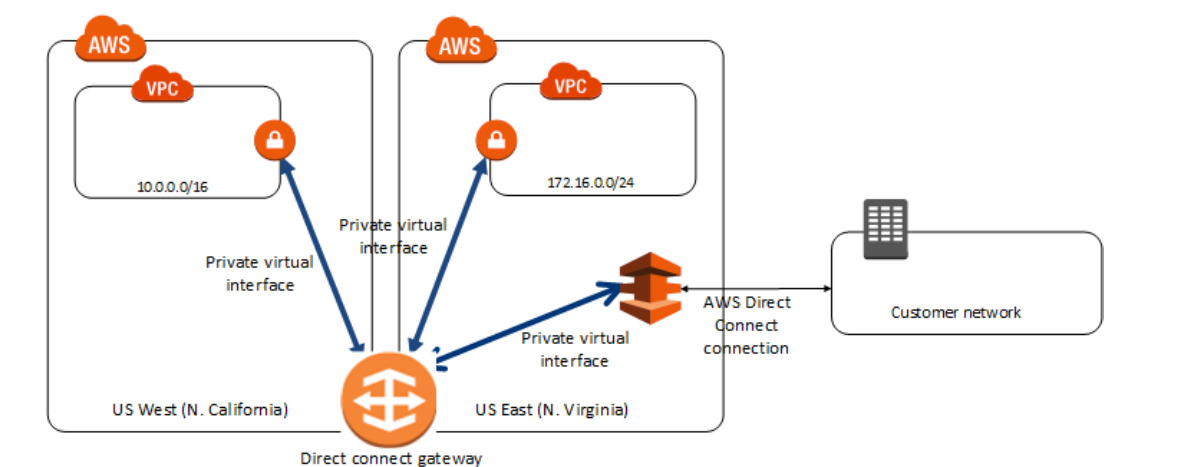
**DIRECT CONNECT**

* 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
* 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)



**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**

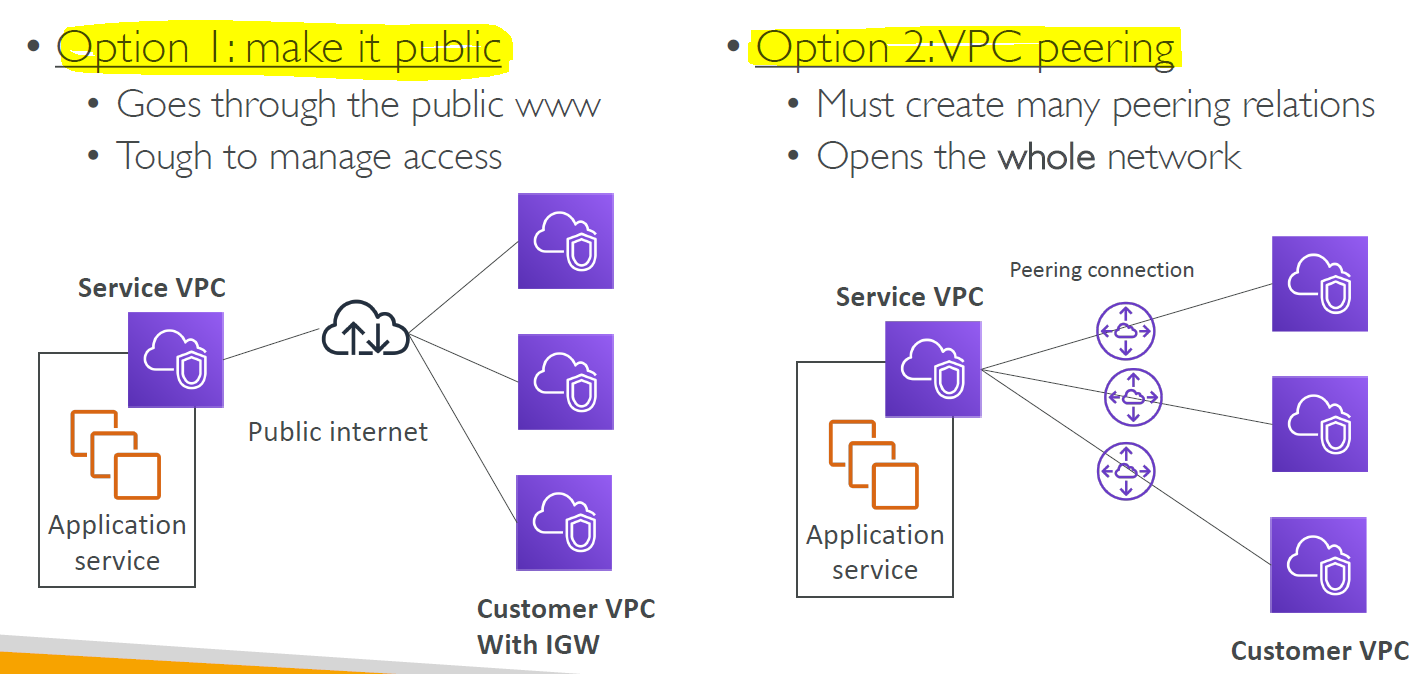
* **Dedicated Connections**: 1Gbps and 10 Gbps capacity
  + Physical ethernet port dedicated to a customer
  + Request made to AWS first, then completed by AWS Direct Connect Partners
* **Hosted Connections**: 50Mbps, 500 Mbps, to 10 Gbps
  + Connection requests are made via AWS Direct Connect Partners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

**EGRESS ONLY INTERNET GATEWAY (Egress – going out)**

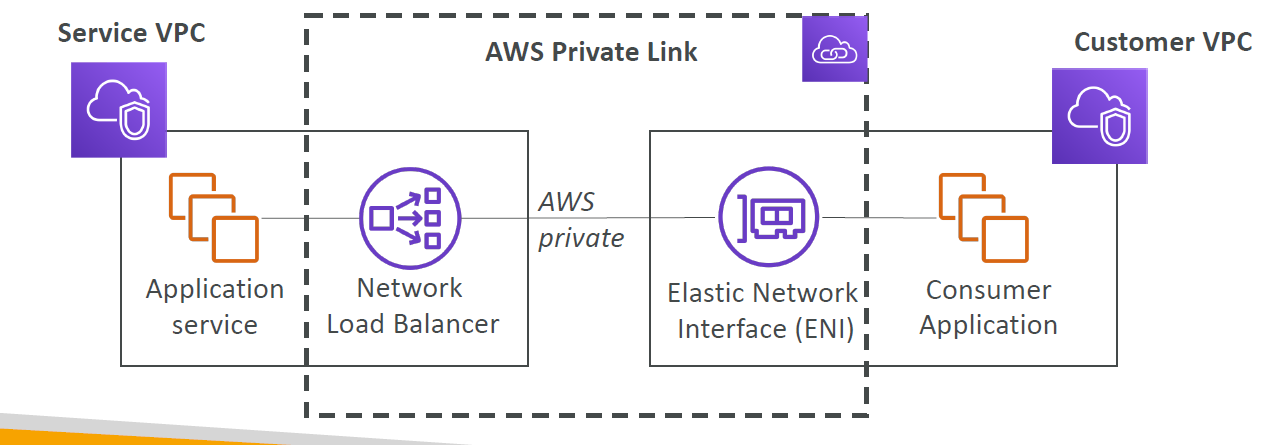
* for IPv6 only - Similar function as a NAT, but a NAT is for IPv4
* Good to know: IPv6 are all public addresses
* Therefore all our instances with IPv6 are publicly accessibly
* Egress Only Internet Gateway gives our IPv6 instances access to the internet, but they won’t be directly reachable by the internet
* After creating an Egress Only Internet Gateway, edit the route tables

**AWS PRIVATELINK - VPC Endpoint Services**

* Suppose, you have to expose AWS services from one VPC to another VPC



* **Endpoint services** solves above problem
* Most secure & scalable way to expose a service to 1000s of VPC (own or other accounts)
* Does not require VPC peering, internet gateway, NAT, route tables…
* Requires a network load balancer (Service VPC) and ENI (Customer VPC)
* If the NLB is in multiple AZ, and the ENI in multiple AZ, the solution is fault tolerant!

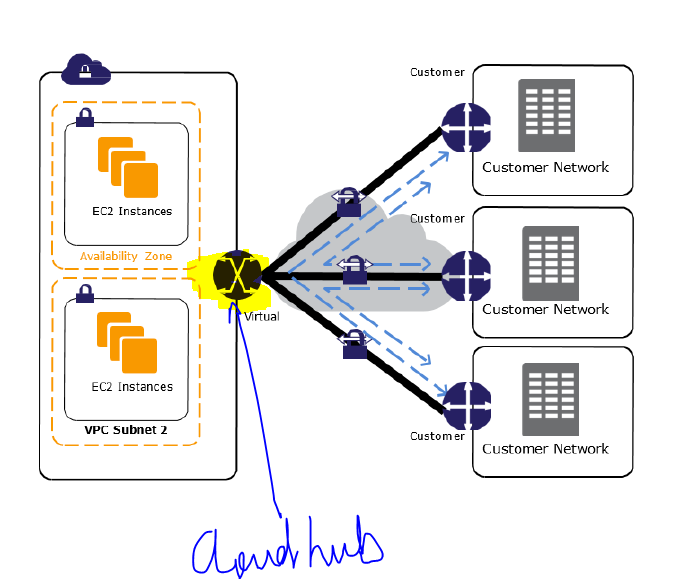


**EC2-Classic & AWS ClassicLink (deprecated)**

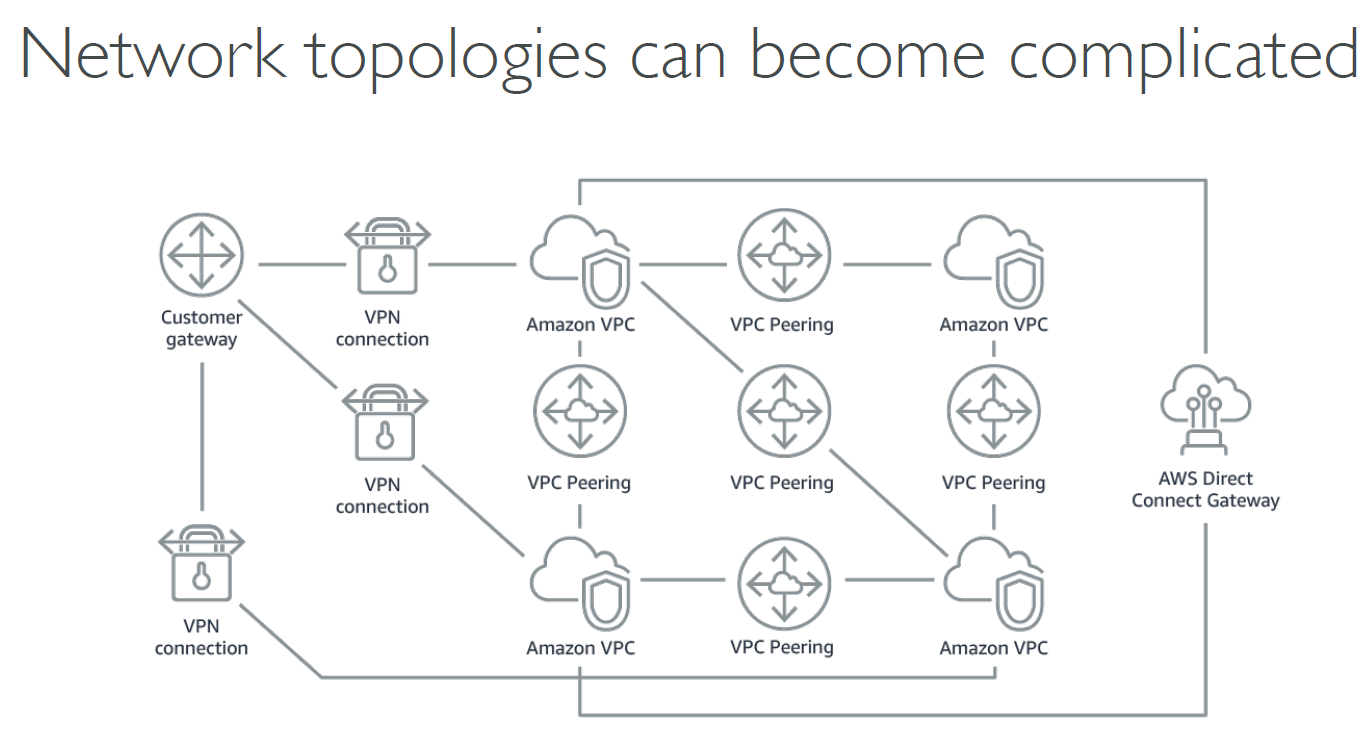
* **EC2-Classic:** EC2 instances run in a single network shared with other customers
* **Amazon VPC**: your instances run logically isolated to your AWS account
* **ClassicLink** allows you to link EC2-Classic instances to a VPC in your account
  + Must associate a security group
  + Enables communication using private IPv4 addresses
  + Removes the need to make use of public IPv4 addresses or Elastic IP addresses
* **Likely to be distractors at the exam**

**AWS VPN CloudHub**

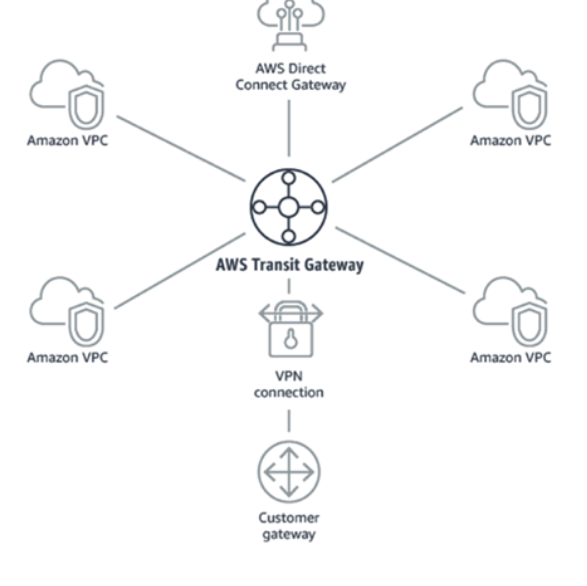
* Provide secure communication between sites, if you have multiple VPN connections
* Low cost hub-and-spoke model for primary or secondary network connectivity between locations
* It’s a VPN connection so it goes over the public internet



**TRANSIST GATEWAY**

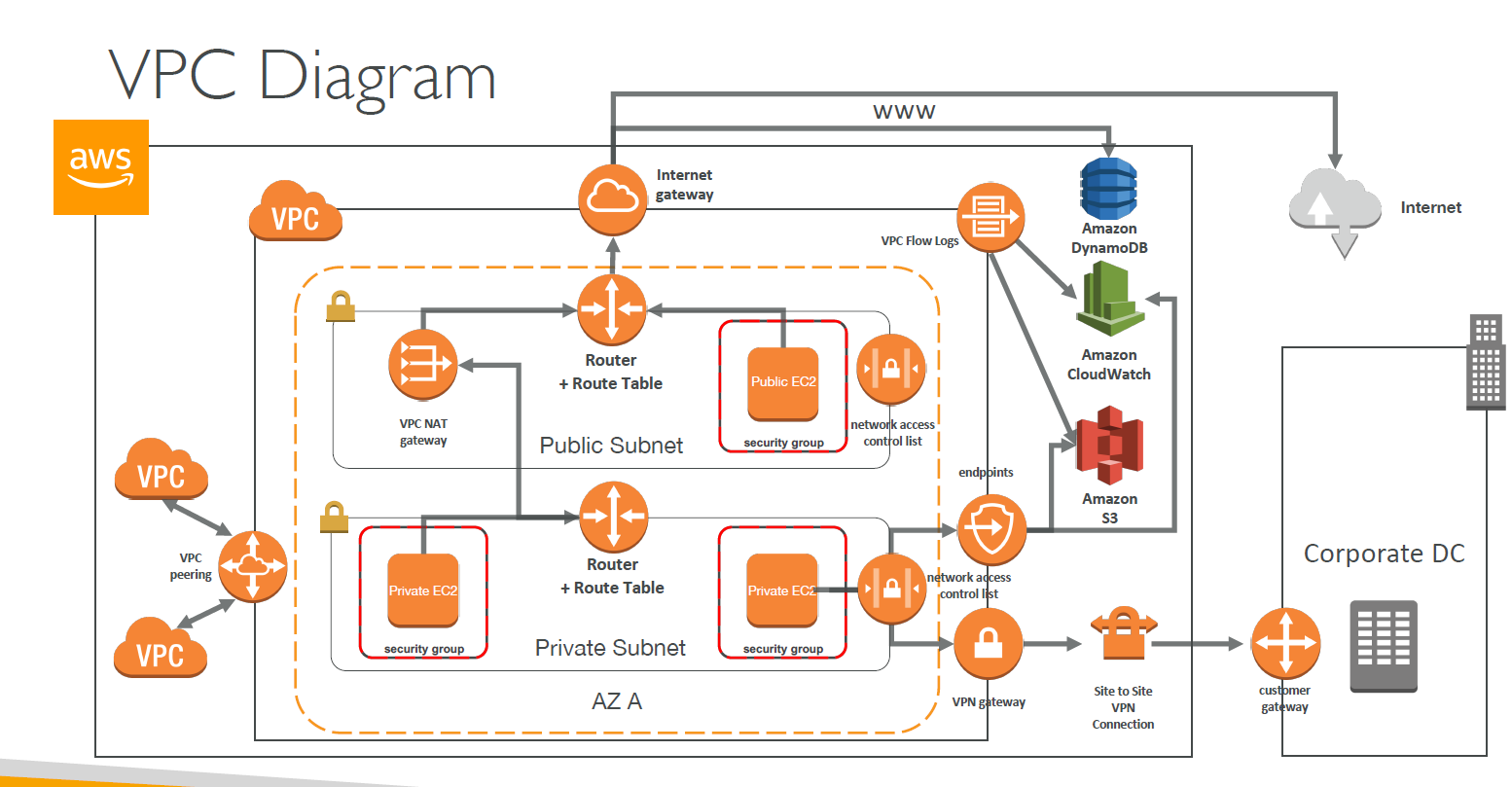


* **For having transitive peering between thousands of VPC and on-premises, hub-and-spoke (star) connection**



* 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

**VPC SUMMARY**



* **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, etc…

* **NAT Instances**: gives internet access to 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 instances, IPv4
* only
* **Private DNS + Route 53**: enable DNS Resolution + DNS hostnames (VPC)
* **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 VPC with non overlapping CIDR, non transitive
* **VPC Endpoints**: Provide private access to AWS Services (S3, DynamoDB, CloudFormation,SSM) within VPC
* **VPC Flow Logs**: Can be setup at the VPC / Subnet / ENI Level, for ACCEPT and REJECTtraffic, helps identifying attacks, analyze using Athena or CloudWatch Log Insights
* **Bastion Host**: Public instance to SSH into, that has SSH connectivity to instances in private subnets
* **Site to Site VPN:** setup a Customer Gateway on DC, a Virtual Private Gateway on VPC, and site-to-site VPN over public internet
* **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 VPC in different regions
* **Internet Gateway Egress**: like a NAT Gateway, but for IPv6
* **Private Link / 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 instances privately to your VPC
* **VPN CloudHub**: hub-and-spoke VPN model to connect your sites
* **Transit Gateway:** transitive peering connections for VPC, VPN & DX