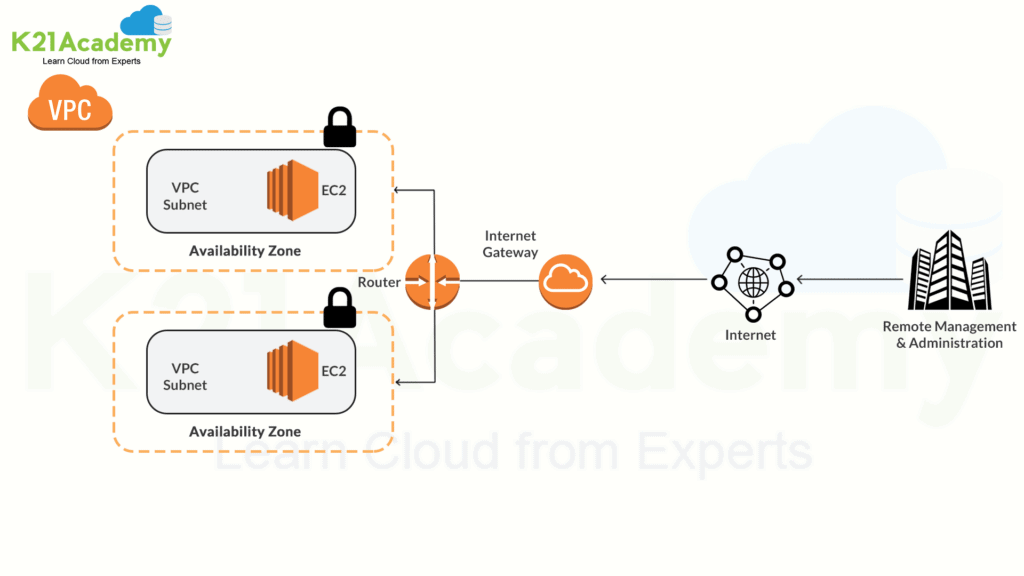
**AWS Networking**

**Networking Basics in AWS:**

**Amazon VPC**

Amazon VPC or Virtual Private Cloud is a service that allows us to create an isolated virtual network for our Amazon resources. A virtual network is a private network that is always hidden from the outside world, and you can perform certain operations that you don’t want to make public. Any user with their AWS account can host Amazon VPC. You can create, access, and manage Amazon VPC with the help of certain tools and services like the Amazon Web Service Management Console, Amazon CLI (Command Line Interface), Amazon SDK, and Query API.



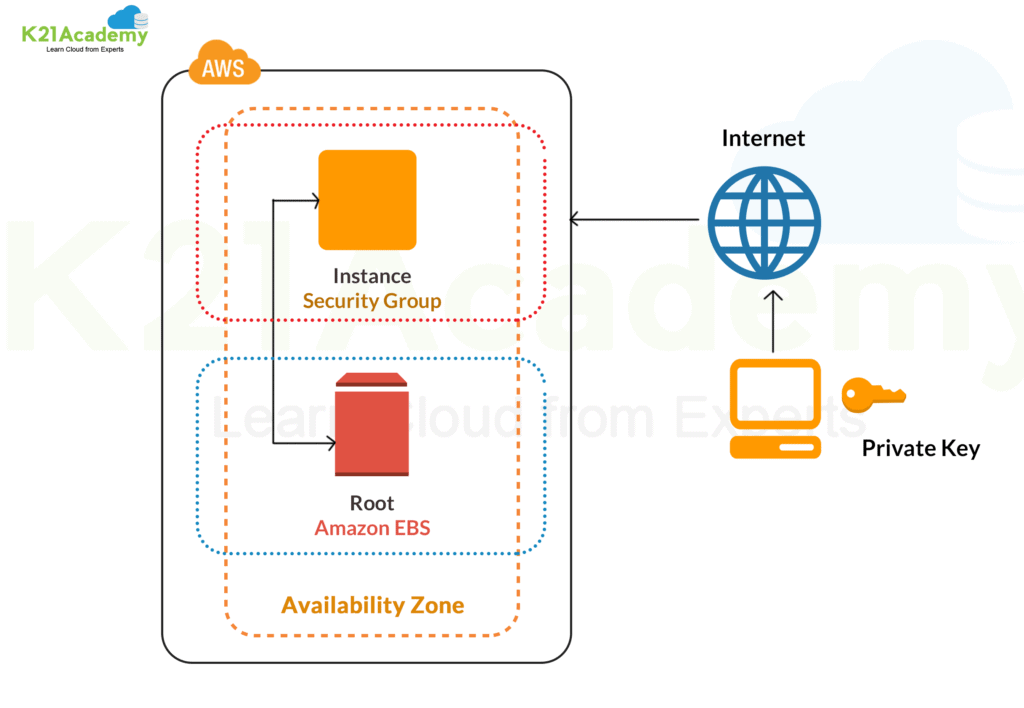
**Some Basic Concepts for Amazon VPC**

* **Subnet** – A subpart of your network with a dedicated range of IP addresses.
* **Route Table** – It is used to direct traffic with a set of rules called routes.
* **Gateway** – It enables communication between your network and resources.

**Also Read:**Our blog post on [**AWS SNS**](https://k21academy.com/amazon-web-services/aws-solutions-architect/amazon-simple-notification-service/).

**Amazon EC2**

Amazon Elastic Compute Cloud or EC2 is one of the most used AWS services to create and launch virtual machine instances. It provides a secure and resizable compute capacity in the cloud. It helps create various virtual machines instances within the cloud that can be easily set up in a while. For Amazon Networking, Amazon EC2 is one of the key services.



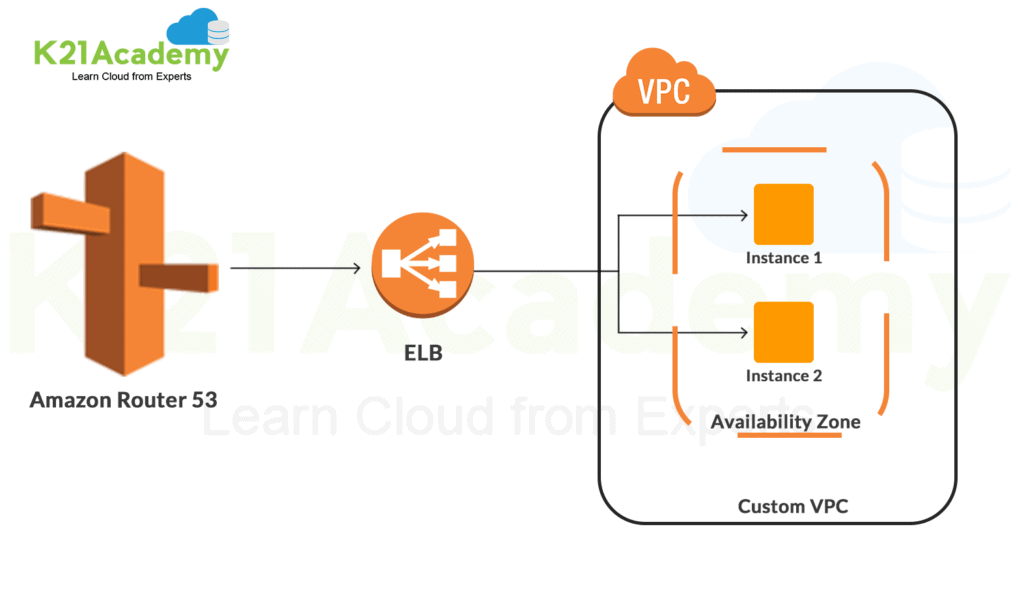
**Key Features of Amazon EC2 Instance**

* Uses predefined and preconfigured Amazon Machine Images (AMI) for easy setup.
* It allows you to configure memory, storage, and networking capacity for the instance based on your need.
* AWS EC2 instance is secured with key pairs for storing login information.
* Uses a secure firewall to specify IP, range, ports, and protocols to access your instance.
* Uses Instance Store Volumes to preserve your temporary data if you stop, hibernate, or terminate the instance.

**Check Out:**Our blog post on [**AWS Secrets Manager**](https://k21academy.com/amazon-web-services/aws-solutions-architect/aws-secrets-manager/).

**Amazon Route 53**

Amazon Route 53 is a scalable and highly available Domain Name System (DNS) service. It enables to route end users to internet applications reliably and cost-effectively. Route 53 connects user requests infrastructure inside or outside of AWS Infrastructure, including Amazon EC2 instances, Elastic Load Balancers, Amazon S3 buckets, and more.



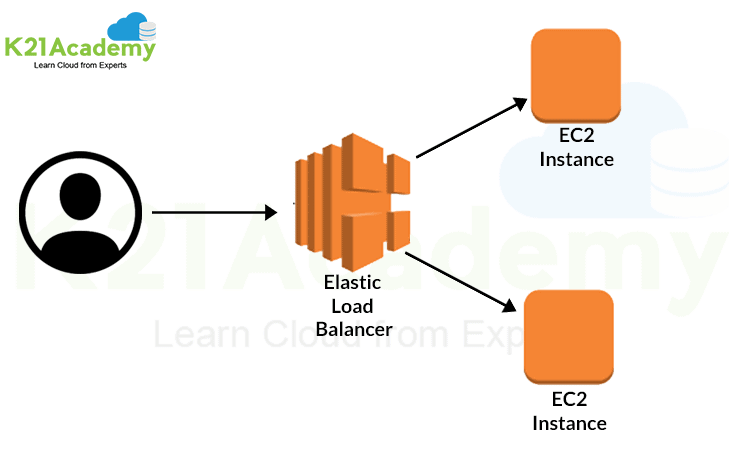
**Key Benefits of Amazon Route 53**

* **Highly Available and Reliable** – In case of failover Amazon Route 53 Traffic Flow routs users to an alternate location if the primary location is unavailable.
* **Flexible and Fast** – A user can create and edit traffic policies. Also, depending on network conditions, it routes users to optimal locations.
* **Simple and Secure** – You can configure DNS settings using AWS Management Console or an easy-to-use API. Also, by integrating Amazon Route 53 with [IAM](https://k21academy.com/amazon-web-services/aws-solutions-architect/aws-identity-and-access-management-iam/) (Identity Access Management), you can create and manage permissions for each user with unique credentials.
* **Scalable and Cost-Effective**– Route 53 can handle large query volumes by auto-scaling. Also, your charges for Route 53 depend on the resources and the number of queries.

**Also Read:**Our blog post on [**Cloud Service Models**](https://k21academy.com/amazon-web-services/aws-solutions-architect/cloud-service-models/).

**Amazon Load Balancing**

Elastic Load Balancing is a load balancing service for AWS. It helps in automatically distributing incoming traffic from various cloud applications and scaling resources to meet the traffic demands. It works over various resources like Amazon EC2 instances, containers, IP addresses, and lambda functions.



**Key Features of Amazon Load Balancing**

* **High Availability** – It is a fully managed service and provides high availability by keeping your applications available across a region without the need for GSLB (Global Server Load Balancing).
* **Security** – It provides robust security features like integrated certificate management, SSL/TLS decryption, and user authentication.
* **Robust Monitoring** – It allows you to monitor the performance and health of your applications in real-time with the help of Amazon Cloud Watch metrics, logging, and tracing.

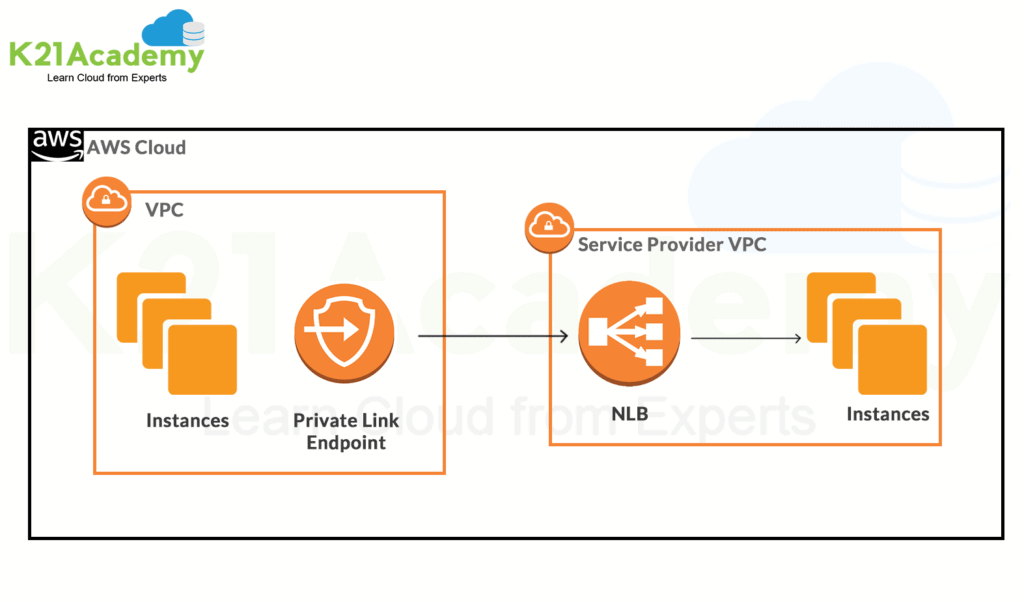
**Types of Load Balancers**

* Application Load Balancer
* Network Load Balancer
* Gateway Load Balancer
* Classic Load Balancer

**Check Out:**[What is AWS WAF](https://k21academy.com/amazon-web-services/aws-solutions-architect/aws-waf/)?

**AWS Private Link**

AWS Private Link provides private connectivity between various amazon services. The connectivity is done between Amazon VPC (Virtual Private Cloud), Amazon Web Services, and on-premises network without exposing traffic to the public internet.



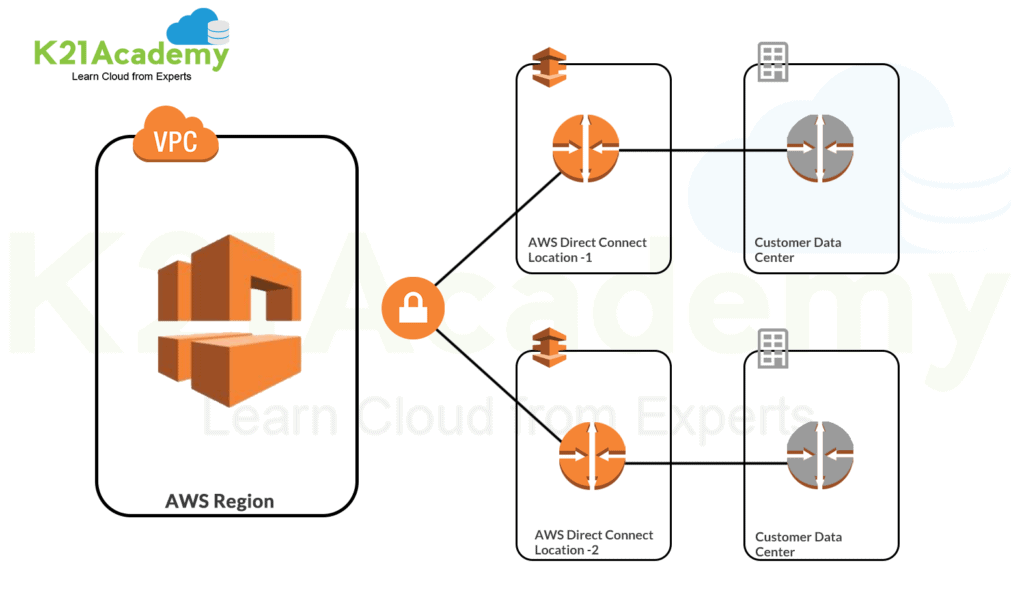
**Features of Private Link**

* **Secure Traffic** – It reduces exposure to various threats like brute force and distributed DOS attacks as it doesn’t traverse the public internet.
* **Simplicity** – It allows to connect with services across Amazon VPC without the need for firewall rules, path definitions, or route tables.

**Also Check:**[AWS Certified Solutions Architect Salary](https://k21academy.com/amazon-web-services/aws-solutions-architect/aws-solution-architect-salary/).

**AWS Direct Connect**

AWS Direct Connect helps in establishing a dedicated network from your premises to AWS. It enables a private and secure connection between AWS and the data center. It is compatible with AWS services and supports a high bandwidth for a more consistent network and better speed. The starting speed is around 50 Mbps and supports scaling up to 100 Gbps.



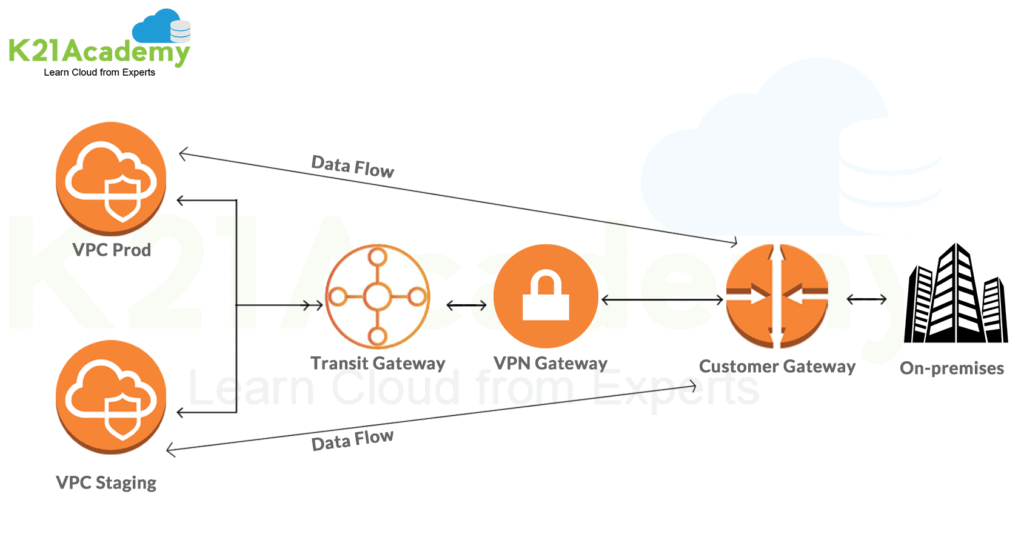
**Key Components of Direct Connect**

* **Connections** – Connections are created for the communication between AWS and on-premises.
* **Virtual Interfaces** – It enables access to AWS services. Public interface access public services like Amazon S3, whereas a private interface access private services like VPC (Virtual Private Cloud).

**Check Out:**Our blog post on [**AWS Well Architected Framework Pillars**](https://k21academy.com/amazon-web-services/aws-solutions-architect/5-pillars-of-aws-well-architected-framework/).

**Amazon Gateway**

Gateway is a service that acts as a gatekeeper for your network. All the request goes that is made from one end goes through a gateway to reach the other end. In simple words, it joins two networks for enabling communication between the devices present in them. For Amazon Networking, two common Amazon Gateways are used **API Gateway** and **Transit Gateway**.

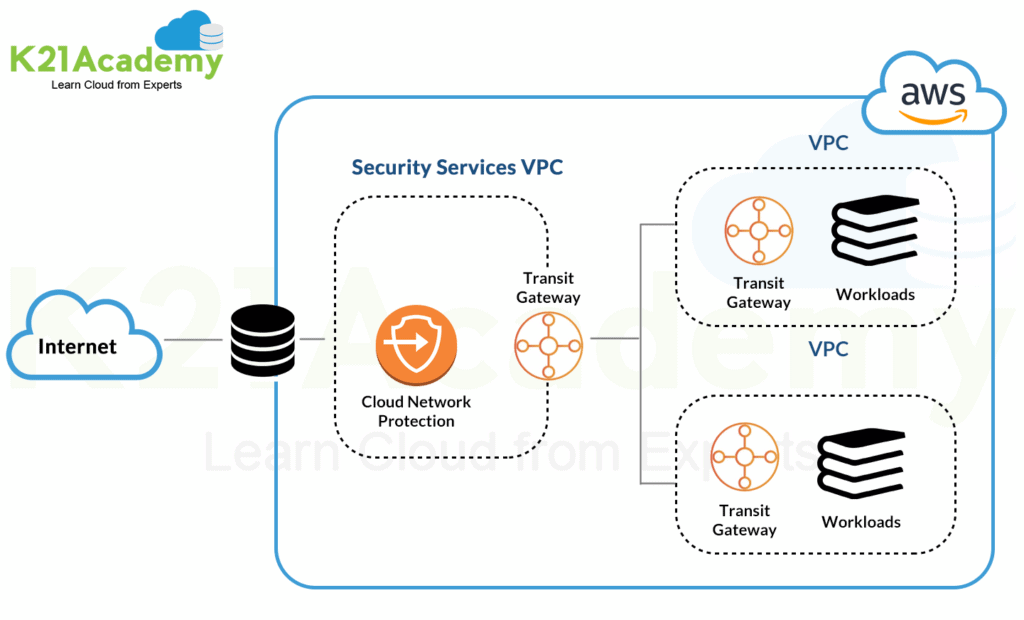


* **Transit Gateway** – Simplifies network and peering relationships by connecting VPC and on-premises networks through a central hub.
* **API Gateway –** It handles all the tasks to accept and process all the API calls. It helps developers to create, publish, monitor, and secure API at any scale.

**Check Out:**[AWS Solution Architect Interview Questions and Answers](https://k21academy.com/amazon-web-services/aws-solutions-architect/interview-questions-and-answers/).

**AWS Network Architecture**

AWS Network architecture is a representation or structured way for connecting various AWS services in a cloud. It serves the connectivity needs by creating a web of interconnected devices. Using the various AWS components and services, a fast and secured network is created.

The above diagram shows a sample of AWS Networking Architecture with Amazon VPC, where multiple components and services are connected to create a network.

Configuring complex environments for VPC requires deep understanding of networking. VPC is about communicating between servers efficiently, developing secure network policies, and keeping our nodes organized.

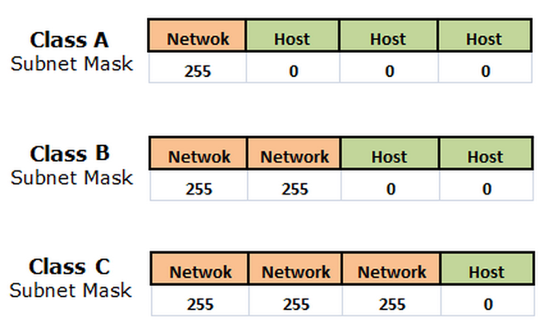
In this chapter, we will discuss the concepts that are involved with designing or interacting with networked computers such as subnets and CIDR notation for grouping IP addresses.

**Understanding CIDR and Subnets**

IP addresses

While IPv6 is becoming more common, in this article, we will be studying the remaining concepts using IPv4 addresses because it is much easier to discuss since its smaller address space.

IP addresses are made of two separate components. The first part of the address is used to identify the **network**, and the other past is used to specify a specific **host** within that network.



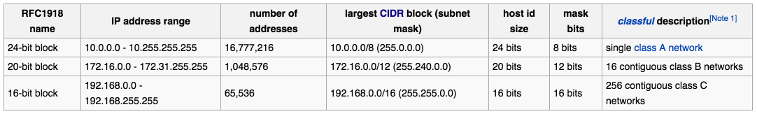
Reserved Private Ranges

There are also some portions of the IPv4 space that are reserved for specific uses.

One of the most useful reserved ranges is the **loopback** range specified by addresses from 127.0.0.0 to 127.255.255.255. This range is used by each host to test networking to itself. Typically, this is expressed by the first address in this range: **127.0.0.1**.

Each of the normal classes also have a range within them that is used to designate private network addresses.

1. **class A**  
   10.0.0.0 - 10.255.255.255
2. **class B**  
   172.16.0.0 - 172.31.255.255
3. **class C**  
   192.168.0.0 - 192.168.255.255



Picture from [Private network - wiki](https://en.wikipedia.org/wiki/Private_network)

Netmasks

**Subnetting** is the process of dividing a network into smaller network sections. This helps to isolate groups of hosts together.

As we discussed in the section ([IP addresses](https://www.bogotobogo.com/DevOps/AWS/aws-VPC-Virtual-Private-Cloud-1-netmast-subnet-default-gateway-CIDR.php#IP-addresses)), each address space is divided into a **network** portion and a **host** portion. For instance, for class C addresses, the first 3 octets are used to describe the network. For the address 192.168.0.12, the **192.168.0** portion describes the network and the **12** describes the host.

Each network, by default, has only one subnet, which contains all of the host addresses defined within. A **netmask** is basically a specification of the amount of address bits that are used for the network portion. A **subnet** mask is another netmask within used to further divide the network.

Each bit of the address that is considered significant for describing the network should be represented as a "1" in the netmask.

For instance, the address we discussed above, in a binary format, 192.168.0.12 looks like this:

1100 0000 - 1010 1000 - 0000 0000 - 0000 1100

The network portion for class C addresses is the first 3 octets, or the first 24 bits. Since these are the significant bits that we want to preserve, the **netmask** would be:

1111 1111 - 1111 1111 - 1111 1111 - 0000 0000

In the normal IPv4 format, this can be expressed as **255.255.255.0**. Any bit that is a "0" in the binary representation of the netmask is considered part of the **host** portion of the address.

We determine the **network** portion of the address by applying a bitwise AND operation to between the **address** and the **netmask**. A bitwise AND operation will basically save the networking portion of the address and discard the host portion. The result of this on our above example that represents our network is:

1100 0000 - 1010 1000 - 0000 0000 - 0000 1100 address

1111 1111 - 1111 1111 - 1111 1111 - 0000 0000 netmast

----------------------------------------------

1100 0000 - 1010 1000 - 0000 0000 - 0000 0000 network

This can be expressed as **192.168.0.0**. The host specification is then the difference between these original value and the host portion. In our case, the host is "0000 1100" or 12.

Subnets

Subnetting is to take a portion of the host space of an address, and use it as an additional networking specification to divide the address space again.

For instance, a netmask of 255.255.255.0 as we saw in previous section leaves us with 254 hosts in the network (Note that we cannot end in 0 or 255 because these are reserved). If we wanted to divide this into two subnetworks, we could use one bit of the conventional host portion of the address as the subnet mask.

For the sample (192.168.0.12), the networking portion is:

1100 0000 - 1010 1000 - 0000 0000

Host portion is:

0000 1100

We can use the first bit of our host to designate a sub-network. We can do this by adjusting the subnet mask from:

1111 1111 - 1111 1111 - 1111 1111 - 0000 0000

To :

1111 1111 - 1111 1111 - 1111 1111 - **1**000 0000

This netmask can be expressed as 192.168.0.128.

We designated the first bit of the last octet as significant in addressing the network! This effectively produced two subnetworks.

1. The first subnetwork is from 192.168.0.**1** to 192.168.0.127.
2. The second subnetwork contains the hosts 192.168.0.**129** to 192.168.0.255.

Note that the subnet itself must not be used as an address.

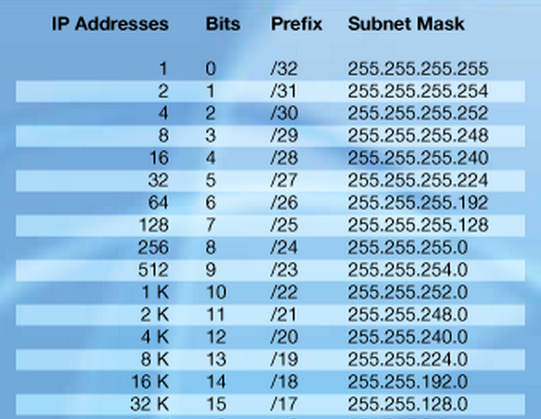
CIDR

**Classless Inter-Domain Routing (CIDR)** is a method for allocating IP addresses and routing Internet Protocol packets.

It is an IP addressing scheme that replaces the older system based on classes A, B, and C.

A single IP address can be used to designate many unique IP addresses with CIDR. A CIDR IP address looks like a normal IP address except that it ends with a slash followed by a number, called the IP network prefix.

CIDR addresses reduce the size of routing tables and make more IP addresses available within organizations.



To be more specific, let's use the example in the previous sections.

We can express the idea that the IP address 192.168.0.12 is associated with the netmask 255.255.255.0 by using the CIDR notation of 192.168.0.12/24. This means that the first 24 bits of the IP address given are considered significant for the network routing.

Default gateway

A **default gateway** in computer networking, is the node that is assumed to know how to forward packets on to other networks.

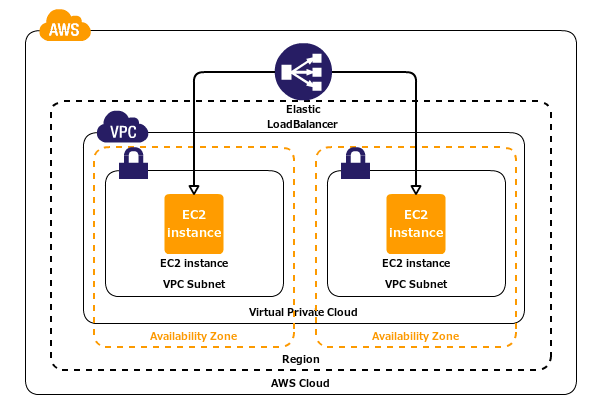
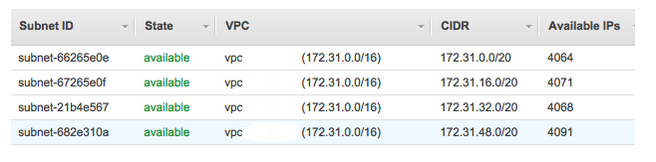
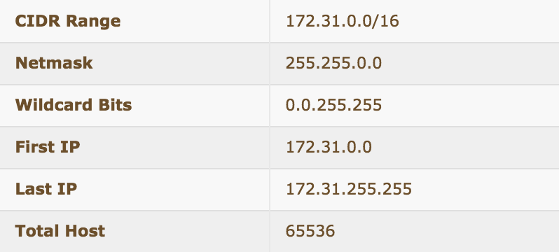
Typically, in a TCP/IP network, nodes such as servers, workstations and network devices each have a defined default route setting, (pointing to the default gateway), defining where to send packets for IP addresses for which they can determine specific route. The gateway is by definition a **router**.

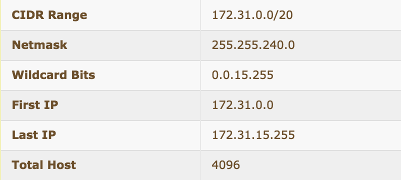
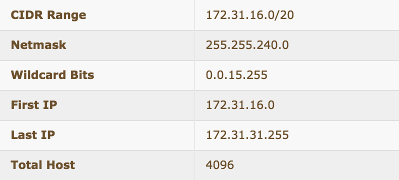
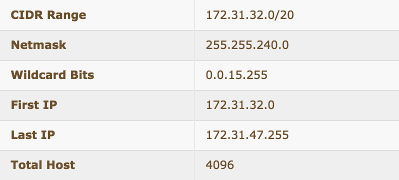
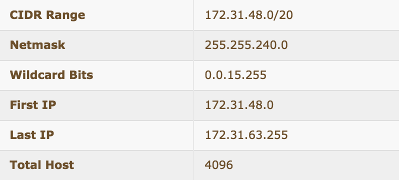
As a simple example, let's think about the following 2 networks :

1. Network: 192.168.1.0/30
2. Gateway: 192.168.1.1
3. Usable: 192.168.1.2
4. Broadcast: 192.168.1.3
5. Network: 192.168.1.0/29
6. Gateway: 192.168.1.1
7. Usable: 192.168.1.2-6
8. Broadcast: 192.168.1.7

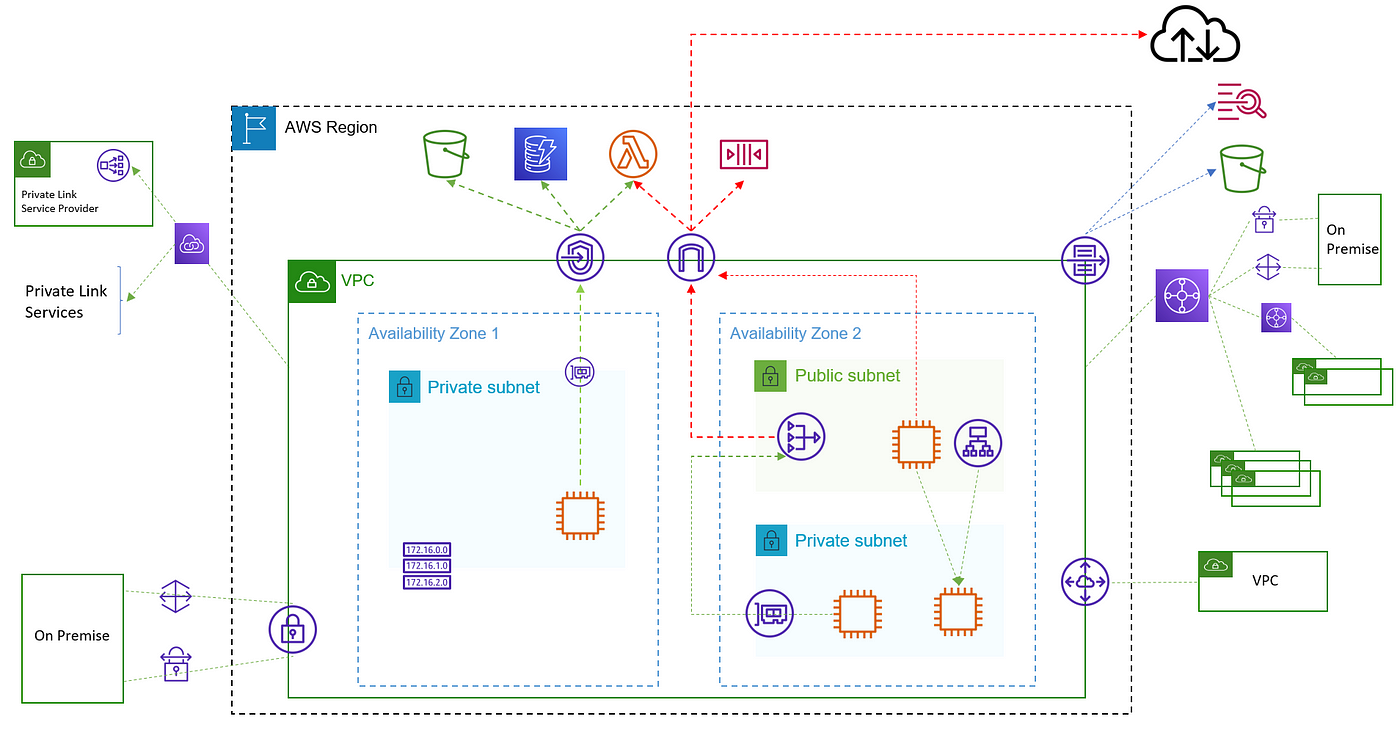
Note that **/29** may cover a range of 8 addresses, but only 5 of them can be used as host endpoints. **/30** has only 1 usable address.

CIDR sample from AWS VPC

1. 172.31.0.0/20  
     
   
2. 172.31.16.0/20  
     
   
3. 172.31.32.0/20  
     
   
4. 172.31.48.0/20  
     
   

**Understanding VPC:**



VPC components snapshot

The above diagram depicts the major components of the AWS VPC service. Let us go through them one by one.

## VPC

Virtual Private Cloud is a service from AWS which enables us to launch AWS resources in a logically isolated virtual network (similar to a local data center). It gives us complete control over our virtual networking environment, including resource placement (which AZ), connectivity(route table, peering), and security (Security Group & NACL). It is a regional service. By default, each region in our account will have a default VPC for us to quickly start.

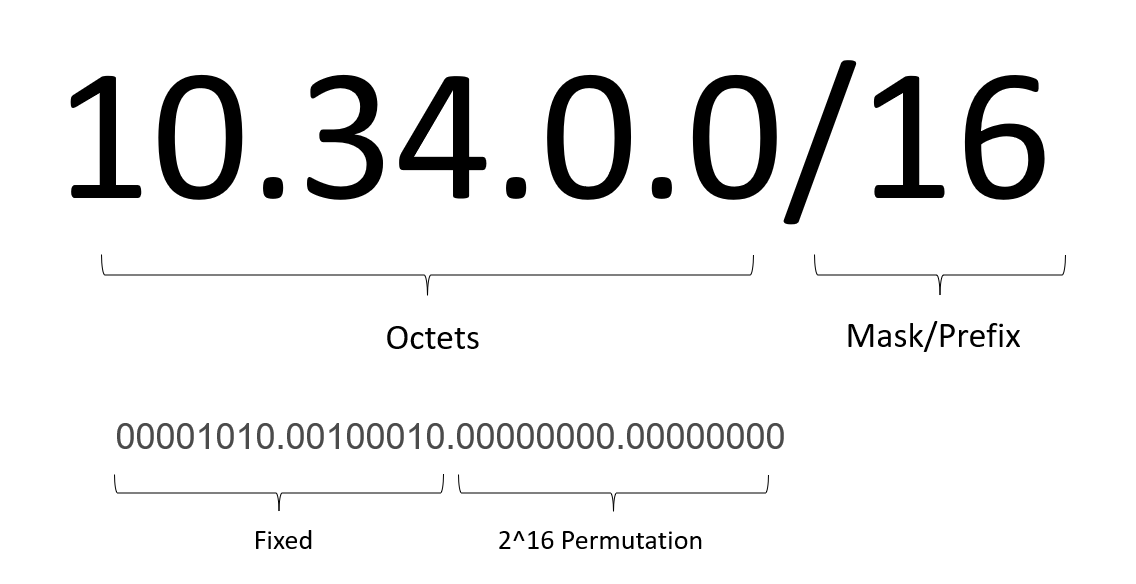
## [Amazon VPC quotas](https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html" \t "_blank)

### [The following tables list the quotas, formerly referred to as limits, for Amazon VPC resources per Region for your AWS…](https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html" \t "_blank)

[docs.aws.amazon.com](https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html" \t "_blank)

## CIDR

To create a VPC we need to supply a Classless Inter-Domain Routing (CIDR) block. Our VPC must be /16 or smaller. We need to be careful in selecting the right size of VPC as we can not modify the range of an existing CIDR block.



CIDR notation

An IPv4 address is represented as four 8-bit decimal numbers or octets separated by dots. Prefix number (Mask) is the number of bits locked from the left side. In the above example, the first 16 bits are fixed. The remaining (32 -16=16) bits’ permutations will decide the IPs in the subnet.

## Internet Gateway

The next thing we need once we have VPC is the ability to access the resources from local over the internet. Internet Gateway is the component that allows resources in the VPC to reach & be reachable over the internet. We create an IGW, then attach it to a VPC and add a route to the routing table (explanation later) pointing to IGW Id. IGW will do a 1-to-1 NAT for the instance address.

To internet: Response address is changed from private IP to instance's public/elastic IP.

From the internet: Target public IP is changed to private IP address of the instance.

## Subnet

A subnet is a range (subset) of IP addresses in your VPC. Each subnet is tied to one AZ.

**Public Subnet:** If a subnet is associated with a route table that has a route to an internet gateway, it’s known as a public subnet.

**Private Subnet:**If a subnet is associated with a route table that does not have a route to an internet gateway, it’s known as a private subnet.

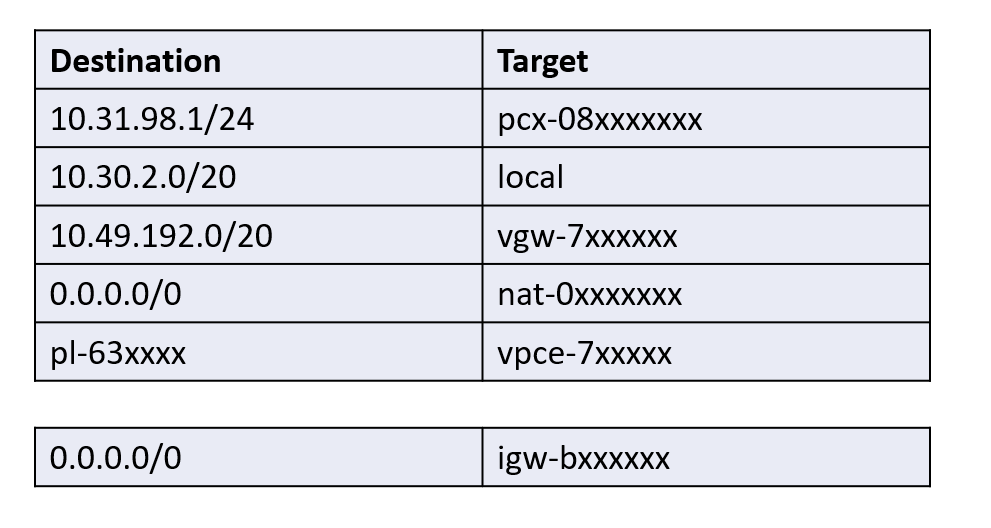
## [Visual Subnet Calculator](https://www.davidc.net/sites/default/subnets/subnets.html" \t "_blank)

### [Edit description](https://www.davidc.net/sites/default/subnets/subnets.html" \t "_blank)

[www.davidc.net](https://www.davidc.net/sites/default/subnets/subnets.html" \t "_blank)

## Route table

Every resource created in a subnet gets a private IP address from the CIDR of the subnet. And instances communicate with each other using this IP. How to reach a particular IP is defined in the subnets associated route table. The table has the following structure. A destination and the target for the destination.



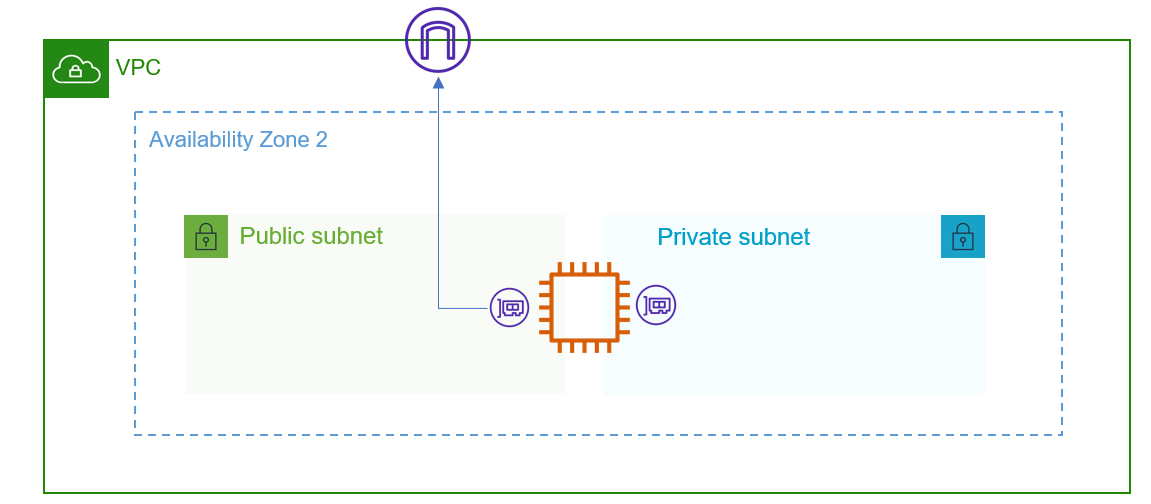
Route table

If our route table has multiple routes, the most specific route that matches the traffic (longest prefix match) will be applied. Let us say we want to reach 10.1.1.24 and in the route table, we have two routes, one for 10.1.1.0/24 and the other 10.1.0.0/16, then it will use the first one.

## Elastic Network Interface

An elastic network interface is a logical networking component in a VPC that represents a virtual network card. Security groups are tied to these interfaces. We can attach multiple interfaces to an instance. Also, each ENI can have multiple private IPs. In EKS we use multiple ENIs and multiple IPs to launch multiple pods on the same EC2 with different IPs.

Another use case is a Dual homed EC2.



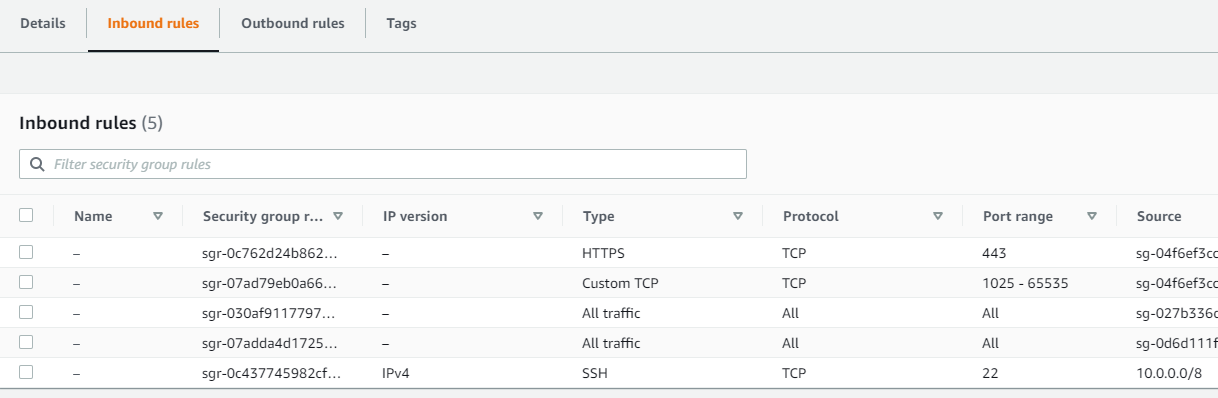
Dual home EC2

## Elastic IP Address

An Elastic IP is a static public IP address that we can attach to our AWS Account. These IPs are fixed, unlike public IPs which change on instance termination. We can allocate an IP from AWS managed IP or bring our own IP to AWS.

## Security Group

A security group acts as a firewall to control inbound and outbound traffic on the ENI. Security groups are stateful. This means when in initiate a request the response traffic for that request is allowed to flow in regardless of inbound security group rules & vice versa. By default, it allows all outbound traffic and denies all inbound traffic.



Sample Security Group

## Network Access Control List

NACLs act as a firewall at the subnet level. A custom NACL by default denies all inbound and outbound traffic. Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa). Each subnet can only be attached to one NACL.

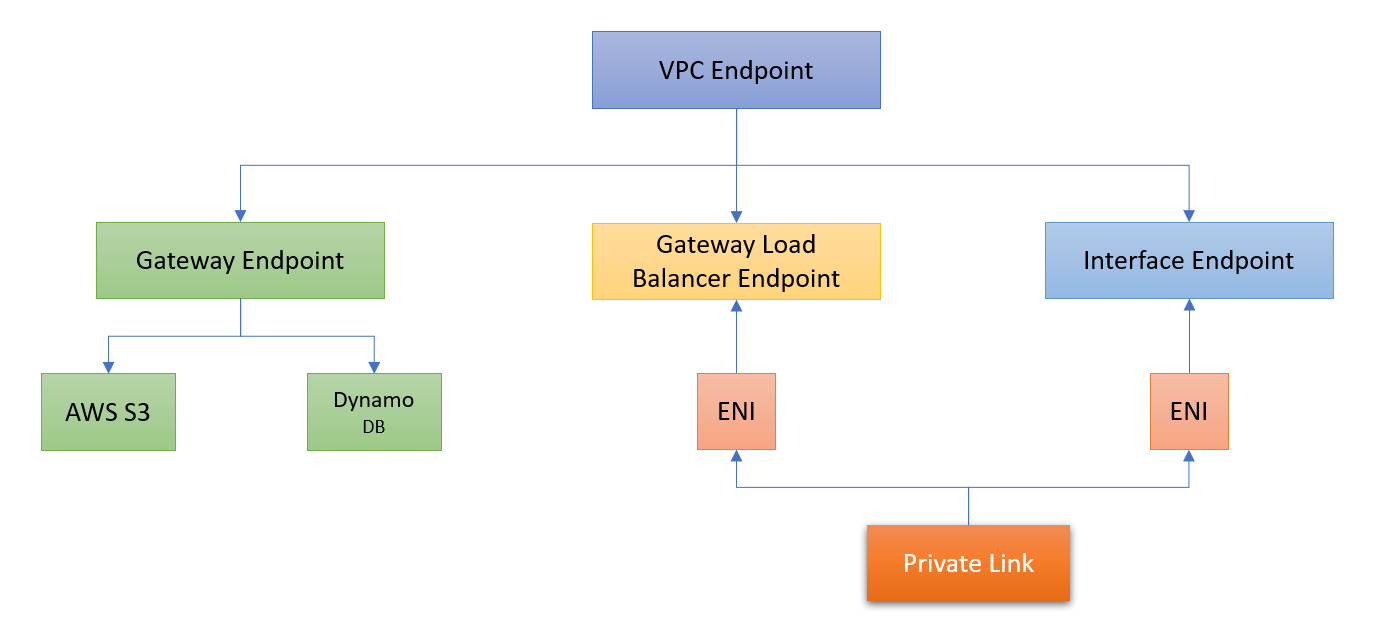
## NAT Gateway

We saw that resources in the private subnet cant be reached from the internet and they also cant reach the internet. But we need access to the internet for various use cases like downloading software. To enable the host from the private subnet to reach the internet we use NAT. NAT device enables instances in a private subnet to connect to the internet but prevents hosts on the internet from initiating connections with the instances. NAT does the Address translation from private IP to its own public IP. Must be created in a public subnet for accessing internet addresses.

*5–45 Gbps Bandwidth  
55,000 connections per minute for destination IP & port combination*

## VPC Endpoint

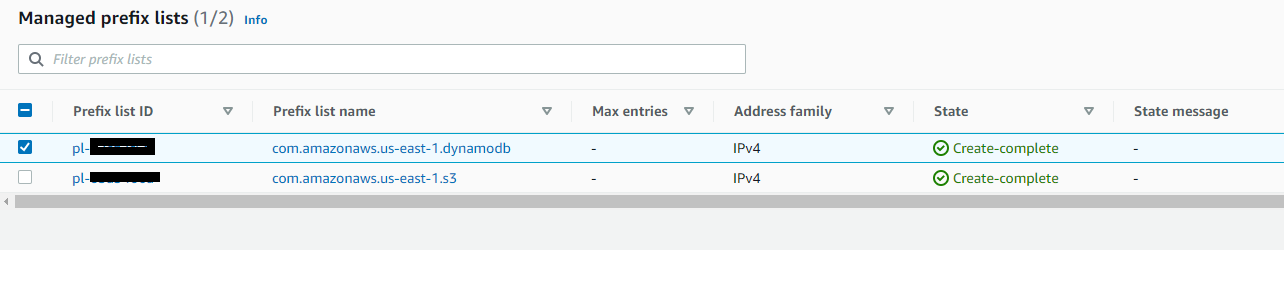
To access AWS services that are not in a VPC like S3, DynamoDb, SNS, and SQS we need to traverse the internet. To reduce the traffic & security AWS provides a service called VPC Endpoint. VPC endpoints are virtual devices. What it does is, creates a prefix-list route in the private subnet’s route table. The endpoint gets a private IP from the selected subnet. Since the IPs are private we don't need to use NAT or IGW. This way the traffic never leaves the AWS network. These endpoints are local to the region and scoped to one VPC. It also supports endpoint policies, using this we can control what can go through these endpoints.



## Prefix List

A Managed Prefix List is a collection of CIDR blocks created from addresses we frequently use. We can use prefix lists to make it easier to configure and maintain our security groups and route tables.

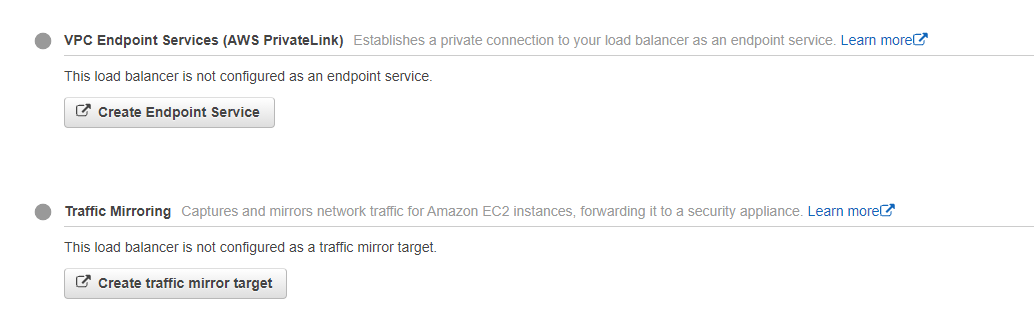
*Customer Managed Prefix List  
AWS Managed Prefix List*

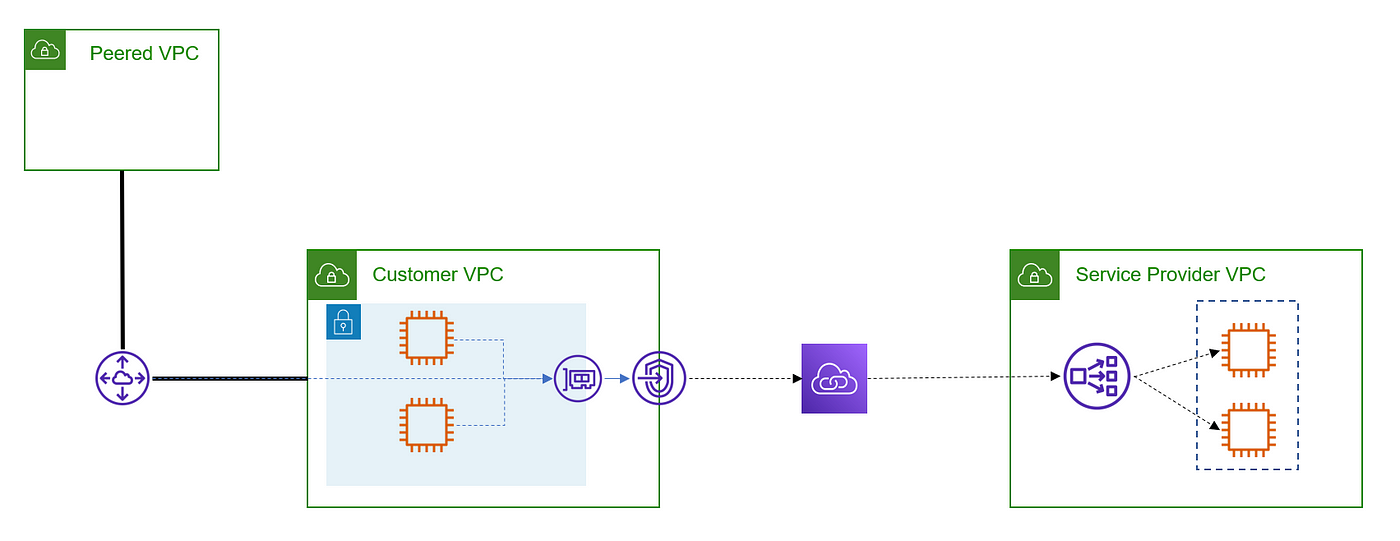


Prefix list

## Private Link

AWS PrivateLink is a highly available, scalable technology that enables you to privately connect your VPC to supported AWS services, services hosted by other AWS accounts (VPC endpoint services), and supported AWS Marketplace partner services. It requires an NLB and can be configured from the NLB page.





## DHCP Options

The Dynamic Host Configuration Protocol (DHCP) is a standard way of passing configuration information hosts on a TCP/IP network. We can create our own DHCP options for our VPC. We can change the following configuration according to our needs:

*DNS Name  
DNS Servers  
NTP Servers*

Settings that can be configured:

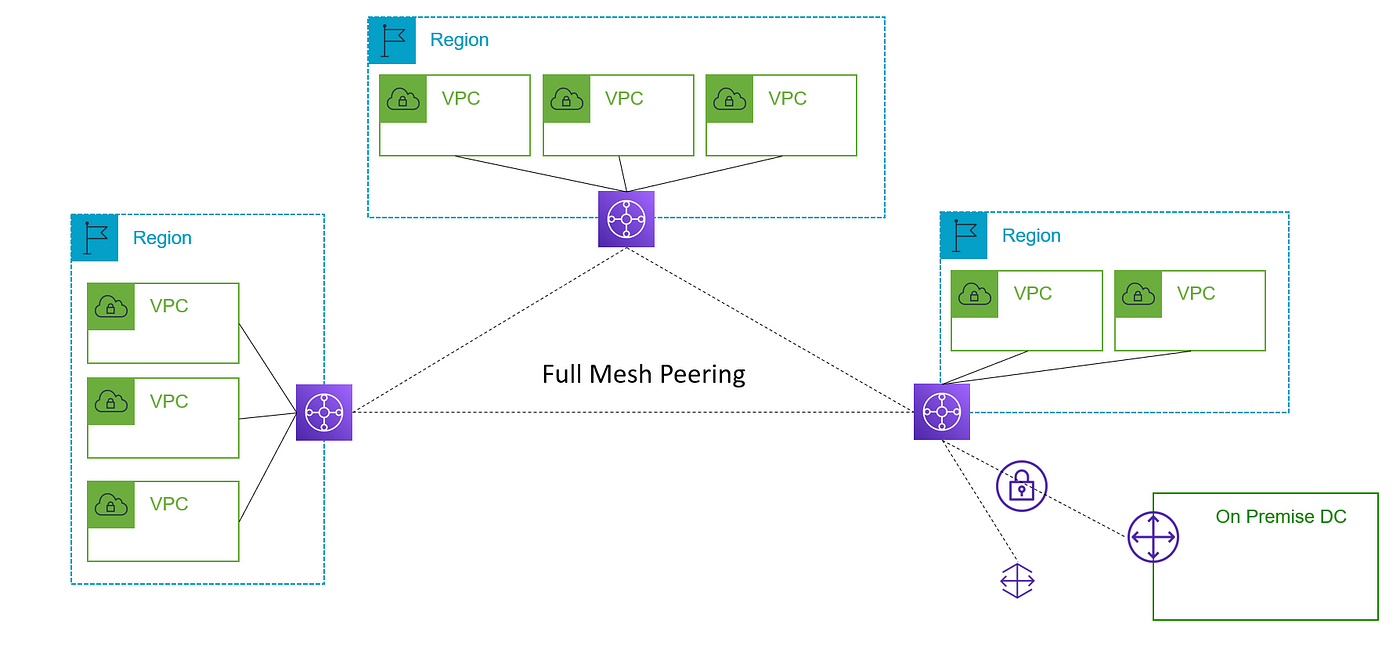
enableDnsSupport — Amazon-provided DNS server at 169.254.169.253  
enableDnsHostname — Instances VPC will get public DNS hostnames

## VPC Peering

A VPC peering connection allows two non-overlapping VPCs to behave as if they are one network. Instances in either VPC can communicate with each other using their private address. Peering connection is created using was infrastructure itself. VPC peering needs to be done in a full mesh way as it does not support edge routing and transitive peering. For an account with N VPCs, we will need (N\*(N-1))/2 peering connections will be needed. On top of this, each VPC can have a max of 125 peering connections.

## Transit Gateway

AWS Transit Gateway acts as a cloud router & simplifies network architecture by removing the need to have complex point-to-point VPC peering. Traffic between VPC and Transit Gateway remains on the AWS private network. Transit Gateway operates on Layer 3. A transit gateway has three major components — **1. Attachment 2. Association 3. Propagation**

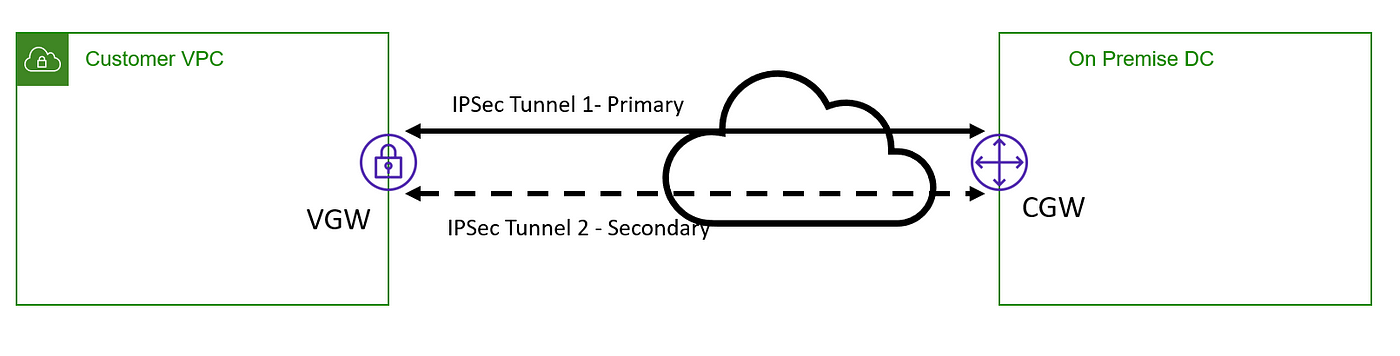


Unlike Private Link, TGW bothers about overlapping CIDRs. If we have a requirement of unidirectional traffic from one VPC to other VPC we can leverage Private Link, Route53 Resolver to achieve a cost-efficient solution.

[Route53 + Transit Gateway](https://aws.amazon.com/blogs/networking-and-content-delivery/centralized-dns-management-of-hybrid-cloud-with-amazon-route-53-and-aws-transit-gateway/)  
[PrivateLink + Route 53 Resolver](https://aws.amazon.com/blogs/networking-and-content-delivery/integrating-aws-transit-gateway-with-aws-privatelink-and-amazon-route-53-resolver/)

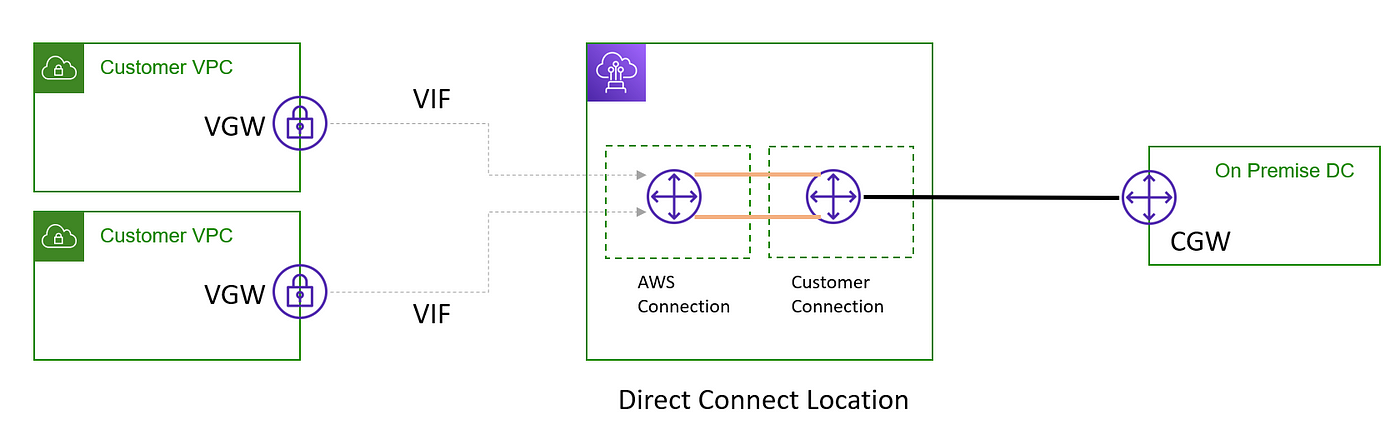
## VPN

AWS VPN establishes secure connections between on-premises networks, remote offices, client devices, and the AWS VPC. We can extend our on-premises networks to the cloud and securely access them from anywhere. On the AWS side of the VPN connection, a Virtual Private Gateway is created. This creates two VPN tunnels (with the public IP addresses). On the on-premise side, a customer gateway (just a representation of the customer’s router) is provisioned. In the end, the route table is updated to use the “vgw” as the target for the on-premise IPs.



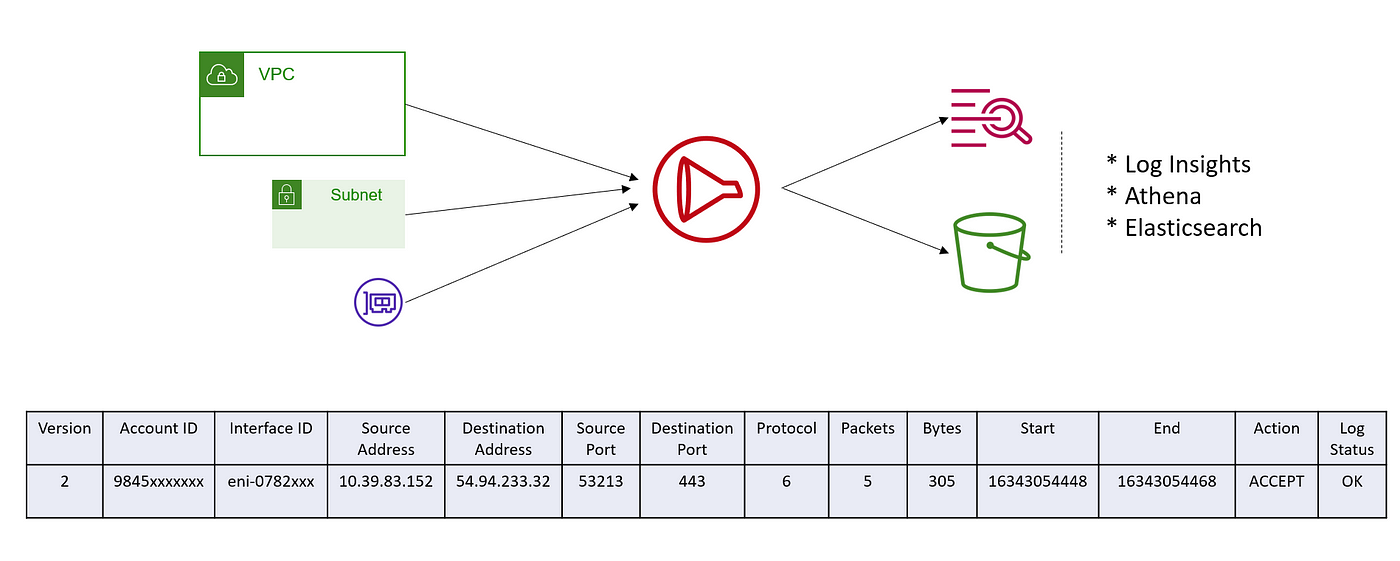
## Direct Connect

Direct Connect establishes private connectivity between AWS and our data center, office, or collocated environment. Direct Connect combination of virtual private gateways & private virtual interfaces. Each VIF is a single VLAN connection between AWS and our Service provider network. We can also provision Public VIF so that services like S3, DynamoDbs can be accessed over a secure & fast network from our data centers.



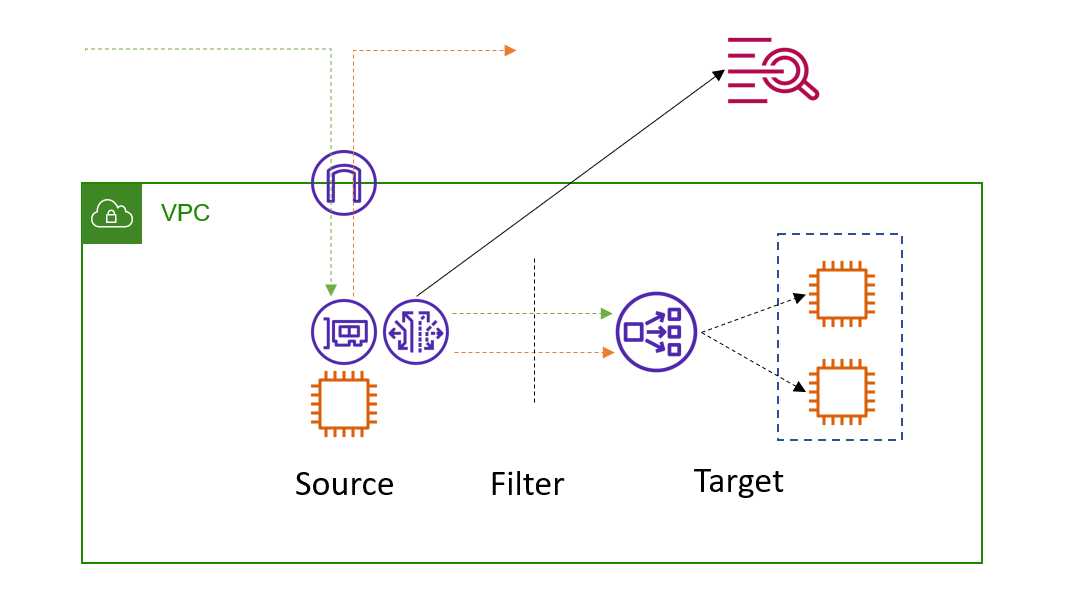
## VPC Flow Log

VPC Flow adds the observability aspect to VPC. This feature enables us to capture information about the IP traffic going to and from network interfaces in our VPC. Aggregation is done at 1 min intervals. So it is not real-time. Since the log is collected outside of the network traffic path, so there is no added latency.



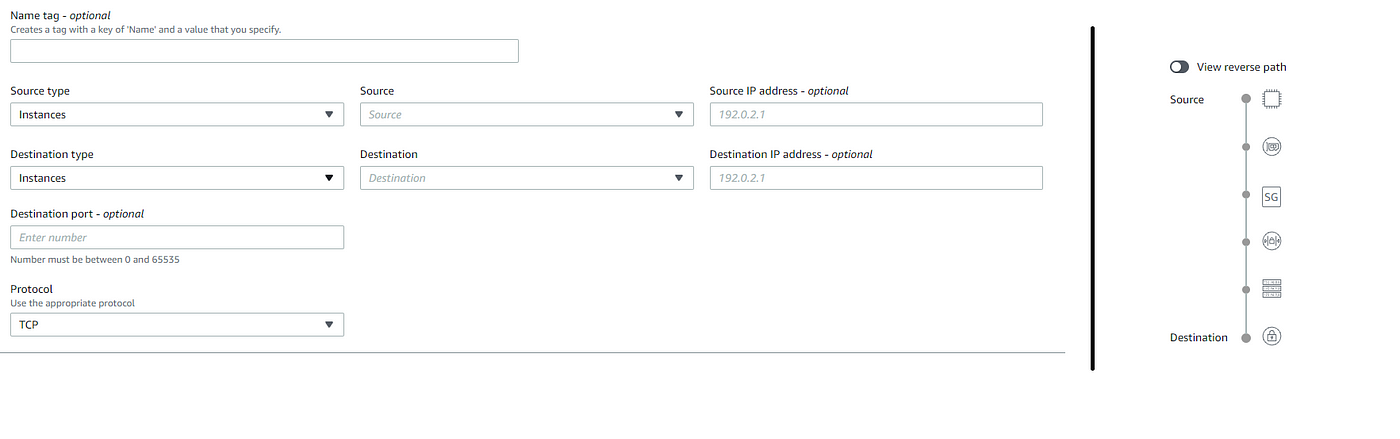
## Traffic Mirroring

Traffic Mirroring is a feature that we can use to copy network traffic from an ENI of EC2 instances. The copied traffic can then be filtered and forwarded to out-of-band security and monitoring appliances for content inspection, threat monitoring & troubleshooting, etc.



## Reachability Analyzer

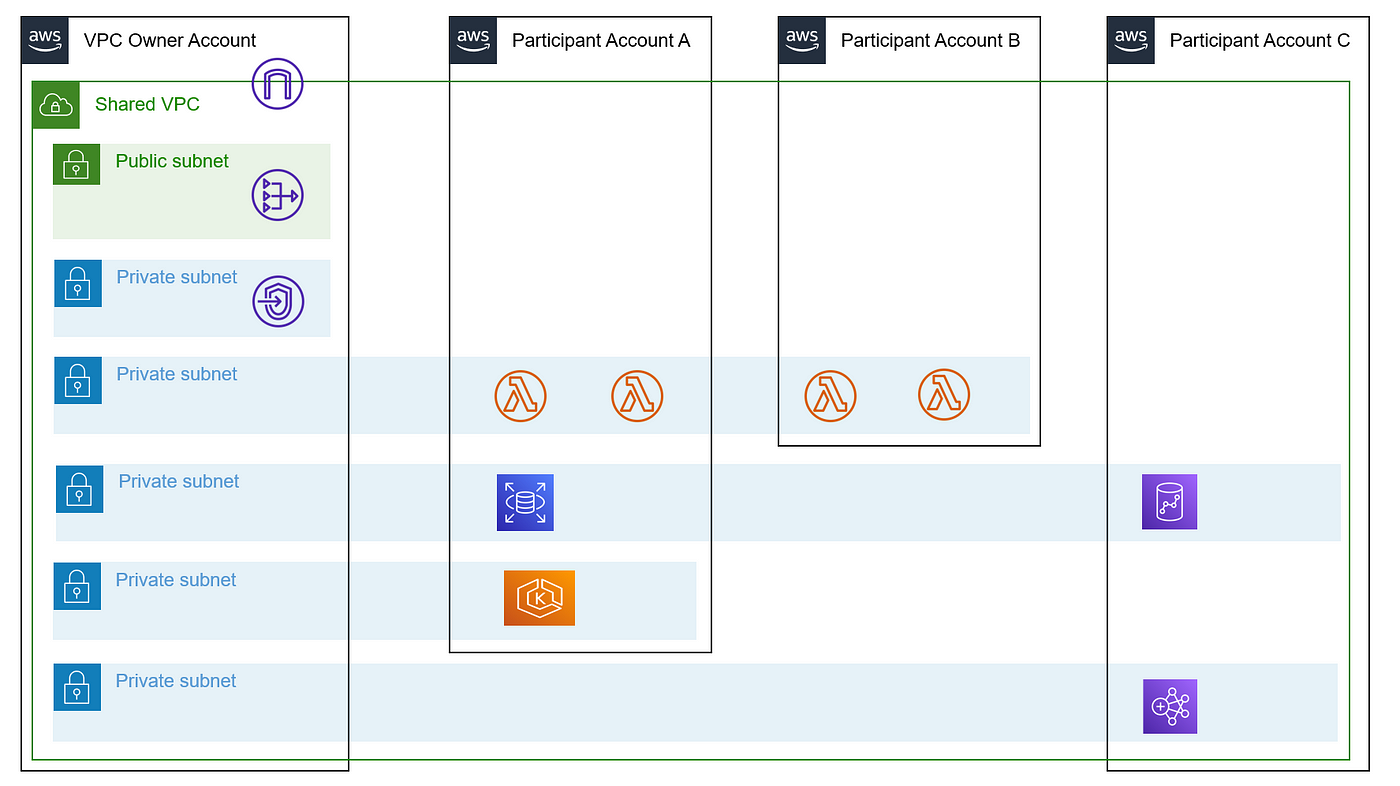
This is a tool to test connectivity between a source and destination in our VPC. It lists hop-by-hop details of the network path between the source and the destination. In case of failures, it identifies the blocking component.



Bonus.

## Shared VPC design

A shared VPC design allows us to have one VPC shared across multiple accounts. This can be useful in centralizing networking work in one account. In the below image we have a VPC shared with three accounts. The public subnet is in the owner account and only private subnets are shared. Any modification to the endpoint, NAT, IGW, or NACL will apply to all accounts which are sharing those subnets.



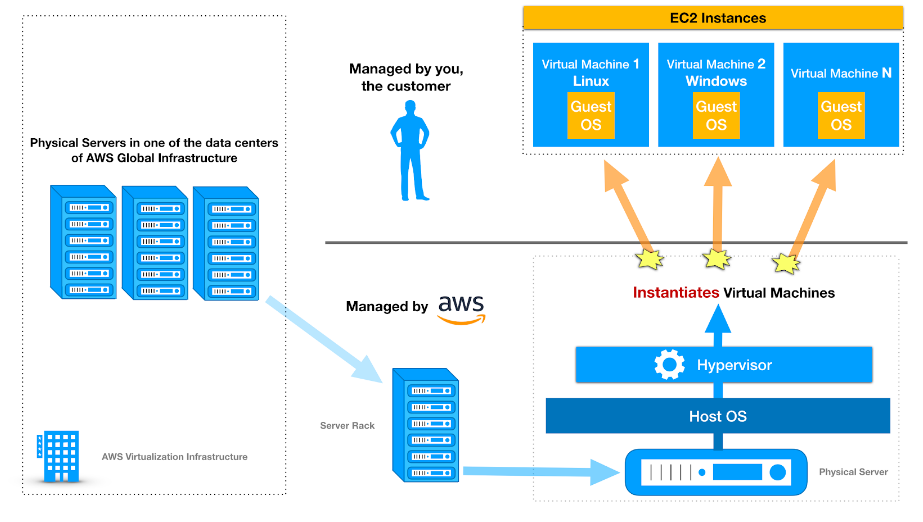
Shared VPC design.

**AWS Compute:**

**AWS** provides a variety of cost-effective and flexible **computing services** to meet the needs of your organization such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), Amazon Elastic Container Service for Kubernetes (EKS), Amazon Lightsail, AWS Batch, and AWS Lambda to name a few. For some services like Amazon EC2, you have extensive control of the underlying resources while for others, AWS has full control.

With these computing services in AWS, you can dynamically provision a number of resources and pay only the computing resources you actually consume. This significantly reduces the upfront capital investment required and replaces it with lower variable costs. Instead of the traditional long-term contracts or up-front commitments, you can opt to pay your compute resources in AWS using an On-Demand or Spot pricing option to easily discontinue your cloud resources if you don’t need them, effectively reducing your operating expenses. Amazon EC2 is a commonly used AWS service which you can integrate with various features and services like Amazon Machine Image, Instance Store, Elastic Block Store, Elastic Network Interface, Elastic IP, Auto Scaling, Elastic Load Balancer, Placements Groups, Enhanced Networking, Security Groups and so much more.

Have you ever heard people say “Amazon Linux EC2 **Instance**” instead of “Amazon Linux EC2 **Server**” when they launch a compute resource in AWS? It is because AWS is programmatically creating a new virtual machine (VM) **instance**, rather than providing you with an actual physical **server**, when you launch an EC2 Instance. AWS has a powerful virtualization infrastructure that is composed of physical servers that they manage. Each physical server has a host operating system that runs a virtual machine monitor (VMM), also known as a hypervisor, which instantiates multiple VM “instances” that you can use. These instances use guest operating systems that you can manage.



AWS manages, operates, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates. Conversely, the customer is responsible for the management of the guest operating system such as installing patches and doing the necessary security configuration.You can also use these compute services in AWS to run your High Performance Computing (HPC) applications. Basically, HPC requires a higher storage I/O and large amounts of memory to perform a complex task. Moving your HPC workloads to AWS eliminates the unnecessary wait times and long job queues that are associated with limited on-premises HPC resources. Since there are no upfront capital expenditures or lengthy procurement cycles, you can get significant cost savings whenever you process time-flexible, stateless workloads.

* [**Amazon EKS**](https://tutorialsdojo.com/amazon-elastic-kubernetes-service-eks/)
* [**Amazon Elastic Compute Cloud (EC2)**](https://tutorialsdojo.com/amazon-elastic-compute-cloud-amazon-ec2/)
* [**Amazon Elastic Container Registry (ECR)**](https://tutorialsdojo.com/amazon-elastic-container-registry-amazon-ecr/)
* [**Amazon Elastic Container Service (ECS)**](https://tutorialsdojo.com/amazon-elastic-container-service-amazon-ecs/)
* [**Amazon Elastic Kubernetes Service (EKS)**](https://tutorialsdojo.com/amazon-elastic-kubernetes-service-eks/)
* [**Amazon Lightsail**](https://tutorialsdojo.com/amazon-lightsail/)
* [**AWS Batch**](https://tutorialsdojo.com/aws-batch/)
* [**AWS Elastic Beanstalk**](https://tutorialsdojo.com/aws-elastic-beanstalk/)
* [**AWS Fargat**](https://tutorialsdojo.com/aws-fargate/)
* [**AWS Lambda**](https://tutorialsdojo.com/aws-lambda/)
* [**AWS ParallelCluster**](https://tutorialsdojo.com/aws-parallelcluster/)
* [**AWS Savings Plan**](https://tutorialsdojo.com/aws-savings-plan/)
* [**AWS Serverless Application Model (SAM)**](https://tutorialsdojo.com/aws-serverless-application-model-sam/)
* [**AWS Serverless Application Repository**](https://tutorialsdojo.com/aws-serverless-application-repository/)
* [**AWS Wavelength**](https://tutorialsdojo.com/aws-wavelength/)