**Chapter 5. AWS Network Services**

**A NOTE FOR EARLY RELEASE READERS**

With Early Release ebooks, you get books in their earliest form—the authors’ raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 9th chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *mpotter@oreilly.com*.

People interact with multiple applications (or with each other) via the internet. We touched upon concepts around ‘Communication Networks and Protocols’ in Chapter 6—this chapter is an extension of Chapter 6 that will introduce you to AWS networking services such as Amazon VPC, Amazon Route 53, Amazon Elastic Load Balancer (ELB), Amazon API Gateway. This is intended to help you set up networking infrastructure on AWS cloud and explore the networking concepts related to different services and how to establish connectivity between them.

To make systems more reliable, safe, and to ensure they operate as per their own business requirements, companies used to prefer setting up their own infrastructure in on-premise data centers, but that comes with a high infrastructure management cost. If you are just launching your start-up idea, it will likely be too costly to set up a personal data center. Cloud provides cost benefits (analyzing the cost of resources and optimizing them to full potential) along with a lot of flexibility. For example, It is much harder to sell back a physical server that you bought for setting up your personal data center than it is to turn off an Amazon EC2 machine (server) in AWS cloud. The decision of whether to go with a Cloud provider based solution (and if yes, then which Cloud provider?) or with an on-premise data center will heavily depend on the business requirements.

AWS operates on a [shared responsibility model](https://docs.aws.amazon.com/whitepapers/latest/aws-risk-and-compliance/shared-responsibility-model.html), meaning the customer and AWS both work together to make best use of services,in a secure and cost effective way where AWS is responsible for “security of the cloud” and customers are responsible for “security in the cloud”. Before we deep dive into multiple AWS Networking Components, let’s dig a little into where these AWS services are located and how they are segregated per customer.

**Getting Started With AWS**

To start with AWS Cloud, customers need to [create an AWS account](https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/). An AWS Account is a fundamental entity in Amazon Web Services that provides access to a wide range of cloud services. This account will hold all the information related to your AWS resource (such as compute instances, storage, networking etc) creation, management, operations, support, billing etc. With an AWS account, you can provision and configure resources, monitor usage and costs, set security and access controls, and interact with various AWS services. It will be very important to identify how many AWS accounts should be set up. This will vary from use case to use case and scale the company operates on—below are some general suggestions to consider when deciding what number of AWS accounts one should configure:

* Complete separation of applications, meaning every application is launched in a new AWS account. This option presents benefits of full separation and effective cost management of applications, but presents challenges in the way of too much operational cost.
* Separation of AWS accounts based on business type—for example, a cab booking company operates in domains like cab availability, payments, analytics etc. All applications related to a single business will operate in a single account, which also provides the advantages of correlated applications located near to each other optimizing latency and offloading requirements of resource setup for service connection between different accounts. These applications can be segregated in Amazon VPCs or AZs depending on the use-case.
* Separation of AWS accounts based on software domains—for example, having separate accounts for networking, monitoring, storage, and security and auditing. This kind of setup provides the benefits of easier operations management.For example, all the networking configurations are present in a single account that can be managed by the Network engineering team.

AWS recommends creating a multi account set up to ensure there is clear division of responsibility and keeping future scaling of systems in mind. For example, separate networking accounts will help to keep all networking configurations at a single place and with access only to the networking team. AWS offers a service called [Landing Zone](https://docs.aws.amazon.com/prescriptive-guidance/latest/migration-aws-environment/welcome.html), which provides a baseline to get started with multi-account architecture, identity and access management, governance, data security, network design, and logging. AWS Landing Zone can be orchestrated by customers themselves or as a managed service. One such service, [AWS Control Tower](https://aws.amazon.com/controltower/), can be used to set up initial prescriptive configuration and further customizations can be made as organizations scale.

**AWS Regions**

Once the AWS account is created, you’re ready to launch the first server in the cloud. A very intriguing question though, where is this cloud? AWS Cloud operates from a physical location which might be near or far from you. AWS defines these geographically distributed locations around the world where AWS operates data centers as [AWS regions](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/) - for example the US North Virginia region is called us-east-1. Each region is a separate and independent geographic area, isolated from other regions. You can choose the AWS region to launch resources per your business requirements, such as latency constraints or compliance regulations, or for AWS service availability in the region. For example Netflix architecture has presence in three AWS regions, ensuring high availability even in cases of a regional failure. Keeping resources at multiple locations helps to build resilient systems, which we touched upon in Part-I of this book. Further, there are some global services such as [Amazon S3](https://aws.amazon.com/s3/) or AWS Identity and Access Management (AWS [IAM](https://aws.amazon.com/iam/)) service where region selection is not a requirement. You can check the complete list of regions available across the globe [here](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/).

**NOTE**

There is a possibility that not all AWS services are present in your AWS region.

**AWS Availability Zones**

Customers choose to deploy resources in multiple regions to improve availability, latency, and many other business use-cases. A key point to note is not everyone can afford to replicate resources at multiple AWS regions. Does that mean they compromise application availability then? Cloud providers work on a shared responsibility model which means that in order to ensure a good run on cloud, customers and cloud providers have to work together. A simple example could be deploying your servers in [multiple availability zones](https://aws.amazon.com/builders-library/minimizing-correlated-failures-in-distributed-systems/) instead of just one. That way, even if one AZ goes down, there are servers operational in other AZs to serve the traffic. You can think of a single region as a cluster of data centers and each individual (or combination of) data center is an availability zone. An [availability zone](https://aws.amazon.com/builders-library/static-stability-using-availability-zones/) is physically isolated from other availability zones by meaningful distance with independent power, cooling and fast private fiber-optic low-latency network connectivity. The purpose of AZs is to provide fault tolerance, resilience, and high availability by allowing you to distribute your applications and data across multiple AZs within a region. AZ are classified with suffix to region names—for example, us-east-1a or us-east-1b are AZs within the us-east-1 region.

**AWS Local Zones**

Consider the following scenario: you are all set to launch your startup for the people of New Delhi, India. The application setup requires single digit p99.99 latency for API operations, but the nearest AWS region is Mumbai, India. The operations are optimized well and your use-case doesn’t allow you to use pre-cached data such as utilizing Content Delivery Network or retrieving data from servers located at specific AZ. How can you overcome such an issue? This can potentially be solved by placement of database instances near to customers and this can be achieved via AWS Local Zones. Local Zones help to set up infrastructure near to customers, which is connected to AWS regions via fast paced network.

AWS Local Zones are an extension of an AWS Region and are designed to bring AWS services closer to specific geographic areas with low latency requirements. Local Zones are geographically separate from their parent region and are located in metropolitan areas. They provide a subset of AWS services and are primarily intended for latency-sensitive workloads that require proximity to end-users or specific on-premises resources.

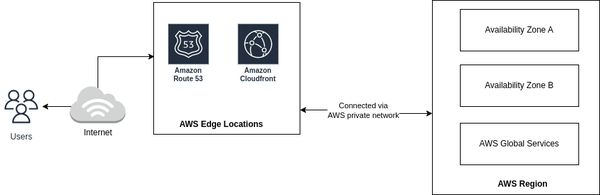
You can check all available locations for Local Zones [here](https://aws.amazon.com/about-aws/global-infrastructure/localzones/locations/) and list of AWS services supported [here](https://aws.amazon.com/about-aws/global-infrastructure/localzones/features/).

**AWS Edge Locations**

Consider that HBO is planning to stream a new season of Game of Thrones and the first episode will be out on upcoming Sunday morning 8 AM. GOT is popular worldwide and it is expected to be watched in multiple countries at the same time. How can HBO ensure the best customer experience via full HD video quality with no video buffering? The AWS Content Delivery Network Service called [Amazon CloudFront](https://aws.amazon.com/cloudfront/) helps to place the content near to users at locations referred as Edge Locations so they can be served in minimum time possible.

AWS Edge Locations are points of presence (PoPs) distributed globally to bring AWS services closer to end-users. Edge Locations help improve the performance of content delivery by acting as caching and content delivery endpoints for Amazon Cloudfront, thus reducing latency and improving data transfer speeds.

You can think of AWS Edge locations as data centers which are connected with AWS regions to support fast upload and download of data. Some other services like Amazon Route 53, an Amazon DNS service use the same setup for faster resolution of DNS queries. [Figure 5-1](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_1_connectivity_to_aws_region_via_edge_location) shows connectivity of users via AWS edge locations to AWS regions. We’ll explore more about Amazon Route 53 and Amazon CloudFront towards the end of this chapter.



**Figure 5-1. Connectivity to AWS region via Edge Location**

We gathered a very high level of view of how AWS Cloud is set up and made accessible to customers across the globe. Let’s dig right into how connectivity can be established with AWS Cloud and look at different networking components within it.

**Introduction to AWS Networking Services**

[Figure 5-2](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_2_ip_address_classes) from the last section described how users are establishing connections to AWS services located within the AWS region. It’s important to understand how this accessibility is maintained and how AWS Cloud provides a similar abstraction to one’s private data center. This section will introduce you to multiple offerings provided by AWS in the context of networking and connectivity.

**Amazon VPC**

Setting up your own data center could be costly both in terms of monetary costs and operations management. Amazon VPC, short for Virtual Private Cloud, provides a similar level of infrastructure separation as you’d have in a personal private data center. We can think of Amazon VPC as your personal data center located inside AWS Cloud.

Amazon Virtual Private Cloud (VPC) is a service provided by AWS that allows you to create a virtual network in the cloud. It provides you with control over your network environment, including IP address range selection, creation of subnets, configuration of route tables, and network gateways. Amazon VPC enables you to securely launch resources like Amazon EC2 instances, Amazon RDS, and Amazon Elastic Load Balancers within a logically isolated section of the AWS cloud.

**NOTE**

This book will not outline the infrastructure setup steps but will focus more on a deep dive into AWS core concepts and will provide links to AWS documentation for the most updated guidance on setup.

AWS Cloud is utilized by a lot of customers and Amazon VPC plays a vital role in setting up boundaries between the customer resources. AWS accounts come with one default VPC to launch resources and further more VPCs can be created as per business use-case requirements. There is a default limit of 5 VPCs per account which can be increased by raising a query with AWS Support Center. Let’s dig into Amazon VPC in more details, starting with some basic networking knowledge helpful in making better decisions on Amazon VPC setup.

**IP Addresses**

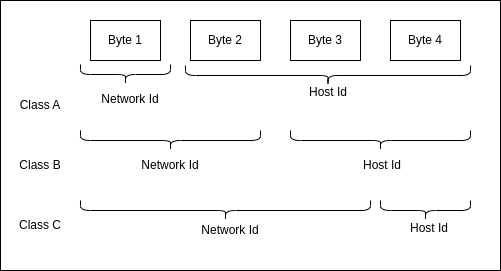
Let’s consider a real-world example. You’re very excited to meet your new friend in person and both of you agreed to meet at Cafe Delhi Heights, 3rd Floor, 301 & 302, Ambience Mall, Gurugram, India. However, you need a specific location, more specifically a table number in this restaurant and not just the restaurant location. This pin-pointed location in the networking world is called an IP address—a unique string of characters that identifies each computer. Every device should have an IP address to connect with another device on the internet. IP addresses can be of type IPv4 or IPv6. Let’s discuss each of these in a bit more detail.

**IPv4**

IPv4 is a 32 bit or 4 byte address space where each byte is represented via decimal numbers (binary octet) and separated by dot (.), for example 192.168.1.0. How can we identify to which destination the traffic should be routed to? To establish a similar analogy, how do you think the Postal Service delivers mail at your doorstep or you know to meet your new friend at Table Number 21 at Cafe Delhi Heights? There are two components required in general for unique identification. For physical mail, it would be your zip code/postalcode/area PIN code (depending on your country) and your house number, and for meeting your new friend, it would be the restaurant location in Ambience Mall, Gurugram and then the table number in this restaurant.

In similar fashion, to deliver network packets to your personal computer, there are two components involved—a network component (network ID) and a host component (host ID). For example, your office network will map to a network ID and your personal computer will map to a host ID.

The division of the number of bits that should be allocated to network Id and host Id is defined via IP classes as shown in [Figure 5-2](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_2_ip_address_classes). There are in total 5 IP classes from A-E, out of which Class D is reserved for multi-tasking and Class E is reserved for research purposes.



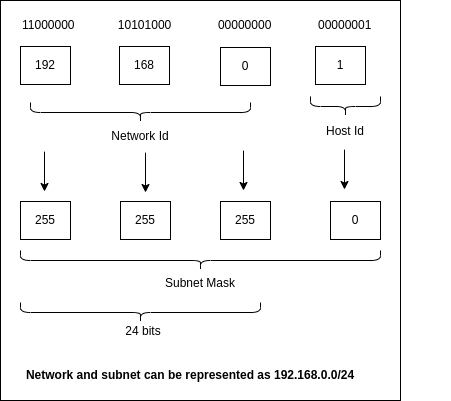
**Figure 5-2. IP address classes**

One difference between Class A, B, and C is the number of bits assigned to each network Id and host Id. Another is the range of IP addresses that are allowed in each class, as represented in Table 9-1.

|  |  |  |
| --- | --- | --- |
| IP Address Class | From Range | To Range |
| Class A | 1.0.0.0 | 127.255.255.255 |
| Class B | 128.0.0.0 | 191.255.255.255 |
| Class C | 192.0.0.0 | 223.255.255.255 |
| Table 5-1. Class-wise IPv4 Address range | | |

Deciding which class suits your specific business use-case depends on the requirement of the number of networks and the number of hosts in a network. For example, Class A provides 126 network IDs and 16,777,214 host Ids, whereas Class C provides 2,097,152 network Ids and 254 host IDs. The division of IP address or network space into network and host address is achieved via Subnet Masks.

As the name suggests, subnets help in dividing the parent network into sub-networks. Subnet mask is the division of IP address into network and host address. It is a 32 bit number where host bits are set to 0 and network bits are set to 1. Amazon VPC logically isolates the resources at regional level and further division into AZ can be achieved via subnets—we’ll be exploring this while creating our first Amazon VPC in follow up sections. [Figure 5-3](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_3_representation_of_ipv4_address_and_subnet_mask) shows an example IPv4 address along with subnet mask representation.



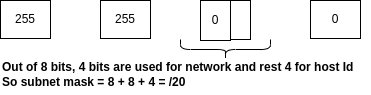
**Figure 5-3. Representation of IPv4 address and subnet mask**

**NOTE**

Keep in mind that 0 and 255 are reserved for special purposes and therefore can’t be assigned to hosts. Hence the number of host Ids in Class A is 126 and not 128.

A specific problem that arises with IPv4 addresses is the very limited number of addresses available in comparison to the number of devices and networks across the world. The intermittent solution to slow down exhaustion of IPv4 addresses is via Classless Inter-Domain Routing (or CIDR).

CIDR is a way of representing an IP address and its subnet mask. The classless concept was introduced in 1993 to slow down the exhaustion of IPv4 addresses. CIDR helps in varying subnet mask length, skipping standard division via classes. For example, you can create either /16 or /24 subnet masks but not /20. This kind of division helps in optimizing the class space of IP addresses. You can see this represented in [Figure 5-4](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_4_variable_length_subnet_mask).

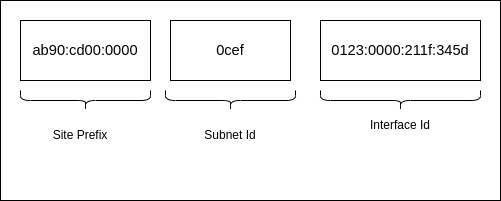


**Figure 5-4. Variable length Subnet mask**

 However, the long term solution is migration to IPv6 addresses.

**IPv6**

IPv6 is 128 bits which helps in much larger network space as compared to 32 bits IPv4 address. IPv6 is represented by 8 groups of 4 hexadecimal digits and separated by colons. For example, ab90:cd00:0000:0cef:0123:0000:211f:345d. [Figure 5-5](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_5_simple_ipv6_address_representation) describes simple IPv6 notation where site prefix represents public topology allocated by an Internet Service Provider(ISP) or Regional Internet Registry, subnet Id represents private topology which is internal to specific network, and interface Id represents a unique device identifier which is configured via interface’s MAC address by using IEEE’s Extended Unique Identifier (EUI-64) format.



**Figure 5-5. Simple IPv6 address representation**

Both IPv4 and IPv6 addresses can be further divided into public and private IP addresses based on their visibility. The concept of public-private is more used for IPv4 addresses considering the limited availability of addresses. We’ll be using examples of IPv4 addresses to walk through it.

**Private and Public IP Addresses**

Let’s refer back to our earlier real-life example. You agreed to meet your friend at a reserved table number 21 at Cafe Delhi Heights and enjoyed the positive vibe and food of the restaurant. You got curious and decided to visit the kitchen where this delicious food was prepared, but the restaurant owner denied access to the kitchen and mentioned only specific individuals can go to the kitchen. Here the kitchen is private space and the general sitting area is public space. Certainly there are ways to get to the private space (aka the kitchen in this scenario) and we’ll explore this in follow up sections.

Consider the example of your work from home setup—you’ve a router for Wi-Fi connection and all the devices (such as mobile phone, office laptop, personal laptop, etc.) connect with this router to access the internet. If you run a Google search on all the devices for, “What’s my IP?”, you’ll get the same IP address. This IP address is called a public IP address and is assigned by the ISP. Restarting the router might lead to a different IP address on all the devices, but it will be the same across the devices.

All these devices can connect with each other without going over the internet via Private IP address. A Private IP address is assigned by the router to each device connected to it. Table 9-2 shows the range of addresses allowed for Private IP—the rest all can be used for public network space.

|  |  |
| --- | --- |
| Class | IP Range |
| Class A | 10.0.0.0 – 10.255.255.255 |
| Class B | 172.16.0.0 – 172.31.255.255 |
| Class C | 192.168.0.0 – 192.168.255.255 |
| Table 5-2. Private IPv4 address space per class | |

The IP addresses are assigned dynamically from the provided IP address range on AWS resource creation. In case there is a requirement to associate specific IP addresses to a resource, we can utilize Elastic IP addresses.

**Elastic IP Address**

Elastic IP address is a static and public IPv4 address associated with an AWS account which can be assigned to an Amazon EC2 instance or network interface without changing on any state change. This can be considered to be used for scenarios with requirements of static IP address without being changed over time. For example, avoid Amazon EC2 instance IP address change if a new instance is spawned up to replace an unhealthy instance.

Please note that there is an additional cost associated with Elastic IP addresses. It is chargeable and bought for a specific region, IP addresses of one region are not accessible in other regions. Now, let’s move forward on key considerations for the creation of Amazon VPC.

**Considerations for Amazon VPC Creation**

The networking ideas we gathered in previous sections will help us to set up our first VPC. The VPC wizard asks for IPv4 CIDR block, IPv6 CIDR block, and tenancy. The IPv4 CIDR block is a required input that must be added to proceed further. The VPC can work in dual mode, operating with both IPv4 and IPv6, and the allowed CIDR block for IPv4 is between /16 to /28 and for IPv6, it is fixed to /64.

You’ll need to consider the following points about CIDR block:

* VPC creation requires an initial IPv4 CIDR block, although you might have a use-case to just use IPv6 address space. AWS VPC supports both IPv4 and IPv6 address space in a single VPC and it can be specified during creation.
* The CIDR block should be specified from the RFC 1918 range, and it is recommended to use a private IP address range, though public can also be chosen.
* The size of the CIDR block can’t be changed once created. Customers should carefully plan the size of CIDR considering future needs to avoid hurdles or rework in the future. In short, bigger CIDR blocks can be chosen if you’re not able to gather concrete details to have more flexibility in the future.
* The CIDR block should not overlap with existing CIDR blocks associated with VPCs. This is also essential for utilizing services such as [VPC Peering](https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html) and [AWS Direct Connect](https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html), the CIDR blocks should not overlap across the VPCs.
* You can associate up to 5 additional CIDR blocks( this is a soft limit and can be adjustable up to 50) to a VPC, the additional CIDR block’s range should be strictly smaller than the primary CIDR block.

**NOTE**

Soft Limit means the limit is adjustable and can be increased by following AWS Service Quotas or with help of AWS Support team.

Next we need to consider tenancy. Let’s go back to our real-world example. There are two ways of booking Cafe Delhi Heights for meeting your new friend. The first option is to book the entire restaurant, as this is a top secret meeting and you don’t want any other people to be available in your booked time. This is called dedicated tenancy.

The second option is to reserve a table for two—other available tables can be booked by other people which is called shared tenancy. It’s logical to go with option two considering the cost unless there are specific reasons such as a secret meeting. The same concept is applicable when we request servers from AWS. The instances can either run on shared hardware (other people’s instances can run on the same hardware via virtualization) or on dedicated hardware where your instances are separated from other customer’s resources, depending on your business needs. This option can be selected at the time of Amazon VPC creation and all the instances launched in VPC are created with the same option by default, unless overridden.

**NOTE**

AWS provides multiple tools for creation of resources such as AWS CLI, AWS CDK or via AWS Console. As a beginner, you can start experimenting via AWS Console but as systems scale, we recommend maintaining a code repository to provision any AWS resources, popularly known as Infrastructure as Code(IaC). This could help on multiple fronts like replicating the same resources in another region or maintaining infrastructure audit.

The next step in VPC configuration is setting up subnets. Subnets help in separation of resources across multiple availability zones. Let’s dig into those.

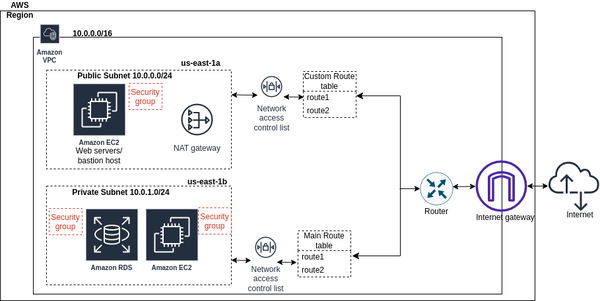
**Subnets**

As mentioned earlier in the chapter, subnet means a sub network, logical subdivision of IP Address range inside the VPC CIDR block. Why should one divide network space into multiple sub-networks? The general idea of separation of resources per availability zone is to ensure high application availability. Each subnet is associated with an availability zone (AZ) within an AWS region.and by associating resources to the specific subnets, we ensure resources are launched into that specific AZ. One interesting fact around subnet mapping to AZ is it could vary from customer to customer—the us-east-1a for you can be us-east-1b for your friend. AWS abstracts out these details from customers to ensure uniform allocation (or as per availability) of resources.

Each subnet should be assigned a CIDR block similar to VPC—the CIDR block to be assigned can be the same as the VPC CIDR block or subset of it. The key point is that the CIDR block across the subnets inside your VPC should not overlap. The first four and the last IP address in each subnet CIDR block are reserved and can’t be assigned to resources. The subnet can be a public subnet or private subnet depending on its connectivity to the internet. Let’s dig a little deeper into that.

**Public Subnet vs Private Subnet**

Public and private subnets look similar, the difference lies in their ability to connect to the internet. For example, a web application’s servers can be placed in a public subnet so that it is accessible to the general public, and the database servers can be placed in a private subnet to limit accessibility from the general public, see [Figure 5-6](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_6_overview_of_different_networking_components) for reference. You were denied access to take a look at Cafe Delhi Heights kitchen as it’s located in a private space (or private subnet) though you can freely roam in a general sitting area, as this is public space or public subnet.



**Figure 5-6. Overview of different networking components**

The resources in the public subnet can access the internet due to the presence of a direct route to an [Internet Gateway](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html), whereas resources in the private subnet require a Network Address Translation(NAT) service or a NAT instance to access the internet—more details are captured in the [NAT gateway](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html) and Internet gateway section.

The key point is that the direct route to the internet gateway is the only differentiating factor between public and private subnets. A subnet with resources having a public IPv4 address but no direct route to the internet gateway is referred to as a private subnet.

When setting up your VPCs and subnets inside your AWS account, consider these following best practices:

* Use multiple subnets: Create multiple subnets within different availability zones to achieve fault tolerance and high availability.
* Isolate resources: Use separate subnets for different types of resources to improve security and network segmentation.
* Public and private subnets: Place resources with public access in public subnets and sensitive resources in private subnets.

**Internet Connectivity**

As we mentioned earlier, VPC provides a logically isolated network in AWS Cloud, but how does this logically isolated section connect with the internet? Or how can internet traffic access the resources located in the subnets? The routing of traffic and securing of resources is achieved via components such as route tables, internet gateway, security groups, etc which we’ll be focusing on in the following sections.

**Route Tables**

In the AWS Cloud, you just need to configure required routes in the route table and don’t have to worry about setting up a router. A route table is a collection of rules, referred to as routes, that determine where network traffic from your subnet or gateway is directed. We can assume AWS internally maintains a router to facilitate this routing.

Every subnet should be associated with a route table, and the main route table is created implicitly to provide private access among the subnets. You should create more custom route tables as required and assign them to subnets—note that the same route table can be associated with multiple subnets. The key consideration here should be to create a route table for clear division of routing responsibility—this helps in better maintainability and readability of routes by not overcrowding a single route table.

Let’s consider the two kinds of route tables:

*Main Route Table*

The Main Route Table, which automatically comes with a VPC, by default contains the first entry for local routing in the VPC, which helps resources in different subnets to establish connectivity with each other. Every subnet creation leads to implicit association to the main route table. If required, customers can create a custom route table and explicitly associate it to a subnet.

There are a couple of things you need to consider about the Main Route Table:

* It can’t be deleted and the custom route table can’t be set as the main route table, though it can be replaced with a custom route table for subnet association.
* You can add additional routes to the main table—our recommendation would be to avoid this and create custom route tables for adding custom routes.

*Custom Route Table*

Custom Route Tables don’t contain any routes by default and should be updated as per network traffic routing requirements. Custom route tables can be deleted unlike the main route table though there should be no subnet associations for this operation to be successful.

Now that you understand the two types of route tables and their purpose, let’s look at how these routes are added and what they look like. There are two important inputs in the route table - Destination and Target.

* Destination is added as CIDR, basically a range of IP addresses that specifies where the network packets should go, for example, 10.0.0.0/16.
* Target specifies how the network packets will reach the destination, such as gateway, network interface or a connection—for example internet gateway(igw) for routing traffic to internet or VPC peering connection. An example Route Table is shown in Table 9-3.

|  |  |
| --- | --- |
| Destination | Target |
| VPC CIDR | Local |
| 0.0.0.0/0 | igw-id |
| Table 5-3. Route Table with internet connectivity via Internet Gateway | |

Route Tables direct network traffic in and out of a subnet but it doesn’t apply any security filters on this traffic. AWS provides software firewalls, [Security Groups](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html)(SGs) and [Network Access Control Lists](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html)(NACLs) to implement traffic filters which are useful in controlling the network traffic permissions. Both of these components help to control the traffic that can flow in and out of VPC.

**Security Groups**

Let’s understand how SGs can help to control the traffic we want to essentially filter out or disallow any unwanted traffic. SGs are created at the VPC level and assigned at an instance level, controlling inbound and outbound traffic at an instance level based on protocols, ports and IP addresses. Your EC2 instance can have one or more SGs. There will always be one SG associated to an instance, and, if not created, a default SG will be associated, which is created at time of VPC creation.

To secure and limit access for incoming and outgoing traffic to instances, you need to add inbound and outbound rules to your SG. Inbound rules define traffic that is allowed to the instance and outbound defines traffic that is allowed from the instance.

Consider the following facts about SGs:

* SGs are stateful. For example, you fire one request from your personal laptop to an EC2 instance to get some data and your IP address is added as part of Inbound rules. You’ll get a response back even if there are no outbound rules included for your personal laptop. In short, rules allowed in one direction will automatically be allowed in the opposite direction, there is no requirement for explicitly adding them.
* You can’t delete the default SG. Our recommendation would be to create a custom SG as needed. Multiple SGs can be associated to an instance, but you should create SGs keeping future scale in mind. For example, you should avoid duplicate rules SG and reuse wherever it’s possible.
* You should only add required access and not overexpose the resources. For example, for SSH connection, only allow Port 22 for a set of IP addresses and not the entire internet.
* SG doesn’t provide an option to define explicit deny rules—all the rules which don’t match the allowed rules condition are implicitly denied access to resources.

For each rule added as part of inbound or outbound rules, you’ll need to specify the following  parameters as shown in Table 9-4:

*Type*

It represents the type of traffic. On the basis of chosen ‘type’ value, AWS determines Protocol and Port range automatically. There is also a custom type to add custom values,for example for port range.

*Source or Destination*

Source attribute is added for inbound rules and Destination attribute is added for the outbound rules. Here, you can add specific IP addresses, complete CIDR blocks or other security groups as well.

*Description*

It is helpful to identify why a certain rule is added or what purpose it solves. This is an optional field, but we recommend adding a description for easy future references.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Protocol | Port Range | Destination | Description |
| SSH | TCP | 22 | 117.212.92.68 | Test SSH rule |
| Table 5-4. Example SG Outbound Rule | | | | |

**Network Access Control Lists**

NACLs are stateless packet filters which are attached at the subnet level, unlike SGs, thus controlling inbound and outbound traffic at the subnet level. NACLs provide the ability to add, allow, or deny rules for outbound or inbound traffic at the subnet level. You may think of NACLs as an additional layer of security on top of SGs which ensure to block the traffic if SGs are too flexible.

If you think back to our example of the Cafe Delhi Heights kitchen, consider that there are individual chefs in the kitchen—you may be allowed to enter the kitchen, but still the chef may refuse to talk to you.

There are a few key things that you need to remember about NACLs:

* NACLs are stateless. Customers need to add explicit rules for both inbound and outbound traffic to allow or deny actions.
* VPC comes with a default NACL, which will be attached to all the subnets inside VPC and it allows all inbound/outbound traffic. For fine grain traffic control, you can create a custom NACL or modify rules in an existing one—our recommendation would be to create custom NACLs as needed.
* NACLs define inbound or outbound rules to allow or deny actions. The rules will be evaluated in sequence and once a particular rule succeeds, all the rules in sequence below will be skipped. If none of the rule succeeds, then final rule marked as ‘\*’ will evaluate it as a deny action.

For each inbound or outbound rule, you’ll need to specific the following parameters as shown in Table 9-5:

*Rule Number*

Rule numbers a sequence of numbers in which rules are evaluated which can be numbered from 1 to 32766.

*Type*

Type represents type of traffic. Based on chosen type, AWS pre-populates protocol and port range. Custom protocol and port range can also be added as per selected type.

*Source*

Source represents inbound rules added as a CIDR block.

*Destination*

Destination represents outbound rules added as a CIDR block.

*Allow or Deny*

For every rule, an explicit action should be added as ‘Allow’ or ‘Deny’. This action determines if traffic is allowed or denied.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule Number | Type | Protocol | Port Range | Source | Allow/Deny |
| 100 | All traffic | All | All | 0.0.0.0/0 | Allow |
| \* | All traffic | All | All | 0.0.0.0/0 | Deny |
| Table 5-5. Example NACL Inbound Rule | | | | | |

Route Tables, SGs and NACLs help to configure the routes and configure network security. To enable the connectivity as per the configured routes, AWS offers Internet Gateway or NAT Gateway which we’ll be exploring in the next section.

**Amazon VPC To Internet Connectivity**

We described VPC as a personal data center in AWS Cloud, but how will the users establish connectivity with this personal data center and how will this personal data center connect to the internet? In this section, we’ll dig into different AWS components which helps to resolve these connectivity hurdles.

**Internet Gateway**

In the section ‘Route Tables’ Table 9-3, we showed you a route as Destination ‘0.0.0.0/0’ and Target as ‘igw-id’ for allowing internet access to resources. The destination for this route refers to the entire internet and the target is an internet gateway identifier.

Internet Gateway (IGW) is a horizontally scalable and highly available AWS managed VPC software component that provides connection between your VPC and the internet. IGW is managed by AWS and is a highly available, redundant and horizontally scalable application.

Internet Gateway is attached to VPC and can span across the subnets in different AZs.

In the context of our Cafe Delhi Heights restaurant, IGW represents the front door to get in or get out of the restaurant. Much like you can only go to the public sitting area in the restaurant and can’t go inside the kitchens, the same applies to the IGW—it only helps to connect to resources in public subnets.

Here are a few key points about IGW:

* IGW helps in establishing connectivity in both the directions, from internet to VPC and vice versa by using public IP addresses.
* IGW provides NAT support for instances with public IPv4 addresses in the subnet. For traffic leaving for the internet from instance, IGW makes sure the reply to request is sent back to public IPv4 address, and for traffic destined for resource public IPv4 address, IGW ensures translation to instance’s private IP address before the traffic is delivered to VPC.
* Internet Gateways are used in conjunction with route tables to determine the path of network traffic. A route table associated with the VPC directs traffic destined for the internet to the Internet Gateway. It acts as the default gateway for outbound traffic and routes it to the appropriate destination.
* Internet Gateways enable VPC resources to communicate with other AWS services, such as Amazon S3 or Amazon DynamoDB, over the internet.
* Internet Gateways are stateless, which means they don’t maintain any information about the state of network connections. Each packet is evaluated independently based on routing rules and security settings.
* Internet Gateways are designed to be highly available and scalable. They are automatically replicated across multiple availability zones within a region, providing redundancy and fault tolerance.

IGW helps to establish internet connectivity from public subnets but how about private subnets? There can be scenarios of private subnet resources requiring internet access, one such example could be downloading the latest software update. The private subnet to public internet connectivity is achieved via a NAT Gateway.

**NAT Gateway**

The cooks working in the kitchen asked the head chef if she can collect feedback from customers about the food served—only the head chef is allowed to interact with customers directly in the main restaurant space and cooks only work in the kitchen, just to avoid overwhelming the customers or increasing the crowd in the main area.

The role of head chef is served by the NAT gateway for a subnet—it helps instances in a private subnet to connect with the internet via the Internet Gateway. You may think of NAT gateway as a bridge between internet gateway and private instances. Internet gateway requires public IP for interaction with the internet and NAT gateway facilitates that support. This support is available for TCP, UDP and ICMP protocols.

NAT Gateways (Network Address Translation Gateways) are AWS-managed network devices that allow resources within private subnets in a VPC to initiate outbound internet connections while preventing direct inbound access from the internet by hiding their private IP addresses.

To go a layer deeper, NAT means Network Address Translation, in simple terms it converts private IP address to NAT device public IP address and is mapped back to private IP address on return of response from internet. To establish a NAT Gateway, you need to allocate an Elastic IP address (EIP) and associate it with the NAT Gateway. The EIP serves as a static, public IP address that represents the NAT Gateway and is used for communication with the internet.

Here are the key things you need to consider about NAT Gateway:

* For high availability, NAT Gateway should be set up at the AZ level so that if an AZ goes down, it doesn’t impact traffic serving capability from other AZs.
* Another reason for NAT Gateway division at AZ level could be avoiding packet drops for traffic greater than 10 million packets per second.
* NAT gateway can’t be used by external internet to initiate connection with instances in private subnet.
* NAT gateway provides public(default) and private connectivity. Public will be useful for connectivity with outside internet while private will be useful for connection with other VPCs or on-premise networks.
* NAT Gateway comes with additional infrastructure cost unlike Internet Gateway, whose pricing only depends on traffic flow via it.

As we discussed connectivity to personal data centers(meaning VPC), there could also be scenarios where we own multiple such data centers. The follow up section covers different mechanisms available in AWS to create cross VPC connectivity,

**Amazon VPC to Amazon VPC Connectivity**

You may have use cases such as AWS resources integration across VPCs, security, presence in multiple regions, etc. where different components need to connect with each other residing in different VPCs, which could be in the same or different AWS accounts while maintaining isolation. To return to our cafe example, Cafe Delhi Heights is becoming more popular day by day and more people are coming in, so the restaurant head opened a new place for customers to sit—however, the food is still prepared in the old kitchen. Now here, there is a requirement to establish good connectivity for faster delivery of food to the newly opened location, in a secure way so as no one finds out. In similar fashion, there can be a need to operate two microservices in different VPCs(or different AWS accounts altogether) and for these microservices to communicate with each other in a secure way, AWS provides different connectivity options.

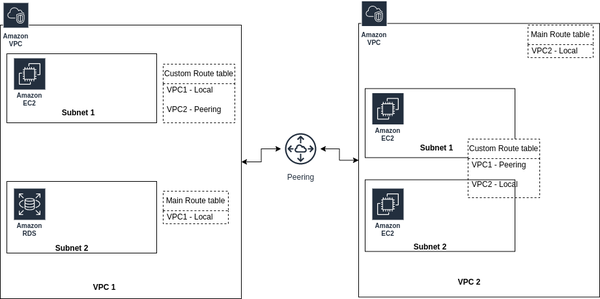
These connectivity options mainly fall under two kinds of relationships: many-to-many and hub-and-spoke. Traffic is managed individually between each VPCs in many-to-many VPC relationships, whereas a central resource manages traffic routing between the VPCs in hub-and-spoke model. The follow up section explores both of these models.

**Amazon VPC Peering**

[VPC Peering](https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html) is based on a many-to-many approach where one VPC peers with another VPC to enable full bidirectional private network connectivity. It enables resources in different VPCs to communicate with each other using private IP addresses as if they were in the same network. VPC Peering doesn’t support transitive dependency and can be a cost effective interconnectivity method if the number of VPC is less than 10. As the number of VPC increases, the mesh can become really complex to manage and operate.

You can manage the connections via route tables, SGs and NACLs  to allow specific resources or subnets to utilize the VPC peering connection. The network packets between the VPCs flow via AWS private network with no bandwidth constraints—there is no physical hardware required for this setup, you only pay for the amount of data transfer.

Think of a scenario where all the Cafe Delhi Heights restaurants have their own bar, but food is prepared at a central location and supplied to all the smaller setups, extending this example to a system architecture consisting of web frontend and backend. The connectivity between the backend and frontend tier can be established via VPC Peering assuming both are hosted in separate VPC. [Figure 5-7](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_7_vpc_peering) shows connectivity between servers running in different VPCs via VPC Peering connection.



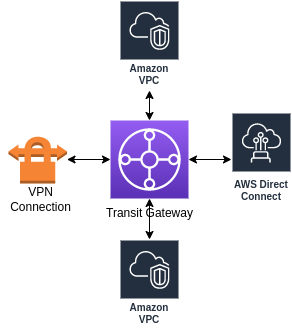
**Figure 5-7. VPC Peering**

The downside of establishing connection with VPCs at large scale via VPC Peering is resolved with another AWS Service called AWS [Transit Gateways](https://aws.amazon.com/transit-gateway/).

**AWS Transit Gateways**

Transit Gateway is a scalable solution to establish connectivity between multiple VPCs, on-premise networks and other AWS services. Transit Gateway is a regional resource based on hub and spoke model which acts as intermediary to set up all the network routing at a single place via routing tables, be it VPCs or hybrid connectivity methods such as Virtual Private Network ([VPN](https://aws.amazon.com/vpn/)) or [AWS Direct Connect](https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html).

For better control of networking routes, Transit Gateway can be set up in a separate networking AWS account where network engineers can manage at centralized location. [Figure 5-8](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_8_aws_transit_gateway) shows Transit Gateway acting as a central resource for providing connectivity between VPCs, AWS Direct Connect and VPN Connection.



**Figure 5-8. AWS Transit Gateway**

The key difference between Transit Gateway and VPC Peering is scale—transit gateway is scalable for connectivity across thousands of VPCs. Other parameters are favorable to VPC peering such as lower cost, no bandwidth constraints and reduced latency.

[Figure 5-8](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_8_aws_transit_gateway) depicted frontend and backend tier connectivity via VPC Peering which are present in different VPCs—another solution for this kind of setup is utilizing AWS [PrivateLink](https://aws.amazon.com/privatelink/" \t "_blank).

**AWS PrivateLink**

AWS PrivateLink helps to privately expose an application to consumers in another VPC and ensures traffic flows in AWS backbone network without going over the internet. The key difference is traffic flow direction—VPC Peering provides bidirectional connectivity, but if clients need to server requests only using private IP addresses, AWS PrivateLink will facilitate this kind of connectivity.

**Connectivity via AWS PrivateLink**

AWS PrivateLink is established between two parties: the one which allows access to its specific service can be referred as the service provider and others consuming this service are considered the consumers.

*Service Provider*

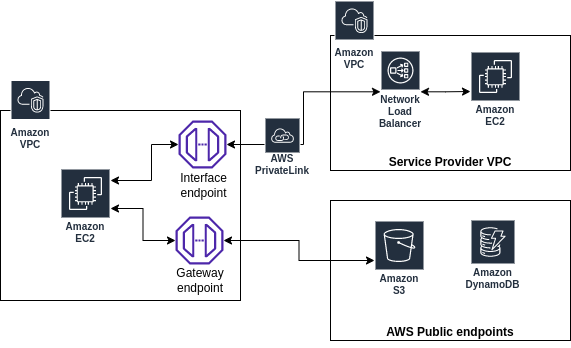
To make service available in a region, service providers will create an endpoint service which is mapped with a network load balancer. Application Load balancer can’t be directly attached with endpoint service.

*Service Consumer*

Service consumers create a VPC endpoint to connect their VPC to endpoint services by specifying the service name which is being created by the service provider.

As represented in [Figure 5-9](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_9_connectivity_via_endpoint_services), the endpoint could be an  interface endpoint or gateway endpoint. An Interface endpoint helps to establish connection to an endpoint service in another VPC, where Gateway helps to connect with Amazon S3 or DynamoDB using private IP addresses.

One more type is gateway load balancer which can be used to manage third party virtual appliances, for example compliance. (More on load balancer shortly.)



**Figure 5-9. Connectivity via Endpoint Services**

Apart from establishing connectivity between multiple VPCs, there are certain businesses which still operate on on-premise data centers. In the process of migrating to Cloud, there can always be an intermediary state where some operations are served by on-premise data centers and rest via AWS Cloud. Let’s explore different solutions offered by AWS to establish connectivity between these two separate data centers.

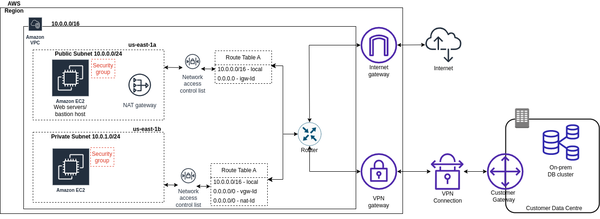
**Hybrid Connectivity**

Current infrastructure of Cafe Delhi Heights is that food is prepared at a central location and a bar is located at all the locations. To make our systems more efficient, we’re planning to move most of our food preparation to all the locations as well, but still some of the key dishes will be prepared at the central location. In this scenario, the operations happen at two places, this is what we mean by hybrid connectivity.

It could be possible that you started with your personal data center but over time you decided to move to AWS Cloud. Now, the service infrastructure is maintained by AWS but the databases are still managed on an on-premise data center. To support this, there needs to be some mechanism to establish connectivity between on-premise data centers and AWS data centers—that’s a Virtual Private Network.

**AWS Virtual Private Network**

What mechanism can be used to securely connect on-premise data centers to AWS VPC? The service framework which ensures this data security is Virtual Private Network, aka VPN. [Figure 5-10](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_10_vpn_connection) describes an example connection setup using AWS VPN from customer data center to VPC located in an AWS region.



**Figure 5-10. VPN Connection**

The components for VPN setup are:

*Customer Gateway*

A software or hardware component which is setup at the customer’s end. You can read more about device requirements [here](https://docs.aws.amazon.com/vpn/latest/s2svpn/your-cgw.htm).

*VPN Connection*

A secure and encrypted connection  channel between the customer gateway and VPN gateway. There are two tunnels being setup so as to avoid any availability issue due to failure or scheduled maintenance.

*VPN Gateway*

This component is present at AWS side to ensure communication between VPC and VPN connection.

VPN  transfers data packets over the internet anonymously. AWS provides another service to configure dedicated networks for connectivity, referred to as AWS Direct Connect.

**AWS Direct Connect**

AWS Direct Connect provides the capability to configure a dedicated network connection (Ethernet fiber-optic cable) for data transfer from the customer data center to an AWS direct connect location without the use of the internet. AWS direct connect location is configured with a router to route the traffic and connect to the AWS backbone network.  It provides consistent and high-bandwidth connectivity, suitable for large-scale or latency-sensitive workloads. You can use [AWS Direct Connect Resiliency toolkit](https://docs.aws.amazon.com/directconnect/latest/UserGuide/resiliency_toolkit.htm) to ensure maximum availability of connections set up from personal data center to AWS Direct Connect Location. [Figure 5-11](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_11_aws_vpc_connectivity_with_on_prem_data_center_via) is an extension of [Figure 5-6](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_6_overview_of_different_networking_components) and describes an example connection setup from customer data center to Amazon VPC via Direct Connect location. The main components are:

*Customer Router*

The customer router is installed at an on-premises data center holding all networking rules and helps routing the traffic from data center to AWS Direct Connect Location. Customer Router connects with router at Direct Connect Location via 802.1Q VLAN ethernet cable.

*Direct Connect Location*

Direct Connect is available worldwide at multiple [locations](https://aws.amazon.com/directconnect/locations/). We can select a location closest to on-premise data center to minimize cost and latency. You can check for all networking requirements on [AWS doc](https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html#overview_requirements).

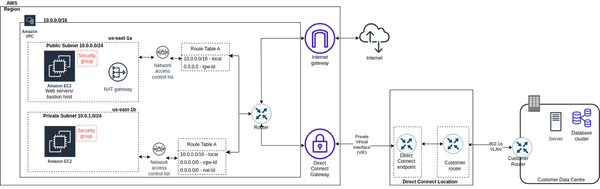
*Direct Connect Gateway*

Direct Connect Gateway helps in establishing connection between VPC and Direct Connect Location via private virtual interface.

*Virtual Interface(VIF)*

Virtual Interfaces are helpful in setting up private secure connections to required resources such as S3 without going over the internet. You can select either Public VIF, Private VIF or transit VIF depending on use-case.

* Public VIF is helpful in connecting to AWS resources over public IP addresses such as Amazon S3.
* Private VIF is helpful in connecting to AWS resources using their private IP addresses hosted in AWS VPC.
* Transit VIF is helpful in connecting to AWS resources using their private IP addresses hosted in AWS VPC through the transit gateway.



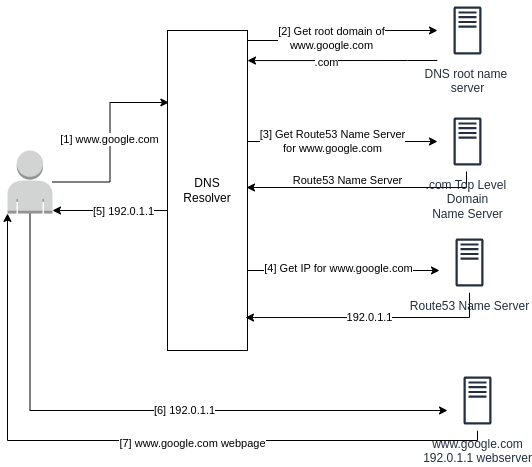
**Figure 5-11. AWS VPC connectivity with On-prem data center via Direct Connect**

We explored multiple concepts and AWS services revolving around how network packets flow inside the Amazon network, as well as multiple connectivity options. The rest of the chapter focuses on the entry point for these network packets as applications scale such as [Route 53](https://aws.amazon.com/route53/), [Load Balancers](https://aws.amazon.com/elasticloadbalancing/), [API Gateway](https://aws.amazon.com/api-gateway/). We’ll conclude the chapter by discussing AWS provided Content Delivery Network referred as [AWS CloudFront](https://aws.amazon.com/cloudfront/) for placing data content near to users to enable faster retrieval.

**Amazon Route 53**

Route 53 is a scalable and highly available Domain Name System(DNS) available in the AWS ecosystem which helps in domain registration, DNS routing and health checking. We started our ‘IP Addresses’ section by stating that every device on the internet requires an IP address to establish connection with other devices, but what if a human being is operating one side of a connection? It can be very difficult to remember all the IP addresses, well unless you’ve photographic memory.

Human beings are good at remembering names as compared to numbers—it’s easier to remember [www.google.com](http://www.google.com/) instead of 192.168.1.0 or to remember Cafe Delhi Heights, Ambience Mall instead of latitude and longitude as 28.525446566084423, 77.09008858115097. Domain Name System makes our lives easy by facilitating the conversion from domain name to IP addresses. [Figure 5-12](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_12_dns_resolution_via_route_53) shows how a domain such as [www.google.com](http://www.google.com/) is resolved to an IP address and finally accessible to end users. After the DNS resolution in Step 1 to 5, the browser establishes TCP connection with google web servers and then sends a HTTP request to the servers. The server handles the request and sends back an HTTP response as final step 7. The browser renders this response in the browser tab for further user interactions. We should have this general understanding of what happens when we type in [www.google.com](http://www.google.com/) in the web browser to understand the data flow on the internet.



**Figure 5-12. DNS resolution via Route 53**

These are the key considerations about Route 53:

* Route 53 can be used to register new domain names and additionally you can transfer existing DNS to be managed by it.
* You can create, update and manage your public DNS records via AWS Route 53 along with health checks to monitor the health of applications, web servers and related resources.
* You can configure [routing policies](https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html) for traffic management on DNS records, specifying how Route53 responds to queries.
* Route53 supports various DNS record types such as A, AAAA, CNAME, MX, TXT, and more, enabling flexible DNS configuration.
* DNS records of your domain are collectively stored in Hosted Zones to answer the domain queries.
* Route 53 utilizes Anycast as networking and routing technology, which helps in reduced latency via routing requests through the nearest data center and higher availability via presence of multiple servers to respond to traffic instead of just one origin server.

Next we’ll turn our attention to the AWS Elastic Load Balancer.

**Amazon Elastic Load Balancer**

As discussed in Chapter 5 on Scaling Approaches and Mechanisms, Load Balancers help improve the availability, scalability, and fault tolerance of applications by distributing traffic across healthy targets. Load balancers can automatically scale based on traffic patterns and health checks, ensuring optimal performance.  
The AWS Elastic Load Balancer (ELB) is a managed service which scales for the customer’s traffic on the go and provides the following capabilities:

* Distribute the incoming traffic among a pool of resources.
* Capability to serve requests without disruption as new resources are added or old ones removed.
* Monitoring resource health via ELB provided health checks.

AWS ELB is available as Application Load Balancer (ALB), Network Load Balancer (NLB), Classic Load Balancer(CLB) and Gateway Load Balancer(GWLB). As the name suggests, ALB operates at application layer(Layer 7) for HTTP, HTTPS and gRPC protocols, NLB operates at network layer(Layer 4) for TCP, UDP and TLS protocols and CLB is a legacy version of load balancer which supports both layer4/layer7 traffic. GWLB is used as a Layer 3 Gateway and Layer 4 load balancer for IP protocol.

You can read a detailed product comparison on [AWS](https://aws.amazon.com/elasticloadbalancing/features/#compare).

Our recommendation is to take a bottoms up approach by thinking about the base feature requirements for a given workload. These requirements along with pricing will decide which load balancer type is best suitable to your workload—there can be additional features offered by ELB that will be cherry on the top. [Figure 5-13](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_13_elastic_load_balancer) shows traffic distribution by ELB to different types of targets. The main components of ELB are:

*Load Balancer*

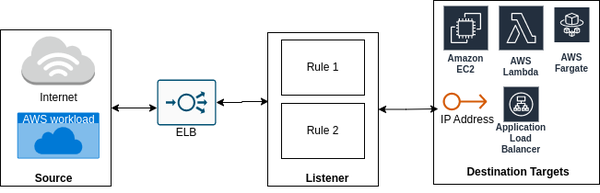
LB is a single point of contact from the client’s perspective, it can then further forward the request to configured listeners.

*Listeners*

Listener is a process that checks for the client’s request using the configured protocol and port number. The request is forwarded to target groups based on the rules associated with a listener.

*Target Groups*

Target Groups directs traffic to configured targets using the configured port and protocol, for example EC2 instances, IP addresses, etc.



**Figure 5-13. Elastic Load Balancer**

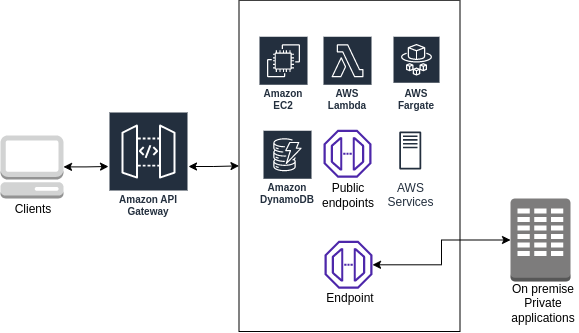
A few key considerations about ELB are:

* NLB can have targets such as ALB, containers, instances and IP. ALB can have targets as containers, instances, IP and lambda, and GWLB can have targets as IP and instance.
* Gateway Load Balancers are useful in systems such as firewalls, intrusion detection and prevention systems, deep packet inspection systems, etc. It enables customers to deploy, scale and manage virtual appliances. One example scenario for inspection systems is placement of GWLB in between source and destination for packet analysis and monitoring.
* ALB doesn’t provide support for private link and static IP addresses for inbound traffic— NLB is a suitable choice for this use-case. In scenarios where additional features of ALB are value-add to NLB or vice-versa, ALB can be used as a target to NLB to gain benefits from both worlds.
* As ALB has request header data, it can support request routing in multiple ways, such as path based, host based, query string parameter based and source IP address based routing. In contrast to this, NLB doesn’t have capability to inspect the HTTP request which makes it a little lighter in processing as compared to ALB resulting in reduced latency.
* NLB is optimized to handle sudden spiky traffic patterns; for ALB it’s a best practice to inform AWS Support in advance if a traffic spike is expected to pre-allocate the capacity.
* NLB supports long lived TCP connections unlike ALB. This is helpful in scenarios with requirements such as a huge number of persistent connections, such as WebSocket connection for an online gaming application.
* ALB provides support for AWS Web Application Firewall(WAF) and authentication mechanisms such as Amazon Cognito, OpenID Connect, etc. This helps in offloading this responsibility from the application and making it lighter.
* The load balancer will reside in a public subnet and you can host the backend resources in a private subnet, which are not directly accessible to outside traffic.
* Idle Timeout configuration for ALB can be set between 1 to 4000 seconds (the default is 60 seconds) where for NLB this configuration is 350 seconds.

Now let’s turn our attention to the API gateway.

**Amazon API Gateway**

API Gateway is a fully managed AWS service that helps in creating, publishing, maintaining, monitoring and securing REST, HTTP and WebSocket APIs. Consider a scenario that instead of human beings serving your order at Cafe Delhi Heights, there are robots deployed who take and serve the food orders. You order food by selecting the food items from the kiosk installed at the table and once the food is prepared, it is served by robots to you—well in this scenario you don’t know if in the kitchen the food is prepared by robots or human beings. That’s the beauty of resource abstraction, customers use the API Gateway’s published API to perform specific functions at scale and API Gateway can internally connect with any AWS service as shown in [Figure 5-14](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_14_api_gateway_connectivity_with_aws_services).



**Figure 5-14. API Gateway connectivity with AWS Services**

There are a few key considerations with API Gateway:

* API Gateway supports both stateless and stateful APIs.
  + REST and HTTP APIs are type of stateless APIs. These both support the same basic functionality and are HTTP based to support standard methods like  GET, PUT, POST, etc. REST APIs support some additional features such as API keys, per-client rate/usage throttling, request validation etc.
  + WebSocket APIs are stateful which operate on the basis of WebSocket protocol with full duplex client-server communication.
* API Gateway can support authentication via AWS IAM(Identity and Access Management) policies, Lambda authorizer functions and Amazon Cognito user pools.
* API Gateway provides monitoring via CloudWatch and CloudTrail services.
* API Gateway can avoid web exploits such as SQL injection via AWS Web Application Firewall(WAF) integration.
* API Gateway can directly connect with AWS Services such as DynamoDB(DDB) via service APIs reducing intermediary infrastructure cost. Consider a scenario of  retrieving DDB record basis partition key. The general implementation would be API Gateway invokes a Lambda which then connects with DDB to get data. API Gateway removes the need for AWS Lambda in between and can directly connect with DDB and serve responses.
* API Gateway doesn’t provide health checks for backend resources the way it is supported via ELB.
* API Gateway REST APIs timeout configuration lies between 50 milliseconds to 29 seconds, 30 seconds for HTTP APIs and 2 hrs connection duration for WebSocket API with idle timeout as 10 minutes.
* API Gateway can be used for cross account/region integration—for example, AWS Lambda owned by different teams in different AWS accounts as per business requirements but served by central AWS account API Gateway.
* API Gateway supports caching of endpoint’s responses which helps in reducing traffic to endpoint and improvement in latency. Time to Live(TTL) for caching can vary from 0(caching is disabled) to 3600 seconds with default as 300 seconds.

API Gateway and Load Balancers help to abstract out backend infrastructure from customers. Consider a scenario where a customer’s requests hit API Gateway and further one of the microservice fetches data from S3, this operation is definitely latency intensive if the file size is large. The latency can be reduced by placing the content near to the customer’s location via the AWS Content Delivery Network service referred as CloudFront.

**Amazon CloudFront**

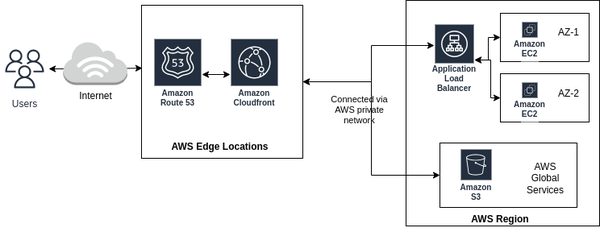
We briefly touched on two ways to manage food preparation and delivery operations for Cafe Delhi Heights, first one was food is prepared at a central location and delivered to smaller storefronts as per demand and second was to set up kitchens at all the locations.

There is definitely an extra cost to set up kitchens at every location—another option is analyzing the food item demands and preparing the food in advance at a central location and storing it at smaller outlets ahead of time. This way the food is prepared at only a single place and the customers are served at all the outlets in the minimum time possible.

In software systems, one key factor to improve customer’s experience is via serving the request with minimum latency. One potential solution is to replicate infrastructure in multiple AWS regions and utilize AWS local zones. This solution should only be accessed as a last resort for your business architecture and latency should not be the key factor to finalize it. Another cost effective solution is to cache the content near the customer’s location via a Content Delivery Network (CDN), as discussed in Chapter 4. Amazon CloudFront is CDN, a world-wide network of data centers called edge locations which helps to achieve low latency for serving both static and dynamic content in a secure way via AWS Shield, IAM, WAF and TLS certificates.

There are a few things you need to keep in mind about CloudFront:

* CloudFront can be used to serve both static and dynamic content over HTTP or WebSocket protocols. For example, static content placed in S3 buckets or dynamic content generated via any web service such as running on EC2 servers can be directly served via CloudFront as described in [Figure 5-15](https://learning.oreilly.com/library/view/system-design-on/9781098146887/ch05.html#fig_15_content_distribution_via_amazon_cloudfront).
* CloudFront by default integrates with AWS Shield to help avoid DDoS attacks and additionally can be integrated with AWS WAF for application layer security.
* CloudFront provides both encryption at rest and in transit. Data at edge locations is always stored in encrypted format and Amazon Certificate Manager or custom certificates can be used for in-transit traffic.
* For ensuring user level access to content, CloudFront provides options for signed cookies, signed URLs and geo-restriction.



**Figure 5-15. Content Distribution via Amazon CloudFront**

In short, Amazon CloudFront allows us to cache the content near our application users and serve the user queries faster.

**Conclusion**

We started off this chapter with discussion around how you can get started with AWS Cloud and then we dove deep into different networking concepts and services offered by AWS to set up networking infrastructure on the AWS Cloud. AWS Cloud comes with a plethora of services and it’s really important to understand how a specific service will solve a problem statement. It’s not easy and feasible for everyone to set up their own personal data center and AWS solves this problem by offering Amazon VPC as a solution.

You can assume Amazon VPC as your personal data center. There may be requirements for multiple data centers to connect and AWS provides different connectivity mechanisms to enable this support. It is also important that only intended users access the resources hosted in AWS Cloud and this can be ensured by utilizing security mechanisms such as security groups, NACLs and adding appropriate rules in the route tables.

Finally we discussed how the internal resources can be abstracted out from external users by using solutions such as Amazon ELB and Amazon API Gateway. AWS also offers a highly scalable DNS service, Amazon Route 53 for managing domain names. Consider a simple example to illustrate the usage of Amazon Route 53—let’s say we started off with using NLB and the users hit this endpoint. At a later point, we realized ALB was a more appropriate choice instead of NLB. In this situation, there are three options–we could share the new ALB endpoint with customers, keep NLB as front-facing to ALB with ALB redirecting traffic to the application, and use Amazon Route 53 domain so the same domain name is accessible to customers with no impact on how internal resources change over time.

Data storage solutions are an important aspect to consider while storing and maintaining data in the cloud. In the next chapter, we’ll explore different types of AWS Storage services such as DynamoDB, S3, Relational Databases and further discuss each of these services with respect to the use cases they best serve.