MicroDegree

AWS Study Materials

ಕನ್ನಡಿಗರಿಗೆ IT Job-Ready ಮಾಡುವ ಅಭಿಯಾನ

Overview:

- Introduction Page 3
- Networking Page 16
- EC2(Elastic Compute Cloud) Page 34
- Identity Access Management Page 36
- Elastic Load Balancing(ELB) Page 41
- Logging & Monitoring tool Page 58
- S3(Simple Storage Service) Page 61
- Database on AWS Page 77
- Server less Architectures Page 78

Introduction:

- What is Cloud Computing?
- Why do we go for Cloud Computing?
- Cloud Computing Models
- Computer Network , Infrastructure as a Company
- IP Addresses

What is Cloud Service?

Services and Solutions that are delivered and consumed in real time over internet are Cloud Services.
When you store your photos online, use webmail or social networking site, you will use “Cloud computing” Service.

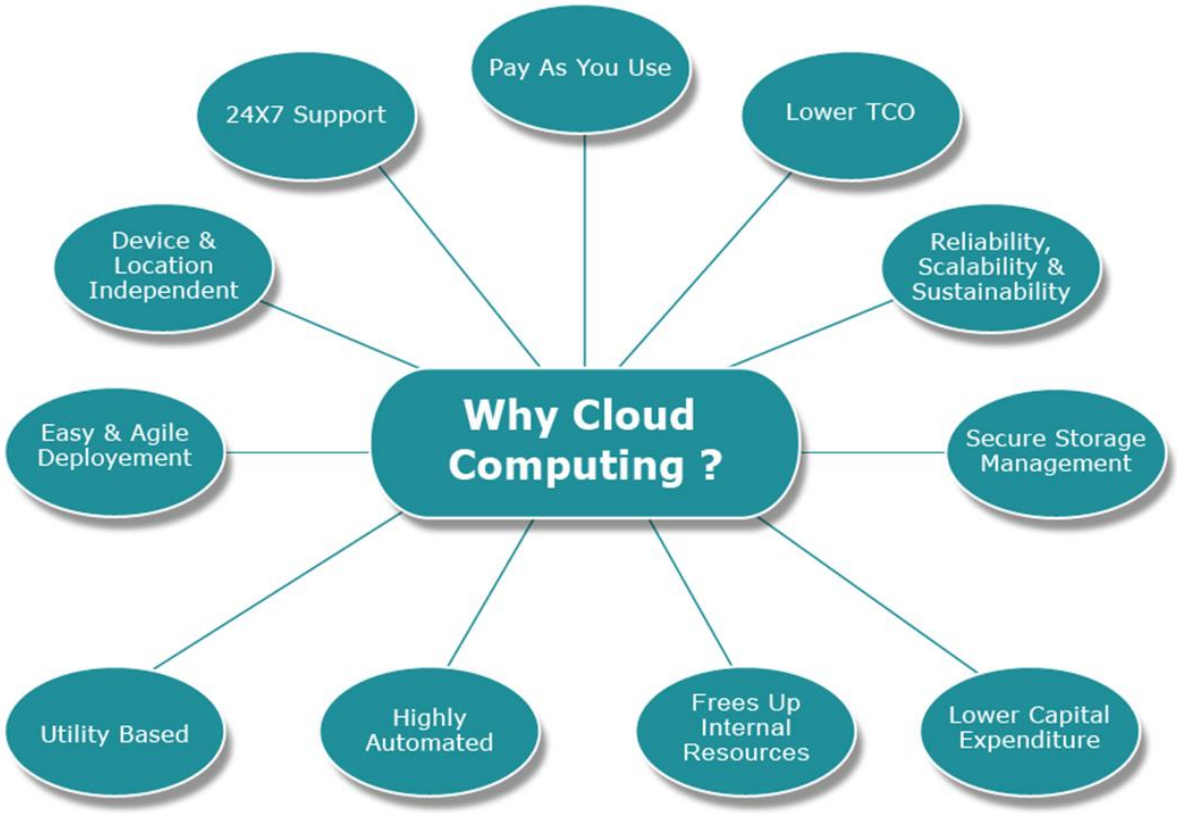
What is Cloud Computing?

Cloud Computing is a delivery model of Computing services over the internet.
It enables real time development, development and delivery of broad range of products, services and solutions.
“Cloud computing is a style of computing where massively scalable IT-related capabilities are provided as a service across the Internet to multiple external customers”

Why do we go for Cloud Computing?

- Lower Computing Cost
- Improved Performance
- Reduced Software Cost
- Instant Software Updates
- Unlimited Storage Capacity
- Increased Data Reliability

- Device Independence and the “always on! Anywhere and any place”
- Free from Maintenance and the “no-need to-know”



Cloud Computing Models

There are three main models for cloud computing. Each model represents a different part of the cloud computing stack.

Infrastructure as a Service (IaaS)

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

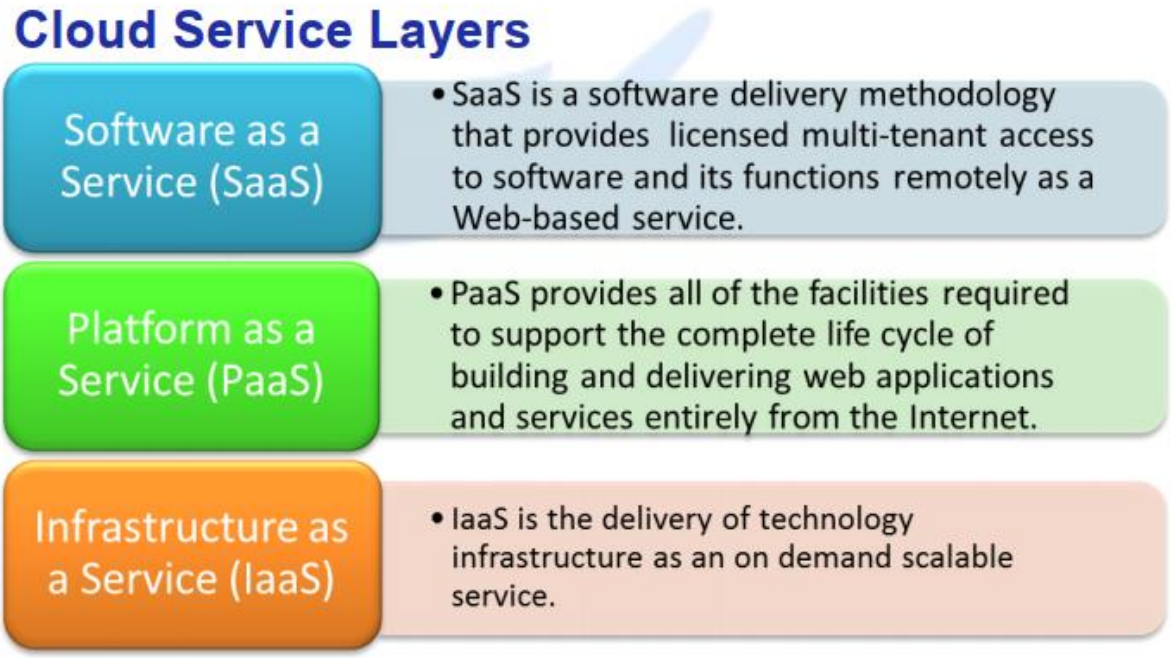
Platform as a Service (PaaS)

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don’t need to

worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Software as a Service (SaaS)

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.



IP Addresses

What is an IP Address?

- An unique identifier for a computer or device (host) on a TCP/IP network
- A 32-bit binary number usually represented as 4 decimal numbers separated by a period

Example:

206 . 40 . 185 . 73

11001110.00101000. 10111001.01001001

- Each address is 32 bits wide
- Valid addresses can range from 0.0.0.0 to 255.255.255.255, because 11111111b = $255_{10.}$
- Theoretically, a total of » 4.3 billion addresses are available, because 2^{32} = $4,294,967,296_{10.}$
- Each address consists of two parts

1. The network address
2. The host address

- Five Network Classes

1. Class A – begins with 0

00000001 (1_{10}) to 01111111 (126_{10})*

2. Class B – begins with 10

10000000 (128_{10}) to 10111111 (191_{10})

3. Class C – begins with 110

11000000 (192_{10}) to 11011111 (223_{10})

4. Class D – begins with 1110(224_{10} to 239_{10}) Reserved for multicasting

5. Class E – begins with 1111 (240_{10} to 254_{10})

Reserved for future use

How to Sign In to the AWS Console?

The AWS Management Console provides a web-based user interface that you can use to create and manage your AWS resources. For example, you can start and stop Amazon EC2 instances, create Amazon Dynamo DB tables, create Amazon S3 buckets, and so on.

Before you can use the AWS Management Console, you must sign in to your AWS account. The process that you will use to sign in to your AWS account depends on what type of AWS user you are. There are two different types of users in AWS. You are either the account owner (root user) or you are an IAM user. The root user is created when the AWS account is created using the email address and password that were used to create the account. IAM users are created by the root user or an IAM administrator within the AWS account.

Sign in as the root user

Before you sign in to an AWS account as the root user, be sure that you have the following required information.

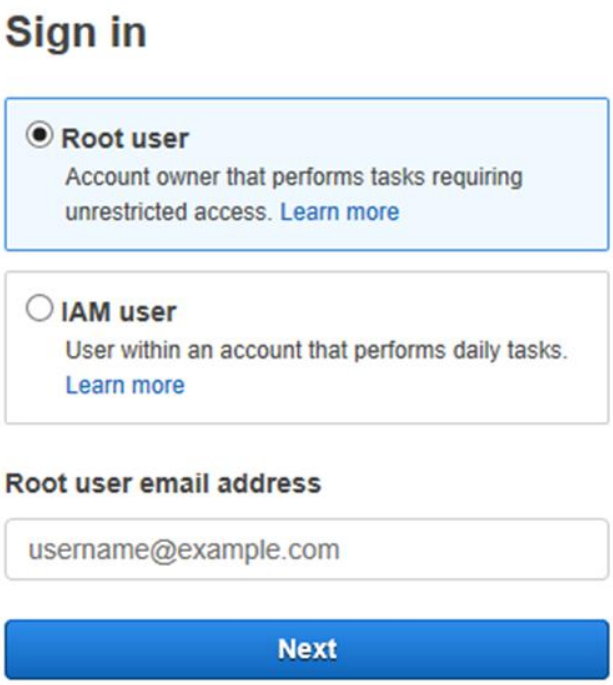
Requirements

- The email address used to create the AWS account.
- The password for the root user.

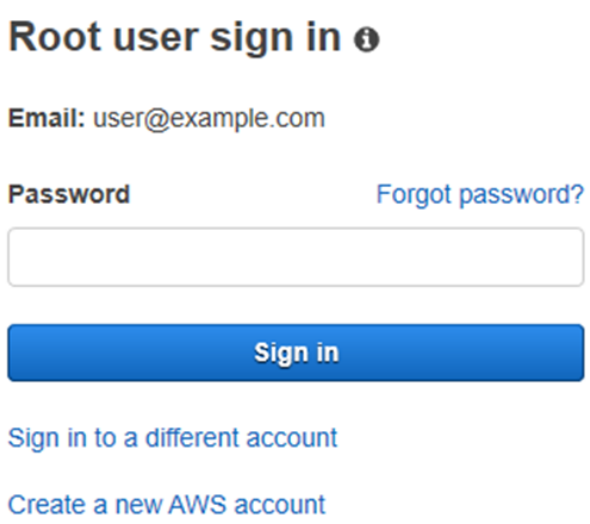
To sign in to an AWS account as the root user

1. Open https://console.aws.amazon.com/

2. If you have not signed in previously using this browser, the main sign-in page appears as follows. Choose **Root user**, enter the email address associated with your account, and choose **next**.



3. If you have signed in as a root user previously using this browser, your browser might remember the email address for the AWS account. If so, you'll see the screen shown in the next step instead. If you have signed in previously as an IAM user using this browser, your browser might display the IAM user sign in page instead. To return to the main sign-page, choose **Sign in using root user email**.
4. Enter your password and choose **Sign in**.



Sign in as an IAM user

Before you sign into an AWS account as an IAM user, be sure that you have the following required information. If you do not have this information, contact the administrator for the AWS account.

Requirements

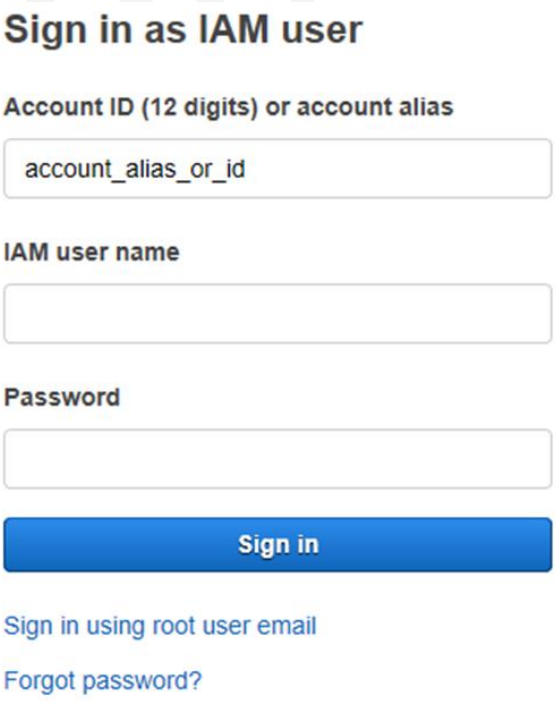
- One of the following:
  - The account alias.
  - The 12-digit AWS account ID.
- The user name for your IAM user.
- The password for your IAM user.

If you are a root user or IAM administrator and need to provide the AWS account ID or AWS account alias to an IAM user, see Your AWS account ID and its alias.

If you are an IAM user, you can log in using either a sign-in URL or the main sign-in page.

To sign in to an AWS account as an IAM user using an IAM users sign-in URL

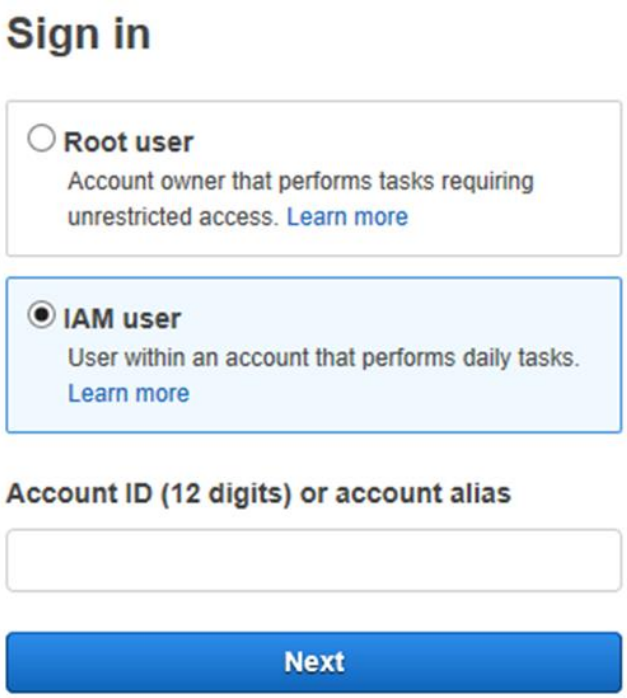
1. Open a browser and enter the following sign-in URL, replacing *account_alias_or_id* with the account alias or account ID provided by your administrator.
2. Enter your IAM user name and password and choose **Sign in**.



To sign in to an AWS account as an IAM user using the main sign-in page

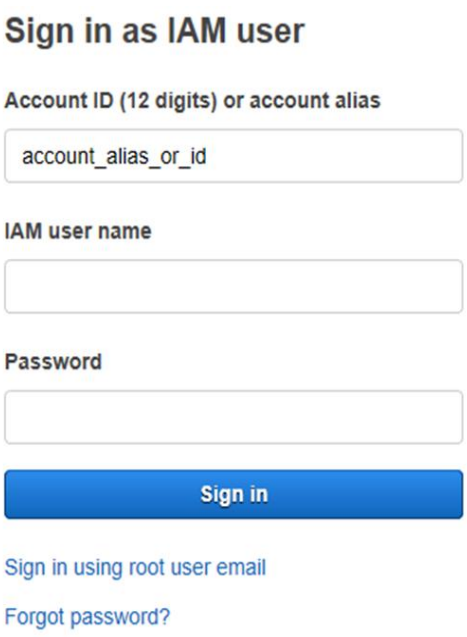
1. Open https://console.aws.amazon.com/.

2. If you have not signed in previously using this browser, the main sign-in page appears. Choose **IAM user**, enter the account alias or account ID, and choose **Next**.



If you have signed in as an IAM user previously using this browser, your browser might remember the account alias or account ID for the AWS account. If so, you'll see the screen shown in the next step instead.

Enter your IAM user name and password and choose **Sign in**.



If you have signed in as an IAM user for a different AWS account previously using this browser, or you need to sign in as a root user instead, choose **Sign in using root user email** to return to the main sign-in page.

For more click on:

https://docs.aws.amazon.com/IAM/latest/UserGuide/console.html

Networking

- What is VPC?
- Private & Public Subnets
- Cloud front
- Route53
- Securing VPC
- VPC peering
- Transit Gateway
- Updating and working with domain names

What is VPC?

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Creating Public and Private Subnets

VPC with public and private subnets (NAT)

The configuration for this scenario includes a virtual private cloud (VPC) with a public subnet and a private subnet. We recommend this scenario if you want to run a public-facing web application, while maintaining back-end servers that aren't publicly accessible. A common example is a multi-tier website, with the web servers in a public subnet and the database servers in a private subnet. You can set up security and routing so that the web servers can communicate with the database servers.

The instances in the public subnet can send outbound traffic directly to the internet, whereas the instances in the private subnet can't. Instead, the instances in the private subnet can access the internet by using a network address translation (NAT) gateway that resides in the public subnet. The database servers can connect to the internet for software updates using the NAT gateway, but the internet cannot establish connections to the database servers.

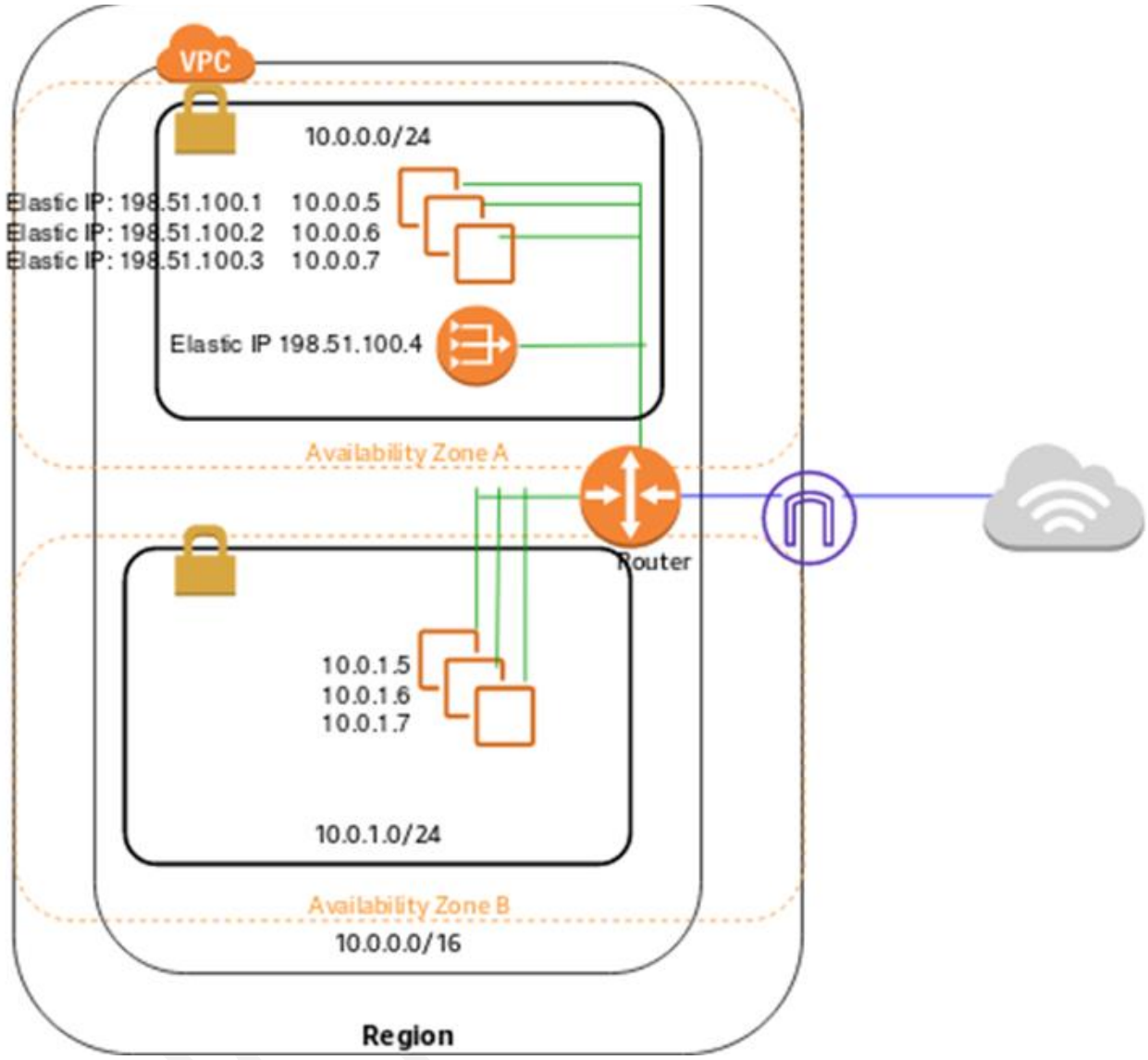
This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnets with associated IPv6 CIDR

blocks. Instances launched into the subnets can receive IPv6 addresses, and communicate using IPv6. Instances in the private subnet can use an egress-only internet gateway to connect to the internet over IPv6, but the internet cannot establish connections to the private instances over IPv6. For more information about IPv4 and IPv6 addressing, see IP Addressing in your VPC.

For information about managing your EC2 instance software, see Managing software on your Linux instance in the *Amazon EC2 User Guide for Linux Instances*.

Overview

The following diagram shows the key components of the configuration for this scenario.



The configuration for this scenario includes the following:

- A VPC with a size /16 IPv4 CIDR block (example: 10.0.0.0/16). This provides 65,536 private IPv4 addresses.
- A public subnet with a size /24 IPv4 CIDR block (example: 10.0.0.0/24). This provides 256 private IPv4 addresses. A public subnet is a subnet that's associated with a route table that has a route to an internet gateway.

- A private subnet with a size /24 IPv4 CIDR block (example: 10.0.1.0/24). This provides 256 private IPv4 addresses.
- An internet gateway. This connects the VPC to the internet and to other AWS services.
- Instances with private IPv4 addresses in the subnet range (examples: 10.0.0.5, 10.0.1.5). This enables them to communicate with each other and other instances in the VPC.
- Instances in the public subnet with Elastic IPv4 addresses (example: 198.51.100.1), which are public IPv4 addresses that enable them to be reached from the internet. The instances can have public IP addresses assigned at launch instead of Elastic IP addresses. Instances in the private subnet are back-end servers that don't need to accept incoming traffic from the internet and therefore do not have public IP addresses; however, they can send requests to the internet using the NAT gateway (see the next bullet).
- A NAT gateway with its own Elastic IPv4 address. Instances in the private subnet can send requests to the internet through the NAT gateway over IPv4 (for example, for software updates).
- A custom route table associated with the public subnet. This route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC over IPv4, and an entry that enables instances in the subnet to communicate directly with the internet over IPv4.

- The main route table associated with the private subnet. The route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC over IPv4, and an entry that enables instances in the subnet to communicate with the internet through the NAT gateway over IPv4.

For more information about subnets, see VPCs and subnets. For more information about internet gateways, see Internet gateways. For more information about NAT gateways, see NAT gateways.

Overview for IPv6

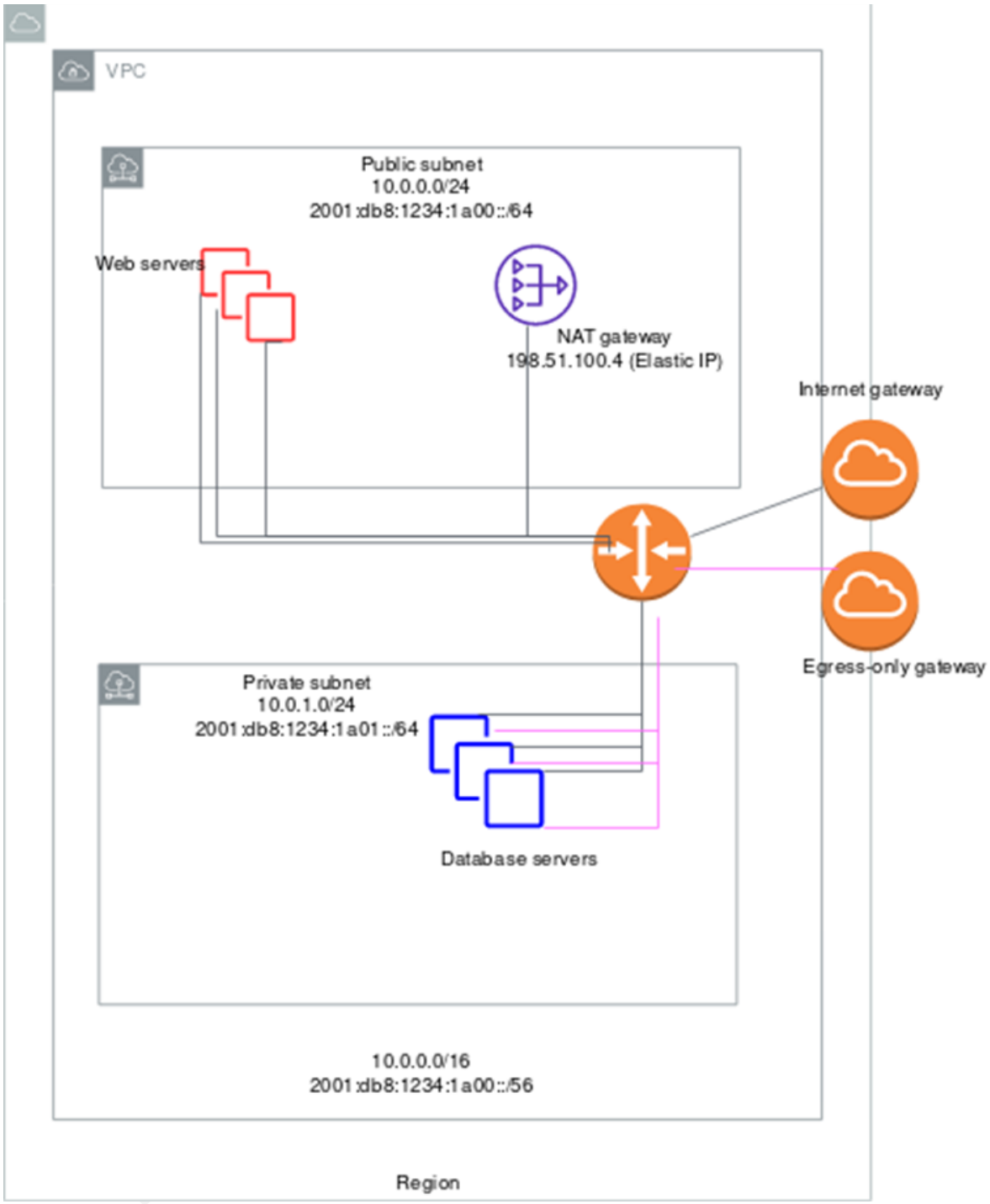
You can optionally enable IPv6 for this scenario. In addition to the components listed above, the configuration includes the following:

- A size /56 IPv6 CIDR block associated with the VPC (example: 2001:db8:1234:1a00::/56). Amazon automatically assigns the CIDR; you cannot choose the range yourself.
- A size /64 IPv6 CIDR block associated with the public subnet (example: 2001:db8:1234:1a00::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the VPC IPv6 CIDR block.
- A size /64 IPv6 CIDR block associated with the private subnet (example: 2001:db8:1234:1a01::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the subnet IPv6 CIDR block.
- IPv6 addresses assigned to the instances from the subnet range (example: 2001:db8:1234:1a00::1a).

- An egress-only internet gateway. You use the gateway to handle requests to the internet from instances in the private subnet over IPv6 (for example, for software updates). An egress-only internet gateway is necessary if you want instances in the private subnet to be able to initiate communication with the internet over IPv6. For more information, see Egress-only internet gateways.

- Route table entries in the custom route table that enable instances in the public subnet to use IPv6 to communicate with each other, and directly over the internet.

- Route table entries in the main route table that enable instances in the private subnet to use IPv6 to communicate with each other, and to communicate with the internet through an egress-only internet gateway.



For more click on:

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html

What is Amazon Cloud Front?

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.
- If the content is not in that edge location, CloudFront retrieves it from an origin that you've defined—such as an Amazon S3 bucket, a Media Package channel, or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

As an example, suppose that you're serving an image from a traditional web server, not from CloudFront. For example, you might serve an image, sunsetphoto.png, using the URL http://example.com/sunsetphoto.png.

Your users can easily navigate to this URL and see the image. But they probably don't know that their request is routed from one network to another—through the complex collection of interconnected networks that comprise the internet—until the image is found.

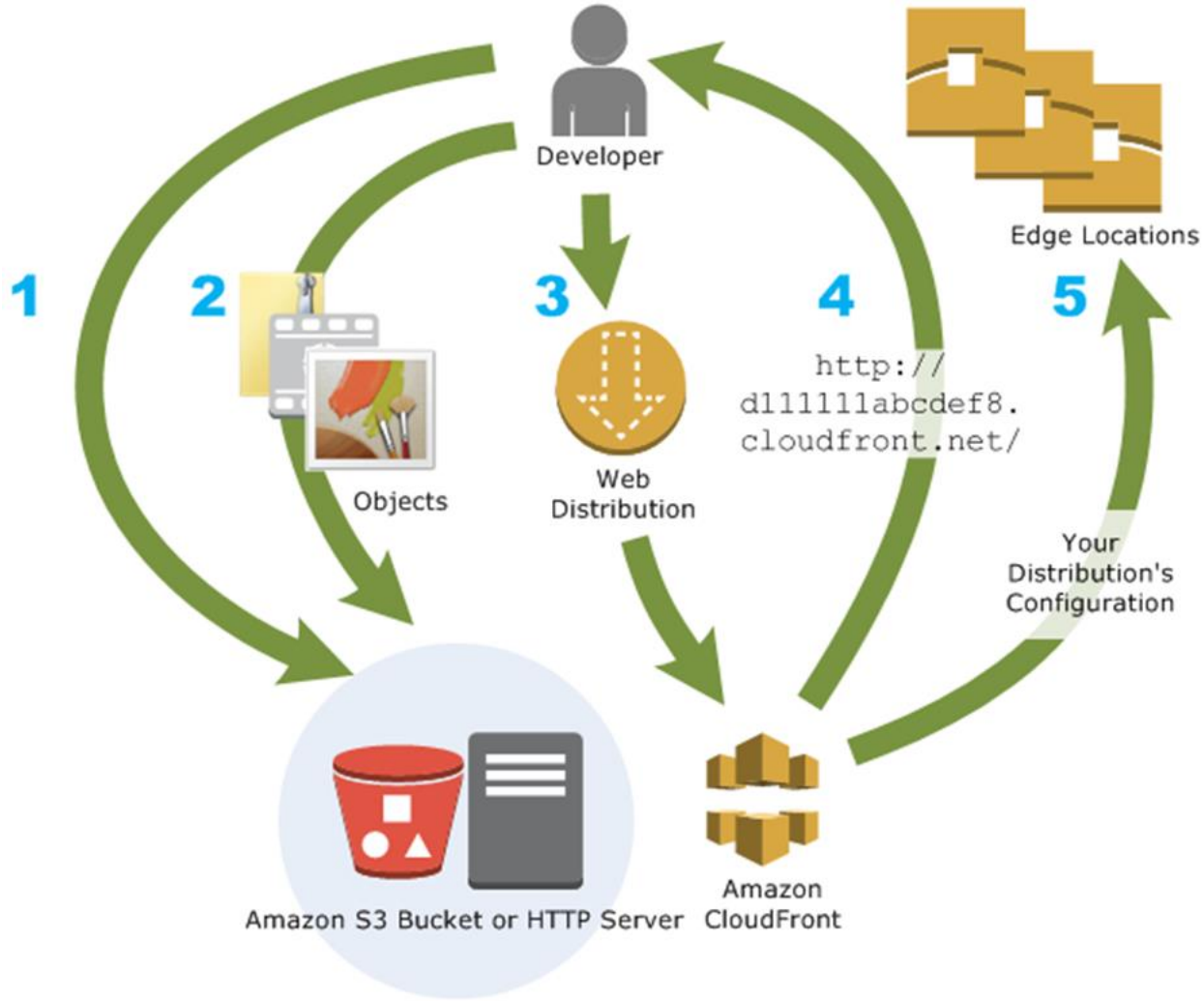
CloudFront speeds up the distribution of your content by routing each user request through the AWS backbone network to the edge location

that can best serve your content. Typically, this is a CloudFront edge server that provides the fastest delivery to the viewer. Using the AWS network dramatically reduces the number of networks that your users' requests must pass through, which improves performance. Users get lower latency—the time it takes to load the first byte of the file—and higher data transfer rates.

You also get increased reliability and availability because copies of your files (also known as *objects)* are now held (or cached) in multiple edge locations around the world.

How you set up CloudFront to deliver content

You create a CloudFront distribution to tell CloudFront where you want content to be delivered from, and the details about how to track and manage content delivery. Then CloudFront uses computers—edge servers—that are close to your viewers to deliver that content quickly when someone wants to see it or use it.


Developer
Edge Locations
1
2
3
4
5
Objects
Web Distribution
http://
d111111abcdef8.
cloudfront.net/
Your Distribution's Configuration
Amazon S3 Bucket or HTTP Server
Amazon CloudFront

How you configure CloudFront to deliver your content

1. You specify *origin servers*, like an Amazon S3 bucket or your own HTTP server, from which CloudFront gets your files which will then be distributed from CloudFront edge locations all over the world.
   An origin server stores the original, definitive version of your objects. If you're serving content over HTTP, your origin server is either an Amazon S3 bucket or an HTTP server, such as a web server. Your HTTP server can run on an Amazon Elastic Compute Cloud (Amazon EC2) instance or on a server that you manage; these servers are also known as *custom origins.*
   a. You upload your files to your origin servers. Your files, also known as *objects*, typically include web pages, images, and media files, but can be anything that can be served over HTTP.
      If you're using an Amazon S3 bucket as an origin server, you can make the objects in your bucket publicly readable, so that anyone who knows the CloudFront URLs for your objects can access them. You also have the option of keeping objects private and controlling who accesses them. See serving private content with signed URLs and signed cookies.
   b. You create a CloudFront *distribution*, which tells CloudFront which origin servers to get your files from when users request the files through your web site or application. At the same time, you specify details such as whether you want CloudFront to log all requests and whether you want the distribution to be enabled as soon as it's created.
   c. CloudFront assigns a domain name to your new distribution that you can see in the CloudFront console, or that is returned in the response to a programmatic request, for example, an API request. If you like, you can add an alternate domain name to use instead.

d. CloudFront sends your distribution's configuration (but not your content) to all of its *edge locations* or *points of presence* (POPs) — collections of servers in geographically-dispersed data centers where CloudFront caches copies of your files.

As you develop your website or application, you use the domain name that CloudFront provides for your URLs. For example, if CloudFront returns d111111abcdef8.cloudfront.net as the domain name for your distribution, the URL for logo.jpg in your Amazon S3 bucket (or in the root directory on an HTTP server) is http://d111111abcdef8.cloudfront.net/logo.jpg.

Or you can set up CloudFront to use your own domain name with your distribution. In that case, the URL might be http://www.example.com/logo.jpg.

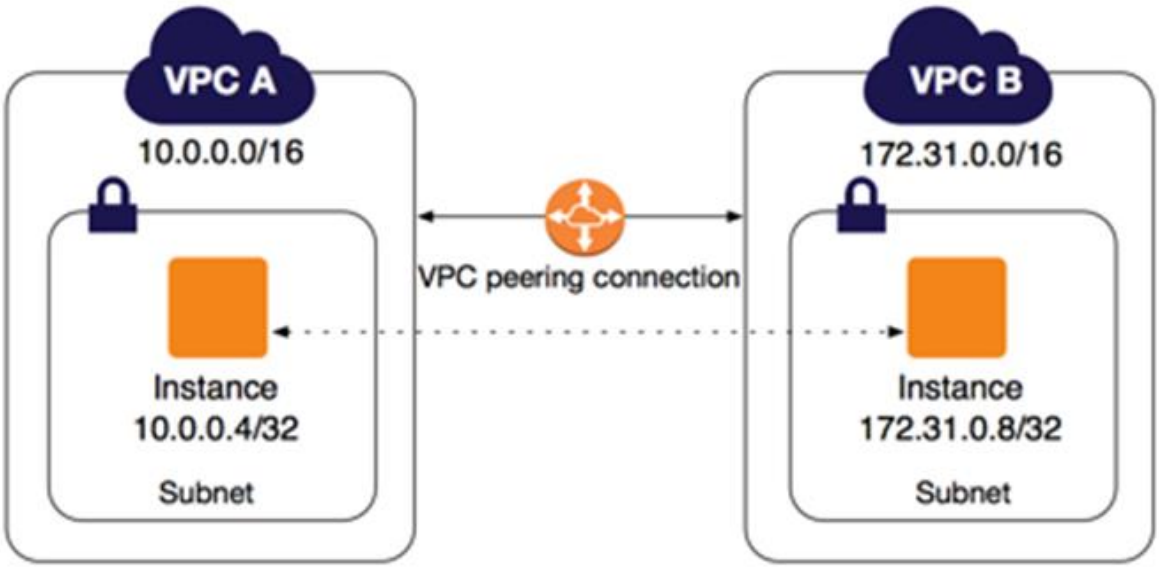
Optionally, you can configure your origin server to add headers to the files, to indicate how long you want the files to stay in the cache in CloudFront edge locations. By default, each file stays in an edge location for 24 hours before it expires. The minimum expiration time is 0 seconds; there isn't a maximum expiration time.

For more click on:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html

What is VPC peering?

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined.

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account. The VPCs can be in different regions (also known as an inter-region VPC peering connection).



AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck.

A VPC peering connection helps you to facilitate the transfer of data. For example, if you have more than one AWS account, you can peer the VPCs across those accounts to create a file sharing network. You can also use a VPC peering connection to allow other VPCs to access resources you have in one of your VPCs.

You can establish peering relationships between VPCs across different AWS Regions (also called Inter-Region VPC Peering). This allows VPC resources including EC2 instances, Amazon RDS databases and Lambda functions that run in different AWS Regions to communicate with each other using private IP addresses, without requiring gateways, VPN connections, or separate network appliances. The traffic remains in the private IP space. All inter-region traffic is encrypted with no single point of failure, or bandwidth bottleneck. Traffic always stays on the global AWS backbone, and never traverses the public internet, which reduces threats, such as common exploits, and DDoS attacks. Inter-Region VPC Peering provides a simple and cost-effective way to share resources between regions or replicate data for geographic redundancy.

For more click on:
https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html

What is a transit gateway?

A *transit gateway* is a network transit hub that you can use to interconnect your virtual private clouds (VPCs) and on-premises networks. As your cloud infrastructure expands globally, inter-Region peering connects transit gateways together using the AWS Global Infrastructure. Your data is automatically encrypted and never travels over the public internet.

Transit gateway concepts

The following are the key concepts for transit gateways:

- **Attachments** — You can attach the following:
  - One or more VPCs
  - A Connect SD-WAN/third-party network appliance
  - An AWS Direct Connect gateway
  - A peering connection with another transit gateway
  - A VPN connection to a transit gateway
- **Transit gateway Maximum Transmission Unit (MTU)** — The maximum transmission unit (MTU) of a network connection is the size, in bytes, of the largest permissible packet that can be passed over the connection. The larger the MTU of a connection, the more data that can be passed in a single packet. A transit gateway supports an MTU of 8500 bytes for traffic between VPCs, AWS Direct Connect, Transit Gateway Connect, and

peering attachments. Traffic over VPN connections can have an MTU of 1500 bytes.

- **Transit gateway route table** — A transit gateway has a default route table and can optionally have additional route tables. A route table includes dynamic and static routes that decide the next hop based on the destination IP address of the packet. The target of these routes could be any transit gateway attachment. By default, transit gateway attachments are associated with the default transit gateway route table.

- **Associations** — Each attachment is associated with exactly one route table. Each route table can be associated with zero to many attachments.

- **Route propagation** — A VPC, VPN connection, or Direct Connect gateway can dynamically propagate routes to a transit gateway route table. With a Connect attachment, the routes are propagated to a transit gateway route table by default. With a VPC, you must create static routes to send traffic to the transit gateway. With a VPN connection or a Direct Connect gateway, routes are propagated from the transit gateway to your on-premises router using Border Gateway Protocol (BGP). With a peering attachment, you must create a static route in the transit gateway route table to point to the peering attachment.

Work with transit gateways

You can create, access, and manage your transit gateways using any of the following interfaces:

- **AWS Management Console** — Provides a web interface that you can use to access your transit gateways.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including Amazon VPC, and is supported on Windows, macOS, and Linux. For more information, see AWS Command Line Interface.
- **AWS SDKs** — Provides language-specific API operations and takes care of many of the connection details, such as calculating signatures, handling request retries, and handling errors. For more information, see AWS SDKs.
- **Query API** — Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Amazon VPC, but it requires that your application handle low-level details such as generating the hash to sign the request, and handling errors.

For more click on https://docs.aws.amazon.com/vpc/?id=docs_gateway

What is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using

Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Features of Amazon EC2

Amazon EC2 provides the following features:

- Virtual computing environments, known as *instances*
- Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*
- Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as *instance store volumes*
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*

- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *Regions* and *Availability Zones*
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
- Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
- Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as *virtual private clouds* (VPCs)

For more click on:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

What is IAM?

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM features

IAM gives you the following features:

Shared access to your AWS account

You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.

Granular permissions

You can grant different permissions to different people for different resources. For example, you might allow some users complete access to Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon Dynamo DB, Amazon Redshift, and other AWS services. For other users, you can allow read-only access to just some S3 buckets, or permission to

administer just some EC2 instances, or to access your billing information but nothing else.

Secure access to AWS resources for applications that run on Amazon EC2

You can use IAM features to securely provide credentials for applications that run on EC2 instances. These credentials provide permissions for your application to access other AWS resources. Examples include S3 buckets and DynamoDB tables.

Multi-factor authentication (MFA)

You can add two-factor authentication to your account and to individual users for extra security. With MFA you or your users must provide not only a password or access key to work with your account, but also a code from a specially configured device.

Identity federation

You can allow users who already have passwords elsewhere—for example, in your corporate network or with an internet identity provider—to get temporary access to your AWS account.

Identity information for assurance

If you use AWS Cloud Trail, you receive log records that include information about those who made requests for resources in your account. That information is based on IAM identities.

PCI DSS Compliance

IAM supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.

Integrated with many AWS services

For a list of AWS services that work with IAM, see AWS services that work with IAM.

Eventually Consistent

IAM, like many other AWS services, is eventually consistent. IAM achieves high availability by replicating data across multiple servers within Amazon's data centers around the world. If a request to change some data is successful, the change is committed and safely stored. However, the change must be replicated across IAM, which can take some time. Such changes include creating or updating users, groups, roles, or policies. We recommend that you do not include such IAM changes in the critical, high-availability code paths of your application. Instead, make IAM changes in a separate initialization or setup routine that you run less frequently. Also, be sure to verify that the changes have been propagated before production workflows depend on them.

Free to use

AWS Identity and Access Management (IAM) and AWS Security Token Service (AWS STS) are features of your AWS account offered at no additional charge. You are charged only when you access other AWS services using your IAM users or AWS STS temporary security credentials.

Accessing IAM

You can work with AWS Identity and Access Management in any of the following ways.

AWS Management Console

The console is a browser-based interface to manage IAM and AWS resources.

AWS Command Line Tools

You can use the AWS command line tools to issue commands at your system's command line to perform IAM and AWS tasks. Using the command line can be faster and more convenient than the console. The command line tools are also useful if you want to build scripts that perform AWS tasks.

AWS provides two sets of command line tools: the AWS Command Line Interface (AWS CLI) and the AWS Tools for Windows PowerShell.

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to IAM and AWS. For example, the SDKs take care of tasks such as cryptographically signing requests, managing errors, and retrying requests automatically.

IAM HTTPS API

You can access IAM and AWS programmatically by using the IAM HTTPS API, which lets you issue HTTPS requests directly to the service. When you use the HTTPS API, you must include code to digitally sign requests using your credentials.

For more click on:
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

What is Elastic Load Balancing?

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP

addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

Load balancer benefits

A load balancer distributes workloads across multiple compute resources, such as virtual servers. Using a load balancer increases the availability and fault tolerance of your applications.

You can add and remove compute resources from your load balancer as your needs change, without disrupting the overall flow of requests to your applications.

You can configure health checks, which monitor the health of the compute resources, so that the load balancer sends requests only to the healthy ones. You can also offload the work of encryption and decryption to your load balancer so that your computer resources can focus on their main work.

Features of Elastic Load Balancing

Elastic Load Balancing supports the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. You can select the type of load balancer that best suits your needs. For more information, see Product comparisons.

For more information about using each load balancer, see the User Guide for Application Load Balancers, the User Guide for Network Load Balancers, the User Guide for Gateway Load Balancers, and the User Guide for Classic Load Balancers.

Accessing Elastic Load Balancing

You can create, access, and manage your load balancers using any of the following interfaces:

- **AWS Management Console**— Provides a web interface that you can use to access Elastic Load Balancing.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including Elastic Load Balancing. The AWS CLI is supported on Windows, macOS, and Linux. For more information, see AWS Command Line Interface.
- **AWS SDKs** — Provide language-specific APIs and take care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see AWS SDKs.
- **Query API**— Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Elastic Load Balancing. However, the Query API requires that your application handle low-level details such as generating the hash to sign the request, and error handling. For more information, see the following:

- Application Load Balancers and Network Load Balancers — API version 2015-12-01
- Classic Load Balancers — API version 2012-06-01

Related services

Elastic Load Balancing works with the following services to improve the availability and scalability of your applications.

- **Amazon EC2** — Virtual servers that run your applications in the cloud. You can configure your load balancer to route traffic to your EC2 instances. For more information, see the Amazon EC2 User Guide for Linux Instances or the Amazon EC2 User Guide for Windows Instances.

- **Amazon EC2 Auto Scaling** — Ensures that you are running your desired number of instances, even if an instance fails. Amazon EC2 Auto Scaling also enables you to automatically increase or decrease the number of instances as the demand on your instances changes. If you enable Auto Scaling with Elastic Load Balancing, instances that are launched by Auto Scaling are automatically registered with the load balancer. Likewise, instances that are terminated by Auto Scaling are automatically de-registered from the load balancer. For more information, see the Amazon EC2 Auto Scaling User Guide.

- **AWS Certificate Manager** — When you create an HTTPS listener, you can specify certificates provided by ACM. The load

balancer uses certificates to terminate connections and decrypt requests from clients.

- **Amazon Cloud Watch** — Enables you to monitor your load balancer and to take action as needed. For more information, see the Amazon Cloud Watch User Guide.

- **Amazon ECS** — Enables you to run, stop, and manage Docker containers on a cluster of EC2 instances. You can configure your load balancer to route traffic to your containers. For more information, see the Amazon Elastic Container Service Developer Guide.

- **AWS Global Accelerator** — Improves the availability and performance of your application. Use an accelerator to distribute traffic across multiple load balancers in one or more AWS Regions. For more information, see the AWS Global Accelerator Developer Guide.

- **Route 53** — Provides a reliable and cost-effective way to route visitors to websites by translating domain names into the numeric IP addresses that computers use to connect to each other. For example, it would translate www.example.com into the numeric IP address 192.0.2.1. AWS assigns URLs to your resources, such as load balancers. However, you might want a URL that is easy for users to remember. For example, you can map your domain name to a load balancer. For more information, see the Amazon Route 53 Developer Guide.

- **AWS WAF** — You can use AWS WAF with your Application Load Balancer to allow or block requests based on the rules in a

web access control list (web ACL).For more information, see the AWS WAF Developer Guide.

For more click on:
https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html

How Elastic Load Balancing works

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.

You configure your load balancer to accept incoming traffic by specifying one or more *listeners*. A listener is a process that checks for connection requests. It is configured with a protocol and port number for connections from clients to the load balancer. Likewise, it is configured with a protocol and port number for connections from the load balancer to the targets.

Elastic Load Balancing supports the following types of load balancers:

- Application Load Balancers
- Network Load Balancers

- Gateway Load Balancers
- Classic Load Balancers

There is a key difference in how the load balancer types are configured. With Application Load Balancers, Network Load Balancers, and Gateway Load Balancers, you register targets in target groups, and route traffic to the target groups. With Classic Load Balancers, you register instances with the load balancer.

Availability Zones & Load Balancer Nodes

When you enable an Availability Zone for your load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone. If you register targets in an Availability Zone but do not enable the Availability Zone, these registered targets do not receive traffic. Your load balancer is most effective when you ensure that each enabled Availability Zone has at least one registered target.

We recommend enabling multiple Availability Zones for all load balancers. With an Application Load Balancer however, it is a requirement that you enable at least two or more Availability Zones. This configuration helps ensure that the load balancer can continue to route traffic. If one Availability Zone becomes unavailable or has no healthy targets, the load balancer can route traffic to the healthy targets in another Availability Zone.

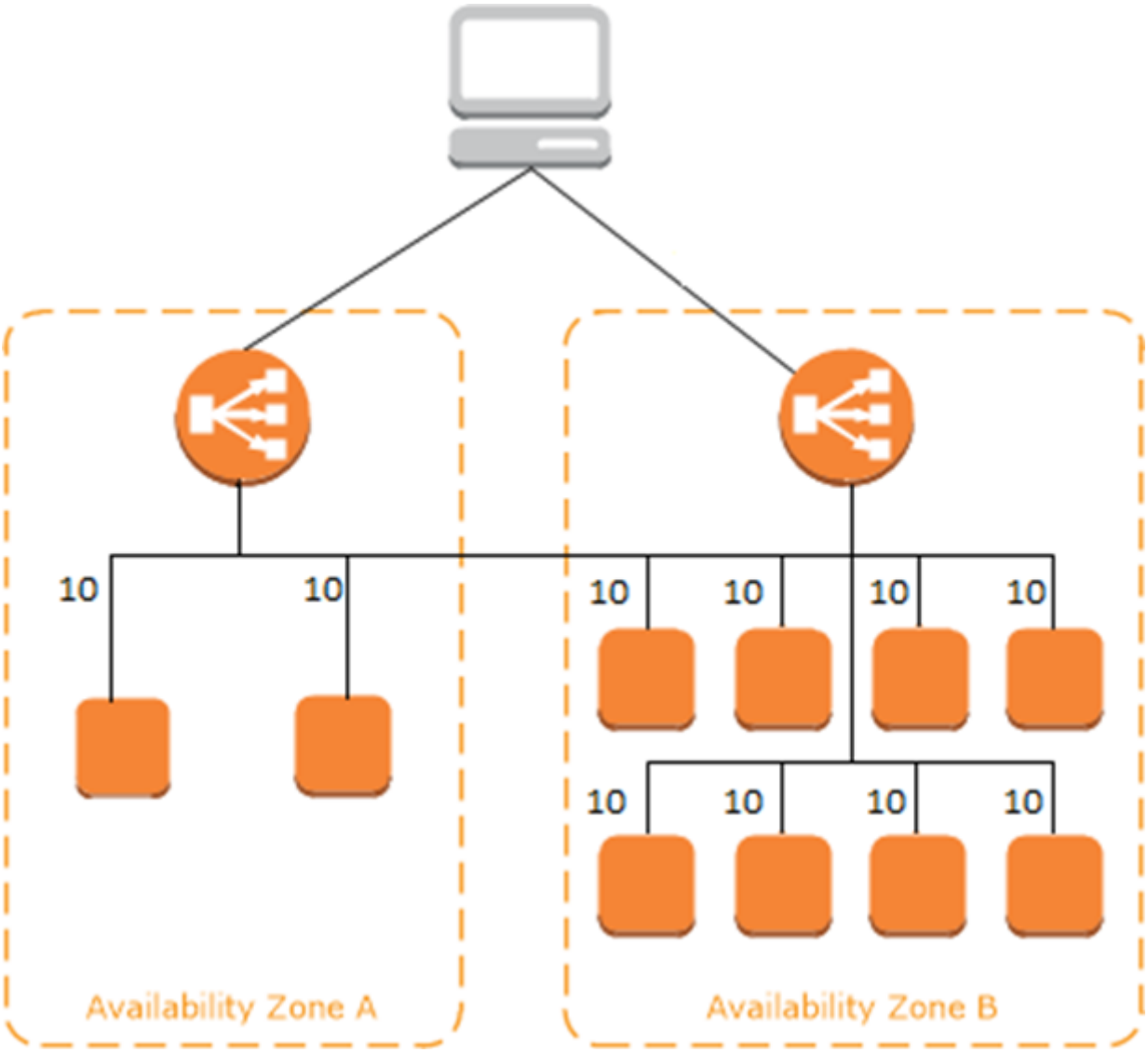
After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer. However, even though they remain registered, the load balancer does not route traffic to them.

Cross-zone load balancing

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

The following diagrams demonstrate the effect of cross-zone load balancing. There are two enabled Availability Zones, with two targets in Availability Zone A and eight targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes. This distributes traffic such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.

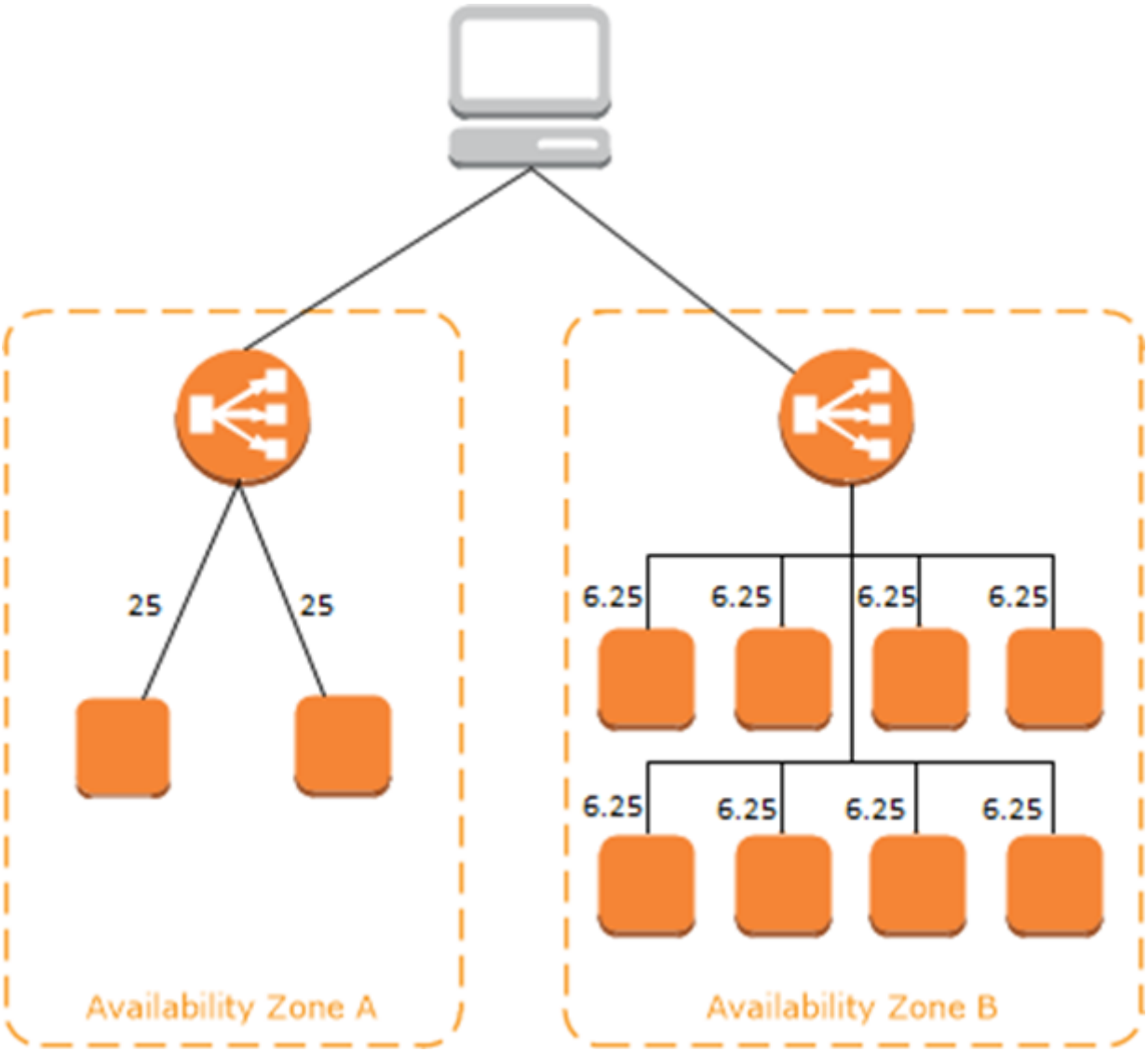
If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled:

- Each of the two targets in Availability Zone A receives 25% of the traffic.
- Each of eight targets in Availability Zone B receives 6.25% of the traffic.

This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.



With Application Load Balancers, cross-zone load balancing is always enabled.

With Network Load Balancers and Gateway Load Balancers, cross-zone load balancing is disabled by default. After you create the load balancer, you can enable or disable cross-zone load balancing at any time.

When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer. With the API or CLI, cross-zone load balancing is disabled by default. With the AWS Management Console, the option to enable cross-zone load balancing is selected by default. After you create a Classic Load Balancer, you can enable or disable cross-zone load balancing at any time. For more information, see Enable cross-zone load balancing in the *User Guide for Classic Load Balancers*.

Request routing

Before a client sends a request to your load balancer, it resolves the load balancer's domain name using a Domain Name System (DNS) server. The DNS entry is controlled by Amazon, because your load balancers are in the amazonaws.com domain. The Amazon DNS servers return one or more IP addresses to the client. These are the IP addresses of the load balancer nodes for your load balancer. With Network Load Balancers, Elastic Load Balancing creates a network interface for each Availability Zone that you enable. Each load balancer node in the Availability Zone uses this network interface to get a static IP address. You can optionally associate one Elastic IP address with each network interface when you create the load balancer.

As traffic to your application changes over time, Elastic Load Balancing scales your load balancer and updates the DNS entry. The DNS entry also specifies the time-to-live (TTL) of 60 seconds. This helps ensure that the IP addresses can be remapped quickly in response to changing traffic.

The client determines which IP address to use to send requests to the load balancer. The load balancer node that receives the request selects

a healthy registered target and sends the request to the target using its private IP address.

Routing algorithm

With **Application Load Balancers**, the load balancer node that receives the request uses the following process:

1. Evaluates the listener rules in priority order to determine which rule to apply.
2. Selects a target from the target group for the rule action, using the routing algorithm configured for the target group. The default routing algorithm is round robin. Routing is performed independently for each target group, even when a target is registered with multiple target groups.

With **Network Load Balancers**, the load balancer node that receives the connection uses the following process:

1. Selects a target from the target group for the default rule using a flow hash algorithm. It bases the algorithm on:
    - The protocol
    - The source IP address and source port
    - The destination IP address and destination port
    - The TCP sequence number
2. Routes each individual TCP connection to a single target for the life of the connection. The TCP connections from a client have

different source ports and sequence numbers, and can be routed to different targets.

With **Classic Load Balancers**, the load balancer node that receives the request selects a registered instance as follows:

- Uses the round robin routing algorithm for TCP listeners
- Uses the least outstanding requests routing algorithm for HTTP and HTTPS listeners

HTTP connections

Classic Load Balancers use pre-open connections, but Application Load Balancers do not. Both Classic Load Balancers and Application Load Balancers use connection multiplexing. This means that requests from multiple clients on multiple front-end connections can be routed to a given target through a single backend connection. Connection multiplexing improves latency and reduces the load on your applications. To prevent connection multiplexing, disable HTTP keep-alives by setting the Connection: close header in your HTTP responses.

Application Load Balancers and Classic Load Balancers support pipelined HTTP on front-end connections. They do not support pipelined HTTP on backend connections.

Application Load Balancers support the following protocols on front-end connections: HTTP/0.9, HTTP/1.0, HTTP/1.1, and HTTP/2. You can use HTTP/2 only with HTTPS listeners, and can send up to 128 requests in parallel using one HTTP/2 connection. Application Load Balancers also support connection upgrades from HTTP to Web Sockets.

However, if there is a connection upgrade, Application Load Balancer listener routing rules and AWS WAF integrations no longer apply.

Application Load Balancers use HTTP/1.1 on backend connections (load balancer to registered target) by default. However, you can use the protocol version to send the request to the targets using HTTP/2 or gRPC. For more information, see Protocol versions. Keep-alive is supported on backend connections by default. For HTTP/1.0 requests from clients that do not have a host header, the load balancer generates a host header for the HTTP/1.1 requests sent on the backend connections. The host header contains the DNS name of the load balancer.

Classic Load Balancers support the following protocols on front-end connections (client to load balancer): HTTP/0.9, HTTP/1.0, and HTTP/1.1. They use HTTP/1.1 on backend connections (load balancer to registered target). Keep-alive is supported on backend connections by default. For HTTP/1.0 requests from clients that do not have a host header, the load balancer generates a host header for the HTTP/1.1 requests sent on the backend connections. The host header contains the IP address of the load balancer node.

HTTP headers

Application Load Balancers and Classic Load Balancers automatically add **X-Forwarded-For**, **X-Forwarded-Proto**, and **X-Forwarded-Port** headers to the request.

Application Load Balancers convert the hostnames in HTTP host headers to lower case before sending them to targets.

For front-end connections that use HTTP/2, the header names are in lowercase. Before the request is sent to the target using HTTP/1.1, the following header names are converted to mixed case: **X-Forwarded-For**, **X-Forwarded-Proto**, **X-Forwarded-Port**, **Host**, **X-Amzn-Trace-Id**, **Upgrade**, and **Connection**. All other header names are in lowercase.

Application Load Balancers and Classic Load Balancers honor the connection header from the incoming client request after proxying the response back to the client.

When Application Load Balancers and Classic Load Balancers receive an **Expect** header, they respond to the client immediately with an HTTP 100 Continue without testing the content length header, remove the **Expect** header, and then route the request.

HTTP header limits

The following size limits for Application Load Balancers are hard limits that cannot be changed.

HTTP/1.x headers

- Request line: 16 K
- Single header: 16 K
- Whole header: 64 K

HTTP/2 headers

- Request line: 16 K

- Single header: 16 K
- Whole header: 64 K

Load balancer scheme

When you create a load balancer, you must choose whether to make it an internal load balancer or an internet-facing load balancer. Note that when you create a Classic Load Balancer in EC2-Classic, it must be an internet-facing load balancer.

The nodes of an internet-facing load balancer have public IP addresses. The DNS name of an internet-facing load balancer is publicly resolvable to the public IP addresses of the nodes. Therefore, internet-facing load balancers can route requests from clients over the internet.

The nodes of an internal load balancer have only private IP addresses. The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes. Therefore, internal load balancers can only route requests from clients with access to the VPC for the load balancer.

Both internet-facing and internal load balancers route requests to your targets using private IP addresses. Therefore, your targets do not need public IP addresses to receive requests from an internal or an internet-facing load balancer.

If your application has multiple tiers, you can design an architecture that uses both internal and internet-facing load balancers. For example, this is true if your application uses web servers that must be

connected to the internet, and application servers that are only connected to the web servers. Create an internet-facing load balancer and register the web servers with it. Create an internal load balancer and register the application servers with it. The web servers receive requests from the internet-facing load balancer and send requests for the application servers to the internal load balancer. The application servers receive requests from the internal load balancer.

Network MTU for your load balancer

The maximum transmission unit (MTU) of a network connection is the size, in bytes, of the largest permissible packet that can be passed over the connection. The larger the MTU of a connection, the more data that can be passed in a single packet. Ethernet packets consist of the frame, or the actual data you are sending, and the network overhead information that surrounds it. Traffic sent over an internet gateway is limited to 1500 MTU. This means that if packets are over 1500 bytes, they are fragmented, or they are dropped if the Don't Fragment flag is set in the IP header.

The MTU size on an Application Load Balancer, Network Load Balancer, or Classic Load Balancer node is not configurable. Jumbo frames (MTU 9001) are standard across all load balancer nodes. The path MTU is the maximum packet size that is supported on the path between the originating host and the receiving host. Path MTU Discovery (PMTUD) is used to determine the path MTU between two devices. Path MTU Discovery is especially important if the client or target does not support jumbo frames.

When a host sends a packet that is larger than the MTU of the receiving host or larger than the MTU of a device along the path, the receiving host or device drops the packet, and then returns the following ICMP message: Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (Type 3, Code 4). This instructs the transmitting host to split the payload into multiple smaller packets, and retransmit them.

If packets larger than the MTU size of the client or target interface continue to be dropped, it is likely that Path MTU Discovery (PMTUD) is not working. To avoid this, ensure that Path MTU Discovery is working end to end, and that you have enabled jumbo frames on your clients and targets. For more information about Path MTU Discovery and enabling jumbo frames, see Path MTU Discovery in the *Amazon EC2 User Guide*.

For more click on:

https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html

What is Amazon Cloud Watch?

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications, and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money.

With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.

Accessing CloudWatch

You can access CloudWatch using any of the following methods:

- **Amazon CloudWatch console** – https://console.aws.amazon.com/cloudwatch/
- **AWS CLI** – For more information, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
- **CloudWatch API** – For more information, see the Amazon CloudWatch API Reference.
- **AWS SDKs** – For more information, see Tools for Amazon Web Services.

For more click on:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

What Is AWS CloudTrail?

AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.

CloudTrail is enabled on your AWS account when you create it. When activity occurs in your AWS account, that activity is recorded in a

CloudTrail event. You can easily view recent events in the CloudTrail console by going to Event history. For an ongoing record of activity and events in your AWS account, create a trail. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

Visibility into your AWS account activity is a key aspect of security and operational best practices. You can use CloudTrail to view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure. You can identify who or what took which action, what resources were acted upon, when the event occurred, and other details to help you analyze and respond to activity in your AWS account. Optionally, you can enable AWS CloudTrail Insights on a trail to help you identify and respond to unusual activity.

You can integrate CloudTrail into applications using the API, automate trail creation for your organization, check the status of trails you create, and control how users view CloudTrail events.

For more click on:

https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html

What is Amazon S3?

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive,

enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

Features of Amazon S3

Storage classes

Amazon S3 offers a range of storage classes designed for different use cases. For example, you can store mission-critical production data in S3 Standard for frequent access, save costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA, and archive data at the lowest costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.

You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers optimized for frequent and infrequent access, and two opt-in archive access tiers designed for asynchronous access for rarely accessed data.

For more information, see Using Amazon S3 storage classes. For more information about S3 Glacier Flexible Retrieval, see the *Amazon S3 Glacier Developer Guide*.

Storage management

Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.

- S3 Lifecycle – Configure a lifecycle policy to manage your objects and store them cost effectively throughout their lifecycle. You can transition objects to other S3 storage classes or expire objects that reach the end of their lifetimes.
- S3 Object Lock – Prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use Object Lock to help meet regulatory requirements that require *write-once-read-many (WORM)* storage or to simply add another layer of protection against object changes and deletions.
- S3 Replication – Replicate objects and their respective metadata and object tags to one or more destination buckets in the same or different AWS Regions for reduced latency, compliance, security, and other use cases.
- S3 Batch Operations – Manage billions of objects at scale with a single S3 API request or a few clicks in the Amazon S3 console. You can use Batch Operations to perform operations such as **Copy**, **Invoke AWS Lambda function**, and **Restore** on millions or billions of objects.

Access management

Amazon S3 provides features for auditing and managing access to your buckets and objects. By default, S3 buckets and the objects in them are

private. You have access only to the S3 resources that you create. To grant granular resource permissions that support your specific use case or to audit the permissions of your Amazon S3 resources, you can use the following features.

- S3 Block Public Access – Block public access to S3 buckets and objects. By default, Block Public Access settings are turned on at the account and bucket level.
- AWS Identity and Access Management (IAM) – Create IAM users for your AWS account to manage access to your Amazon S3 resources. For example, you can use IAM with Amazon S3 to control the type of access a user or group of users has to an S3 bucket that your AWS account owns.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- Access control lists (ACLs) – Grant read and write permissions for individual buckets and objects to authorized users. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM policies for access control instead of ACLs. ACLs are an access control mechanism that predates resource-based policies and IAM. For more information about when you'd use ACLs instead of resource-based policies or IAM policies, see Access policy guidelines.
- S3 Object Ownership – Disable ACLs and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. You, as the bucket owner, automatically own and have full control over every object in

your bucket, and access control for your data is based on policies.

- Access Analyzer for S3 – Evaluate and monitor your S3 bucket access policies, ensuring that the policies provide only the intended access to your S3 resources.

Data processing

To transform data and trigger workflows to automate a variety of other processing activities at scale, you can use the following features.

- S3 Object Lambda – Add your own code to S3 GET requests to modify and process data as it is returned to an application. Filter rows, dynamically resize images, redact confidential data, and much more.
- Event notifications – Trigger workflows that use Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), and AWS Lambda when a change is made to your S3 resources.

Storage logging and monitoring

Amazon S3 provides logging and monitoring tools that you can use to monitor and control how your Amazon S3 resources are being used. For more information, see Monitoring tools.

Automated monitoring tools

- Amazon CloudWatch metrics for Amazon S3 – Track the operational health of your S3 resources and configure billing alerts when estimated charges reach a user-defined threshold.
- AWS CloudTrail – Record actions taken by a user, a role, or an AWS service in Amazon S3. CloudTrail logs provide you with detailed API tracking for S3 bucket-level and object-level operations.

Manual monitoring tools

- Server access logging – Get detailed records for the requests that are made to a bucket. You can use server access logs for many use cases, such as conducting security and access audits, learning about your customer base, and understanding your Amazon S3 bill.
- AWS Trusted Advisor – Evaluate your account by using AWS best practice checks to identify ways to optimize your AWS infrastructure, improve security and performance, reduce costs, and monitor service quotas. You can then follow the recommendations to optimize your services and resources.

Analytics and insights

Amazon S3 offers features to help you gain visibility into your storage usage, which empowers you to better understand, analyze, and optimize your storage at scale.

- Amazon S3 Storage Lens – Understand, analyze, and optimize your storage. S3 Storage Lens provides 29+ usage and activity

metrics and interactive dashboards to aggregate data for your entire organization, specific accounts, AWS Regions, buckets, or prefixes.

- Storage Class Analysis – Analyze storage access patterns to decide when it's time to move data to a more cost-effective storage class.
- S3 Inventory with Inventory reports – Audit and report on objects and their corresponding metadata and configure other Amazon S3 features to take action in Inventory reports. For example, you can report on the replication and encryption status of your objects. For a list of all the metadata available for each object in Inventory reports, see Amazon S3 Inventory list.

Strong consistency

Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your Amazon S3 bucket in all AWS Regions. This behavior applies to both writes of new objects as well as PUT requests that overwrite existing objects and DELETE requests. In addition, read operations on Amazon S3 Select, Amazon S3 access control lists (ACLs), Amazon S3 Object Tags, and object metadata (for example, the HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model.

How Amazon S3 works

Amazon S3 is an object storage service that stores data as objects within buckets. An *object* is a file and any metadata that describes the file. A *bucket* is a container for objects.

To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a *key* (or *key name*), which is the unique identifier for the object within the bucket.

S3 provides features that you can configure to support your specific use case. For example, you can use S3 Versioning to keep multiple versions of an object in the same bucket, which allows you to restore objects that are accidentally deleted or overwritten.

Buckets and the objects in them are private and can be accessed only if you explicitly grant access permissions. You can use bucket policies, AWS Identity and Access Management (IAM) policies, access control lists (ACLs), and S3 Access Points to manage access.

Buckets

A bucket is a container for objects stored in Amazon S3. You can store any number of objects in a bucket and can have up to 100 buckets in your account. To request an increase, visit the Service Quotas Console.

Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `DOC-EXAMPLE-BUCKET` bucket in the US West (Oregon) Region, then it is addressable using the URL `https://DOC-EXAMPLE-BUCKET.s3.us-west-`

2.amazonaws.com/photos/puppy.jpg. For more information, see Accessing a Bucket.

When you create a bucket, you enter a bucket name and choose the AWS Region where the bucket will reside. After you create a bucket, you cannot change the name of the bucket or it's Region. Bucket names must follow the bucket naming rules. You can also configure a bucket to use S3 Versioning or other storage management features.

Buckets also:

- Organize the Amazon S3 namespace at the highest level.
- Identify the account responsible for storage and data transfer charges.
- Provide access control options, such as bucket policies, access control lists (ACLs), and S3 Access Points that you can use to manage access to your Amazon S3 resources.
- Serve as the unit of aggregation for usage reporting.

For more information about buckets, see Buckets overview.

Objects

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object. These pairs include some default metadata, such as the date last modified, and standard HTTP metadata, such as Content-Type. You can also specify custom metadata at the time that the object is stored.

An object is uniquely identified within a bucket by a key (name) and a version ID (if S3 Versioning is enabled on the bucket). For more information about objects, see Amazon S3 objects overview.

Keys

An *object key* (or *key name*) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, object key, and optionally, version ID (if S3 Versioning is enabled for the bucket) uniquely identify each object. So you can think of Amazon S3 as a basic data map between "bucket + key + version" and the object itself.

Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, key, and optionally, a version. For example, in the URL https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg, `DOC-EXAMPLE-BUCKET` is the name of the bucket and `/photos/puppy.jpg` is the key.

For more information about object keys, see Creating object key names.

S3 Versioning

You can use S3 Versioning to keep multiple variants of an object in the same bucket. With S3 Versioning, you can preserve, retrieve, and restore every version of every object stored in your buckets. You can easily recover from both unintended user actions and application failures.

For more information, see using versioning in S3 buckets.

Version ID

When you enable S3 Versioning in a bucket, Amazon S3 generates a unique version ID for each object added to the bucket. Objects that already existed in the bucket at the time that you enable versioning have a version ID of null. If you modify these (or any other) objects with other operations, such as CopyObject and PutObject, the new objects get a unique version ID.

For more information, see using versioning in S3 buckets.

Bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size.

Bucket policies use JSON-based access policy language that is standard across AWS. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies allow or deny requests based on the elements in the policy, including the requester, S3 actions, resources, and aspects or conditions of the request (for example, the IP address used to make the request). For example, you can create a bucket policy that grants cross-account permissions to upload objects to an S3 bucket while ensuring that the bucket owner has full control of the uploaded objects. For more information, see Bucket policy examples.

In your bucket policy, you can use wildcard characters on Amazon Resource Names (ARNs) and other values to grant permissions to a subset of objects. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as .html.

Access control lists (ACLs)

You can use ACLs to grant read and write permissions to authorized users for individual buckets and objects. Each bucket and object has an ACL attached to it as a sub resource. The ACL defines which AWS accounts or groups are granted access and the type of access. ACLs are an access control mechanism that predates IAM. For more information about ACLs, see Access control list (ACL) overview.

By default, when another AWS account uploads an object to your S3 bucket, that account (the object writer) owns the object, has access to it, and can grant other users access to it through ACLs. You can use Object Ownership to change this default behavior so that ACLs are disabled and you, as the bucket owner, automatically own every object in your bucket. As a result, access control for your data is based on policies, such as IAM policies, S3 bucket policies, virtual private cloud (VPC) endpoint policies, and AWS Organizations service control policies (SCPs).

A majority of modern use cases in Amazon S3 no longer require the use of ACLs, and we recommend that you disable ACLs except in unusual circumstances where you need to control access for each object individually. With Object Ownership, you can disable ACLs and rely on policies for access control. When you disable ACLs, you can easily maintain a bucket with objects uploaded by different AWS accounts.

You, as the bucket owner, own all the objects in the bucket and can manage access to them using policies. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

S3 Access Points

Amazon S3 Access Points are named network endpoints with dedicated access policies that describe how data can be accessed using that endpoint. Access Points simplify managing data access at scale for shared datasets in Amazon S3. Access Points are named network endpoints attached to buckets that you can use to perform S3 object operations, such as `GetObject` and `PutObject`.

Each access point has its own IAM policy. You can configure Block Public Access settings for each access point. To restrict Amazon S3 data access to a private network, you can also configure any access point to accept requests only from a virtual private cloud (VPC).

For more information, see Managing data access with Amazon S3 access points.

Regions

You can choose the geographical AWS Region where Amazon S3 stores the buckets that you create. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. Objects stored in an AWS Region never leave the Region unless you explicitly transfer or replicate them to another Region. For example, objects stored in the Europe (Ireland) Region never leave it.

What is Amazon S3 File Gateway

AWS Storage Gateway connects an on-premises software appliance with cloud-based storage to provide seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure. You can use the service to store data in the AWS Cloud for scalable and cost-effective storage that helps maintain data security. AWS Storage Gateway offers file-based, volume-based, and tape-based storage solutions.

Topics

- Amazon S3 File Gateway

Amazon S3 File Gateway

Amazon S3 File Gateway –Amazon S3 File Gateway supports a file interface into Amazon Simple Storage Service (Amazon S3) and combines a service and a virtual software appliance. By using this combination, you can store and retrieve objects in Amazon S3 using industry-standard file protocols such as Network File System (NFS) and Server Message Block (SMB). The software appliance, or gateway, is deployed into your on-premises environment as a virtual machine (VM) running on VMware ESXi, Microsoft Hyper-V, or Linux Kernel-based Virtual Machine (KVM) hypervisor. The gateway provides access to objects in S3 as files or file share mount points. With a S3 File, you can do the following:

- You can store and retrieve files directly using the NFS version 3 or 4.1 protocol.

- You can store and retrieve files directly using the SMB file system version, 2 and 3 protocol.
- You can access your data directly in Amazon S3 from any AWS Cloud application or service.
- You can manage your S3 data using lifecycle policies, cross-region replication, and versioning. You can think of a S3 File as a file system mount on Amazon S3.

A S3 File simplifies file storage in Amazon S3, integrates to existing applications through industry-standard file system protocols, and provides a cost-effective alternative to on-premises storage. It also provides low-latency access to data through transparent local caching. A S3 File manages data transfer to and from AWS, buffers applications from network congestion, optimizes and streams data in parallel, and manages bandwidth consumption. S3 File integrate with AWS services, for example with the following:

- Common access management using AWS Identity and Access Management (IAM)
- Encryption using AWS Key Management Service (AWS KMS)
- Monitoring using Amazon CloudWatch (CloudWatch)
- Audit using AWS CloudTrail (CloudTrail)
- Operations using the AWS Management Console and AWS Command Line Interface (AWS CLI)
- Billing and cost management

In the following documentation, you can find a Getting Started section that covers setup information common to all gateways and also gateway-specific setup sections. The Getting Started section shows you how to

deploy, activate, and configure storage for a gateway. The management section shows you how to manage your gateway and resources:

- Provides instructions on how to create and use a S3 File. It shows you how to create a file share, map your drive to an Amazon S3 bucket, and upload files and folders to Amazon S3.
- Describes how to perform management tasks for all gateway types and resources.

In this guide, you can primarily find how to work with gateway operations by using the AWS Management Console. If you want to perform these operations programmatically, see the *AWS Storage Gateway API Reference.*

For more click on:
https://docs.aws.amazon.com/filegateway/latest/files3/what-is-file-s3.html

What Is Amazon S3 Glacier?

Amazon S3 Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.

With S3 Glacier, customers can store their data cost effectively for months, years, or even decades. S3 Glacier enables customers to offload the administrative burdens of operating and scaling storage to AWS, so they don't have to worry about capacity planning, hardware provisioning, data replication, hardware failure detection and recovery,

or time-consuming hardware migrations. For more service highlights and pricing information, go to the S3 Glacier detail page.

S3 Glacier is one of the many different storage classes for Amazon S3

For more click on:
https://docs.aws.amazon.com/amazonglacier/latest/dev/introduction.html

What is Amazon Relational Database Service (Amazon RDS)?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

For more click on:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

What is AWS Lambda?

Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the languages that Lambda supports.

You organize your code into Lambda functions. Lambda runs your function only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

You can invoke your Lambda functions using the Lambda API, or Lambda can run your functions in response to events from other AWS services. For example, you can use Lambda to:

- Build data-processing triggers for AWS services such as Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.
- Process streaming data stored in Amazon Kinesis.
- Create your own backend that operates at AWS scale, performance, and security.

Lambda is a highly available service. For more information, see the AWS Lambda Service Level Agreement.

Sections

- When should I use Lambda?
- Lambda features
- Getting started with Lambda
- Related services
- Accessing Lambda
- Pricing for Lambda

When should I use Lambda?

Lambda is an ideal compute service for many application scenarios, as long as you can run your application code using the Lambda standard runtime environment and within the resources that Lambda provides.

When using Lambda, you are responsible only for your code. Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you cannot log in to compute instances or customize the operating system on provided runtimes. Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.

If you need to manage your own compute resources, AWS has other compute services to meet your needs. For example:

- Amazon Elastic Compute Cloud (Amazon EC2) offers a wide range of EC2 instance types to choose from. It lets you customize operating systems, network and security settings, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.

- AWS Elastic Beanstalk enables you to deploy and scale applications onto Amazon EC2. You retain ownership and full control over the underlying EC2 instances.

Lambda features

The following key features help you develop Lambda applications that are scalable, secure, and easily extensible:

Concurrency and scaling controls

Concurrency and scaling controls such as concurrency limits and provisioned concurrency give you fine-grained control over the scaling and responsiveness of your production applications.

Functions defined as container images

Use your preferred container image tooling, workflows, and dependencies to build, test, and deploy your Lambda functions.

Code signing

Code signing for Lambda provides trust and integrity controls that let you verify that only unaltered code that approved developers have published is deployed in your Lambda functions.

Lambda extensions

You can use Lambda extensions to augment your Lambda functions. For example, use extensions to more easily integrate Lambda with your favorite tools for monitoring, observability, security, and governance.

Function blueprints

A function blueprint provides sample code that shows how to use Lambda with other AWS services or third-party applications. Blueprints include sample code and function configuration presets for Node.js and Python runtimes.

Database access

A database proxy manages a pool of database connections and relays queries from a function. This enables a function to reach high concurrency levels without exhausting database connections.

File systems access

You can configure a function to mount an Amazon Elastic File System (Amazon EFS) file system to a local directory. With Amazon EFS, your function code can access and modify shared resources safely and at high concurrency.

Getting started with Lambda

To work effectively with Lambda, you need coding experience and expertise in the following domains:

- Linux OS and commands, as well as concepts such as processes, threads, and file permissions.
- Cloud concepts and IP networking concepts (for public and private networks).
- Distributed computing concepts such as HTTP as an IPC, queues, messaging, notifications, and concurrency.
- Familiarity with security services and concepts: AWS Identity and Access Management (IAM) and access control principles, and AWS Key Management Service (AWS KMS) and public key infrastructure.
- Familiarity with key services that interact with Lambda: Amazon API Gateway, Amazon S3, Amazon Simple Queue Service (Amazon SQS), and DynamoDB.
- Configuring EC2 instances with Linux.

If you are a first-time user of Lambda, we recommend that you start with the following topics to help you learn the basics:

1. Read the Lambda product overview and explore the Lambda getting started page.
2. To create and test a Lambda function using the Lambda console, try the console-based getting started exercise. This exercise teaches you about the Lambda programming model and other concepts.

3. If you are familiar with container image workflows, try the getting started exercise to create a Lambda function defined as a container image.

AWS also provides the following resources for learning about server less applications and Lambda:

- The AWS Compute Blog includes useful articles about Lambda.
- AWS Server less provides blogs, videos, and training related to AWS serverless development.
- The AWS Online Tech Talks YouTube channel includes videos about Lambda-related topics. For an overview of serverless applications and Lambda, see the Introduction to AWS Lambda & Serverless Applications video.

Related services

Lambda integrates with other AWS services to invoke functions based on events that you specify. For example:

- Use API Gateway to provide a secure and scalable gateway for web APIs that route HTTP requests to Lambda functions.
- For services that generate a queue or data stream (such as Dynamo DB and Kinesis), Lambda polls the queue or data stream from the service and invokes your function to process the received data.

- Define Amazon S3 events that invoke a Lambda function to process Amazon S3 objects, for example, when an object is created or deleted.
- Use a Lambda function to process Amazon SQS messages or Amazon Simple Notification Service (Amazon SNS) notifications.
- Use AWS Step Functions to connect Lambda functions together into serverless workflows called state machines.

Accessing Lambda

You can create, invoke, and manage your Lambda functions using any of the following interfaces:

- **AWS Management Console** – Provides a web interface for you to access your functions. For more information, see Lambda console.
- **AWS Command Line Interface (AWS CLI)** – Provides commands for a broad set of AWS services, including Lambda, and is supported on Windows, macOS, and Linux. For more information, see Using Lambda with the AWS CLI.
- **AWS SDKs** – Provide language-specific APIs and manage many of the connection details, such as signature calculation, request retry handling, and error handling. For more information, see AWS SDKs.
- **AWS Cloud Formation** – Enables you to create templates that define your Lambda applications. For more information, see AWS Lambda applications. AWS Cloud

Formation also supports the AWS Cloud Development Kit (CDK).

- **AWS Serverless Application Model (AWS SAM)** – Provides templates and a CLI to configure and manage AWS serverless applications. For more information, see AWS SAM.

REACH US

www.microdegree.work

hello@microdegree.work

+91 83108 82795

MicroDegree ಕನ ಡ - YouTube

https://t.me/microdegreekannada

MicroDegree