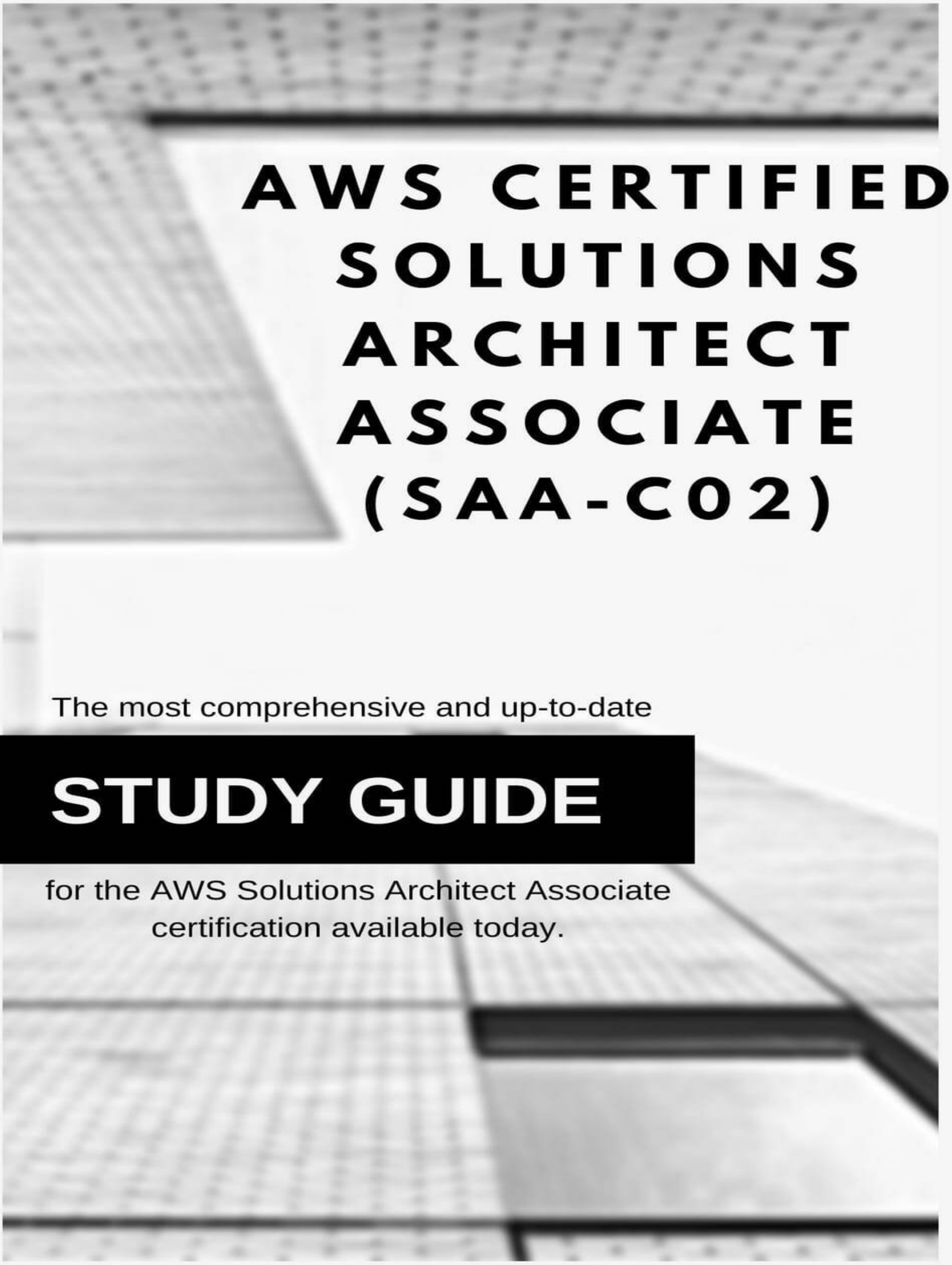


AWS CERTIFIED SOLUTIONS ARCHITECT ASSOCIATE (SAA-C02)

The most comprehensive and up-to-date

STUDY GUIDE

for the AWS Solutions Architect Associate certification available today.

NEAL SANFORD, ALLISON COPE, & MICHAEL REEVES.

AWS CERTIFIED SOLUTIONS ARCHITECT ASSOCIATE (SAA-C02)

The most comprehensive and up-to-date study guide for the AWS Solution Architect Associate certification available today.

Neal Sanford,
Allison Cope, &
Michael Reeves.

PREFACE

Amazon Web Services (AWS) is a collection of web services offered by Amazon that allow third-party developers to connect and develop on the company's technology infrastructure. This suite provides a range of infrastructure services that can be used to supplement or bypass the conventional physical infrastructure that web applications need. Anyone with an Amazon.com account and a credit card can use these web services, which include storage, computing power, a communication system, a payment system, and a database. The best part is that you only pay for what you use for these services. You rent just the infrastructure you need and just when you need it, whether you are only trying out the services or using them as a platform for online applications that serve thousands of users. You can focus on developing your application and growing your company instead of worrying about infrastructure management.

The aim of this book is to provide readers with introductory details, technologies, applications development, security, and most of the advanced topics in AWS cloud computing. This is a comprehensive foundational book that can benefit both beginners and intermediate practitioners. This book is written as an AWS Solutions Architect-Associate Certification study guide for those who want to check, validate, and effectively demonstrate their skills for designing solutions safely and robustly. It is also useful for cloud service providers who want to complement their own customer and developer training programs with a wider view of cloud computing.

In this book, we will look at the suite of web services provided by Amazon that have pay-as-you-go, virtual infrastructure as well as the practicalities of developing and creating applications that take advantage of these Amazon's infrastructure services. We will go into how the services can be used independently and in combination to build real-world systems. The book is organized into ten chapters. The key concepts, data models, behavior, and use cases of Amazon's core and advanced services are explained in nine chapters, with a comprehensive cheat sheet in the final chapter.

We start this book with a general overview of Amazon Web services and then discuss AWS's fundamentals in a wider context. Each type of service in this textbook is categorized under a domain (categorized as chapters). All the widely used domains are Compute, Storage, Database, Management and Monitoring, Network and Content Delivery, AWS Security, and Application integration. These chapters describe the features and capabilities of each service and provide an in-depth explanation of the operations exposed by the service.

In order to provide a detailed explanation of the AWS concepts, this book includes comprehensive AWS concepts, real-time use cases, and architectural diagrams. With the help of these AWS real-life examples, you can learn to create and manage application infrastructure effectively. New topics have been introduced to cover additional topics for the latest SAA-C02 exam blueprint. Cheat sheets are the book's final section that contains specific points for each topic to recall for the exam.

Why this book? The book is well-structured, which starts with foundational concepts and then quickly moves to build complex solutions. Focusing on learning core areas first and then move on to other topics will give a direction to your learning and make it easier. Although the book is primarily written for clearing certification exams, you can use it as a reference. It explains the core principles of AWS very well and provides insights on computing, storage, and networking fundamentals. You can also jump to any chapter or cheat sheet as per your requirement. However, moving from one chapter to another sequentially will help you gain more knowledge and concepts.

This book explains all key exam topics with thorough explanations and expert insights. It also provides a lot of good and relatable examples for each concept. The comprehensive Cheat sheets for each topic are the best part of this book that you will not get anywhere. The book is ideal for beginners who have absolutely no idea about AWS and learn it from scratch, but in-depth. If you have a working knowledge or know about AWS products, it might be a slow read, and you can skip some sections. For newbies, Amazon's AWS documentation is not much clear, and the same information is presented in a neat and easy-to-understand manner by the author. The author’s tone is easy to understand, and you will get the feeling of ‘oh, that’s all AWS is about?’ while reading.

This book is not an endorsement of the products or services of Amazon, Google, or Microsoft, nor is any portion of our work supported financially (or otherwise) by these vendors. All trademarks and products belong to their respective owners, and the underlying principles and approaches, we believe, apply to other cloud vendors as well.

The compilation of this book has been possible because of the efforts and contributions of many individuals. Firstly, the editors would like to thank all contributors for their immense hard work in putting together excellent chapters that are informative, comprehensive, rigorous, and, above all, timely. The editors would like to thank the team led by *Neal Sanford, Allison Cope, & Michael Reeves* for patiently helping us put this book together. Lastly, we would like to thank our families for their support throughout this journey and dedicate this book to them.

We believe you will find the book informative, concise, comprehensive, and helpful to gain AWS cloud knowledge.

Table of contents

INTRODUCTION

AWS is the top leading cloud services provider, which has modified and revolutionized our IT approach. AWS is the hallmark of many top global organizations, such as Netflix, Ubisoft, Spotify, and Expedia. AWS offers all the advantages of high data availability and durability that an individual might expect from a cloud service. AWS is a much better choice than other cloud service providers because it helps their users focus more on their projects and customers without stressing about infrastructure problems. It does not just save the user's efforts, time, and money but also helps them focus on the important business aspects.
AWS rules over the cloud computing market, making AWS certification a required certification for IT experts working with the IT Giants. Presently, AWS certificates are considered the most famous and high-in-demand certifications in the IT sector. These certifications are suitable for individuals interested to become Cloud engineers or Cloud architects. These certifications show a certified individual's ability to develop and operate tech solutions on Amazon's cloud platform. They are consistently ranked as the world's highest-paid information technology certifications. The salary of an IT certified professional can be increased by 26% with the AWS Cloud certifications.

One of the most popular and most valuable IT Certifications is AWS Solution Architect Associate. AWS SAA is considered a common entry point for people joining cloud architecture. An AWS Solution Architect is a person with a profound knowledge of critical cloud engineering concepts to efficiently design and implement reliable and scalable cloud technology solutions. With AWS Solution Architect Certification, you can understand, learn, assess, and demonstrate your ability to design and install secure and stable AWS platform applications. The AWS-certified Solution Architect can earn an average annual salary of $130,000, which is 11.7% higher than the non-certified individuals. So, if you have this certification in your resume, you will be most likely get a higher salary, and you will be high in demand by many employers.

This textbook is one of the in-depth and exam-specific tools specially made to prepare individuals for the AWS Solutions Architect Associate (SAA-CO2) certification exam. It covers all the important AWS services, both basic and newly added elements, needed across all the AWS domains. We use the process of incremental learning to build a strong base of theoretical knowledge for our learners. By the end of this course, you will surely be equipped with enough knowledge to pass the certification exam.

Details of Certification Exam

This *Certification Exam* is intended for the individuals devoted to the Solution Architect's role. Normally, it identifies the participant's ability to develop a solution based on architectural design concepts, which focuses on the clients' needs and helps them execute the proposed solution for their project.

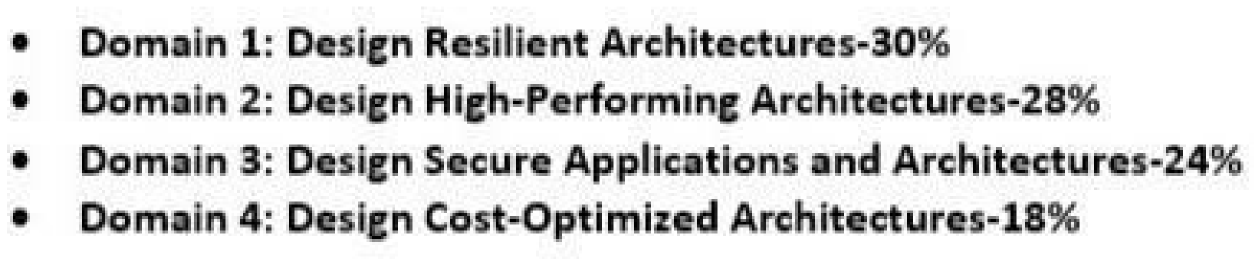
You will need an AWS exam guide to getting all the details and the exam's objectives after you have made your mind to take the test. The official guide is available on the official website of AWS. This certification also includes two levels of a solutions architect, Associate level and Professional level.

Important exam details are listed below:

- The exam includes 65 questions.
- The passing percentage is 65% (720 out of 1000 points).
- The exam duration is 130 minutes.
- This test consists of two types of questions, including multiple answers and multiple-choice questions.
- It is offered in various languages, including Chinese, Korean, English, and Japanese.
- This certification is valid for two years.
- *AWS Solution Architect Associate-level* exam costs $150, *whereas the Professional-level* exam costs $300.
- The latest version of the exam is released in March 2020, referred to as AWS SAA-CO2.

To demonstrate your AWS knowledge in the exam, you will be tested across four different domains. You will be examined through four different domains and each with a total percentage of your overall performance. While proceeding along the course, you will be able to fully understand each domain and gain the requisite knowledge to meet the domain requirements.

These domains are categorized as:



- **Domain 1: Design Resilient Architectures-30%**
- **Domain 2: Design High-Performing Architectures-28%**
- **Domain 3: Design Secure Applications and Architectures-24%**
- **Domain 4: Design Cost-Optimized Architectures-18%**

Note

There is no single standardized material to pass the Solutions Architect exam. So relying on a single resource is not a wise decision. To pass the exam on your first go, we suggest that you gather the following material and start preparing at least three months before the actual exam:

• 2-3 comprehensive and up-to-date study guides,

• AWS whitepapers, and FAQs,

• Test your knowledge via a set of AWS practice exams available on the AWS official website,

• A free tier AWS account to practice exercises by following the instructions, and,

• Last but not the least, cheat sheets for revising all the specific points before sitting in the exam.

Intended Audience

This learning approach is ideal for individuals who want to pass the AWS SAA-CO2 Certification exam on their first go by acquiring in-depth knowledge of all of the AWS services required to become an AWS Solutions Architect. As stated in the blueprint for the AWS SAA exam, it is particularly intended for those who can check, validate, and effectively demonstrate their skills for designing solutions safely and robustly.

The most suitable match for this textbook is the beginners who want to get familiar with the AWS infrastructure and want to enhance their skills. They do not need any previous AWS knowledge or programming expertise to start this course. With this textbook's help, one will be able to create the base for the exam preparation even though he is never signed on to the AWS platform before. The Network Administrators and existing Solution Architects can also use this edition to learn more about the AWS cloud platform's newly added components to add to their expertise. Furthermore, it can be used by the companies concerned to make their solution architects certified for hosting their applications that are entirely scalable and fault-tolerant onto the AWS cloud. Therefore, this approach can surely help you in preparing for the role of AWS Solutions Architect.

Key Learning

This textbook aims to validate your AWS knowledge in many key areas. This learning process enables you to understand all the key components with the help of detailed graphical explanations and real-life use case scenarios. In this way, you can easily grasp the concepts and principles of all components. Through this approach, you can:

1. Determine the right Amazon web service based on the customer requirements for computing, databases, security, and cost control mechanisms.
2. Learn to develop, deploy and maintain secure and flexible applications on the cloud with Amazon Web Services.
3. Get a detailed understanding of fault-tolerant, highly scalable, and highly available systems on AWS.
4. Identify the correct use of Amazon's most basic and newest web services.
5. New users can easily get familiar with the role of each component within the VPC architectural design.

Let us now look at the course outline.

About the Course

In order to pass the AWS SAA exam successfully and get the certification, it is important to demonstrate your ability to operate a variety of AWS services or tools to design and implement secure and robust applications. For that purpose, we start this course with a general overview of Amazon Web services and then discuss AWS's fundamentals in a wider context. In order to provide a detailed explanation of the AWS concepts, this textbook includes comprehensive AWS concepts, real-time use cases, and architectural diagrams. With the help of these AWS real-life examples, you can learn to create and manage application infrastructure effectively. New topics have been introduced to cover additional topics for the latest SAA-C02 exam blueprint.

Each type of service in this textbook is categorized under a domain; all the domains which are widely used are:

- Compute
- Storage
- Database
- Management and Monitoring
- Network and Content Delivery
- AWS Security
- Application integration

This book will focus on the key technology areas and services from the SAA-C02 exam perspective.

Specifically, we will look at ***Amazon EC2***, ***AWS Lambda,*** and ***Elastic Beanstalk*** in detail in compute domain.

For storage, we will see ***Amazon S3, EBS, EFS***, ***Storage Gateways,*** and ***AWS Snow Family***.

We will also explore ***Amazon RDS, Amazon Aurora, DynamoDB,*** and ***Redshift***.

For application monitoring and scaling, we will use ***Amazon CloudWatch, Cloud Trail,*** and ***Cloud Formation***.

For network and content delivery services, we will discuss ***Virtual Private Cloud, Route 53, Autoscaling Groups, ELB, Cloud Front***, and other networking technologies in full detail.

After that, we will examine how you can secure your environment in AWS and manage cost. For that, we will discuss ***AWS IAM, AWS KMS,*** and ***Amazon Cognito***.

Then we will move onto the application integration services module in which the Amazon messaging services such as ***Amazon SQS*** and ***SNS*** are to be discussed**.**

Finally, we will introduce you to some other important terminologies that are considered important for the certification exam. Let us list down the important terminologies this chapter covers, such as:

AWS Management Console,

AWS Development tools,

AMI,

API Gateway,

Amazon kinesis,

Elastic Cache,

Elastic Container Service,

Elastic Fabric Adapter,

Elastic Network Adapter and

the concepts of ***High Availability,***

Vertical & horizontal Scaling.

This chapter also covers some of the newly introduced AWS terminologies like

High-Performance Computing,

Amazon FSx,

AWS Resource Access Manager, and

AWS Cloud Migration.

By the end of this textbook, a comprehensive free ***AWS Cheat-Sheet*** will key highlight the fundamentals of AWS, which is perfect for quick revision of the most important topics before you sit for your exam. You will then be able to take this certification exam with full confidence.

Now, let us get started with the new journey to become Solution Architect Associate!

CHAPTER 1: CLOUD COMPUTING WITH AWS

Overview

The very first chapter of the textbook introduces the very commonly used Amazon *Web Services*. This chapter gives you an insight into the overview of the AWS Cloud and the benefits of using AWS. Along with the basic services that make up the platform, it also covers the life cycle of the AWS Cloud computing platform. Let us get started with our first chapter.

Introduction to AWS

AWS is a globally distributed cloud network operated by Amazon that hosts and manages services on the internet. Currently, it provides more than 150 services, and every new week the number is growing. Almost every service is available on request, and most of them are immediately accessible. Large enterprises, media companies, e-commerce websites, and pharmaceutical research firms prefer to use AWS to handle their data perfectly and grow their business without any hassle as it offers many services like IaaS, PaaS, and SaaS. In AWS, IaaS means that Amazon provides the servers as a facility to its users so that the facility's backup and power supply need not be handled. It provides a platform as a service so that Java, Ruby, and Php can be used as services, and binaries for such applications do not have to be managed. Then, users can have Software as a Service, which enables them to get mailing abilities such as *SES* and queuing services such as SQS. On top of Amazon's IaaS, you can develop and execute applications in the cloud. It follows the PAYG type payment method without capital expenses at the beginning of the service usage. So, you will pay less for using fewer services and more for a lot of services (but still less per unit). Moreover, all web services are available to users in a matter of seconds. On the whole, AWS is a cloud service provider that offers many services to run user applications.

Different Companies use and get benefits from the services that Amazon provides. Some famous companies include *Adobe, Airbnb, Netflix, Autodesk, Bitdefender, Docker, BMW group, COMCAST, ESA, Spotify, HTC, IMDb, McDonald's, NASA,* and *Kellogg's*. This huge popularity and customer values are just a small portion of the much larger and unseen facility. If a thought arises, why so many people prefer using AWS, then AWS would answer this question: People are adopting AWS because of security, durability, storage facilities, end-to-end privacy, and encryption of data. You can trust the AWS way of managing '*Things'* with the help of AWS tools, methods, and suggested guidelines based on several years of industrial experience.

Flexibility: The huge flexibility in AWS lets you choose your desired OS language and database.

Usability: You can use AWS for hosting your applications instantly, beat a new application, or move an existing application in AWS with its considerable speed and quick deployment.

Scalability: Considering the user requirements, any application can be easily scaled up or scale down.

Cost-effectiveness: The payment is only for your computing capacity, storage, and other services you use, and there are no long-term undertakings.

Today, Amazon has a strong lead over its competitors, such as *Google Cloud* and *Microsoft Azure*. Currently, millions of companies and individual users are using Amazon Web Services for their workloads. Therefore, providing Amazon web services worldwide is probably one of Amazon's best business decisions.

Lifecycle of the AWS Cloud Computing Solution

Before we learn more about Amazon Web Services, let us quickly look at the lifecycle of the AWS cloud solution as it will help you understand the AWS cloud infrastructure more clearly. Getting a *Correct Understanding of Requirements* is the first step in the lifecycle of Amazon's cloud computing solution. After that, the appropriate service offered by the provider will then be chosen. The second step would be to *Define the Hardware* after a fine understanding of your requirements. Defining hardware is selecting a compute service that provides the desired support so that the computing capacity can be resized to run applications. Knowing the exact requirement can also help you in choosing the right hardware.

As not all fit one size so, different services and hardware for different user requirements are available in the AWS marketplace. You can use *Elastic Cloud Compute* for IaaS, *Lambda* for serverless computing, and *ECS* for containerized service. You need to select the right hardware according to your requirements.

Now, the third step is *Defining storage*. Select the relevant service for storage to backup your data and a separate storage service archiving the data either locally within the cloud or on the internet. You have to pick up the right storage option, such as *Simple Storage Service (S3)*, especially for backups**.** There is separate storage to archive data, known as *Amazon Glacier*. Therefore, recognizing the difference between them help you pick the best service for your desired requirement. You can see all these services in the following diagram:

EC2
Lambda
Elastic Container Service
S3
EFS
Glacier
VPC
Route 53
Direct Connect
IAM
KMS
Cognito
Cloudwatch
Autoscaling
CloudFormation
CodeStar
CodeBuild
CodePipeline
Athena
EMR
CloudSearch

Next is to *Define the Network*, which delivers data, videos, and applications securely. You have to define the network services and identify them correctly. For instance, you use *Virtual Private Cloud* for Networking, *Direct Connect* for private connection of user’s office to the AWS center, and *Route-53* for Domain Name System. The next step is to establish the *Right Security Services*; for example, *IAM* is used to authenticate and authorize the access while *KMS* is used for data encryption at rest. Therefore, several security services are accessible, so you must choose the correct one according to the requirements.

Then the next step is to *Define the Management Tools and Processes*. Here you can choose from various deployment, automation, and monitoring tools available in the AWS platform. CloudWatch is a monitoring service, while Auto-scaling is an automation service, and CloudFormation is a deployment service. If you identify the management tools properly to monitor the AWS resources and all customized applications on the AWS platform, you can efficiently control your cloud environment. Understanding them would also help you to identify the cloud computing solution's life cycle properly.

Furthermore, many resources are available for *Testing a Process*, such as *CodeStar*, *CodeBuild*, and *CodePipeline*. These are tools that allow you to build, test, and deploy your code quickly. Finally, once everything has been set and done, select the *Analytics Services* to analyze and visualize the data. You can start to query the data immediately and get a response from the analytic services. The Amazon Athena can be used to view the events in your region visually. The *EMR-Elastic Map Reduce* and *CloudSearch are the* other analytical services. So, this is a brief introduction to the lifecycle of the AWS cloud platform. Let us move onto the overview of some basic Amazon web services.

Overview of AWS

AWS is the Cloud Computing technology operated by Amazon, which is available over the internet. Let us now discuss the various types of services provided by AWS. The services we describe here can be classified into categories, such as compute, storage, database, networking, security, management, monitoring services, and application integration services. If you are looking for a service that is not available, these services may be included in the certifications for Big Data or DevOps Engineer. Now, let us discuss a few AWS services that are frequently used for SAA Certification.

Within *Compute Services,* we have Amazon EC2, Amazon Lambda, and Elastic Beanstalk.

Amazon EC2: In cloud services, Amazon EC2 offers to compute capacity. It provides security and can be resized according to the end-users needs.

AWS Lambda: This refers to a compute service that enables users to execute the code without using and managing another service. AWS Lambda will only run the code when necessary and scales it from fewer requests a day to thousands a second. Moreover, the users will only be charged for the computing time that they consume.

Elastic Beanstalk: An Application Orchestration service that allows its users to scale and launch Web applications. Different programming languages are used for their development. A user needs to upload the code. It will then automatically monitor the whole deployment process and the application's performance as well.

Within *Storage Services,* Amazon provides S3, *Glacier, Amazon EBS, and EFS.*

Amazon S3: S3 is generally file storage and sharing service used for uploading and sharing files. It is the object storage that can store and recover data from any platform such as websites, mobile applications, IoT sensors, etc.

Amazon Glacier: This cloud storage service is safe, long-lasting, and exceptionally cost-effective that is used for data storage and permanent backups.

Amazon EBS: This service offers block store volumes for the EC2 instances. EBS service is highly accessible, and it provides secure storage volume, which can be connected to working EC2 instances in a similar AZ. The EBS volumes-attached instances are shown as Storage Volumes that remain independent of the instance's lifetime.

Amazon EFS: It can be used with the EC2 instance as a flexible cloud-based storage solution. It provides a simpler interface with the help of quickly generating and customizing file systems with ease. Amazon file system is designed to scale on users' demand elastically without affecting the application from expanding and shrinking.

Now let us talk about *Database Services in AWS.* We have two main flavors of databases in AWS: Amazon RDS and Amazon Redshift.

Amazon RDS*:* RDS lets you run and organize a database in the cloud. They have nearly every database from SQL Server to Oracle, MySQL PostgreSQL, and Aurora. RDS supports the process involved in the creation, operation, and scaling of a relational database. It offers a cost-effective and resizable capacity as it automates administrative activities like providing hardware, Database setups, repairs, and data backup, which require a considerable amount of time.

Amazon Redshift: It provides a data storage service for analyzing data through SQL and other smart corporate tools. With the help of advanced query optimization against the petabytes of structured information, you can even run complex analytical queries. Moreover, most of the results are generated within seconds.

In the *Management and Monitoring Services,* we have CloudWatch, CloudTrail, and CloudFormation.

Cloud Watch: It is used for tracking applications and resources that run on the AWS platform. It is used Cloud Watch provides two types of monitoring that are Basic monitoring and Detailed monitoring.

Cloud Trail: CloudTrail is a web service in the AWS account that monitors your API operation. It is a very effective monitoring tool that can enable users to track logs, control, and maintain actions-related account activity within the AWS network.

Cloud Formation: It is a service that models and configures AWS resources, allowing you to concentrate more on your application and spend less time handling those AWS resources. In cloud formation, different templates are used to generate user applications instantly. Moreover, infrastructure is defined as code here.

AWS provides *Networking & Content Delivery* services such as Virtual Private Cloud, Route 53, Elastic load balancer, and Cloud Front.

VPC**:** With Virtual Private Cloud, you can set up virtual networks in the cloud for running your servers in those networks. You are not allowed to have full control over the cloud network; instead, you can have VPCs, which are the chunks of your cloud.

Route 53: This Networking and Content delivery service has high availability and is a fully managed DNS service. Being a scalable Domain Name System, it is fully compatible with IPV6 and scales according to the requirement.

Elastic Load Balancer: It is used for network and content delivery, distributing incoming requests to multiple locations, including containers, EC2 instances, and IP addresses. By using ELB, the fluctuating load of your application traffic can be handled in both single and multiple Availability Zones.

Cloud Front: Cloud Front falls under the content delivery network domain that boosts the categorization process of Static and Dynamic Web Content for end-users.

Then in *AWS SECURITY*, we have IAM service and KMS.

AWS IAM: IAM is a security tool that allows its end-users to get and manage the Amazon web services and resources safely. With the help of IAM, creating and managing users, groups, and permissions to accept or reject the AWS resource access has become very easy.

KMS: This service creates and manages the user data encryption keys very easily. The HSMs are the Hardware Security Modules responsible for securing the customer master keys generated in KMS. AWS KMS uses a highly secure encryption method (AES 256-bit encryption) to access the user’s data.

Finally, we have *Application Integration Services* which includes SQS and SNS.

SQS**:** It is a completely managed queuing service that acts as a bridge of communication and connects isolated applications with the help of messaging and queues. It uses a queue, which is temporary storage for messages that need to be processed.

SNS**:** The SNS is a fully managed and highly available pub/sub messaging service which allows you to decouple microservices-based architectures, serverless and distributed systems. When we say decoupling, we refer to application integration which is like a family of AWS services connecting one service to another.

This was a quick overview of a few of the primary Amazon web services. We will be covering a lot more terminologies in the course ahead.

AWS provides 175 services that fall under various domains, but we only focus on the ones important for the Solutions Architect Associate exam preparation. These domains include Compute, Storage, Databases, Network & Content Delivery Services, Security, Management, & Monitoring Services, and Application Integration Services. Let us get started with the basic ones.

CHAPTER 2: COMPUTE FUNDAMENTALS

Overview

AWS Compute Services are designed for capacity provisioning according to the user's demand. There is the problem of under-provision or over-provision the compute resources when using the cloud. AWS Compute Domain provides a solution by providing a set of computing services that are scalable, highly available, and customizable over the cloud to meet customer-specific needs. This section provides all the necessary elements and concepts of AWS Compute services, which can assist you in selecting the right service for your applications and projects.

Before we start, let us define the compute services. To support your business, you will need a compute capacity to run your applications. In a traditional environment, one needs to determine the required compute capacity first, then purchase the hardware to support that capacity, and finally, deploy all the servers to run the user applications. When you launch your application onto these servers, you need to maintain them from the software as well as physical perspective. On the other hand, if you build cloud-native applications, you can use compute as a service model, which allows you to provide and consume raw computing capacity over the internet with the pay-as-use model. It will eliminate the stress of installation and maintenance of these physical servers and allows you to use desired hardware and software to run over them. In addition to the maintenance and installation of computer infrastructure in a typical on-premises environment, when you miscalculate the initial capacity estimation, then there are two possibilities. In the case of limited resources, the user will face slow application performance, and the latency will result in user dissatisfaction that can affect your business. So, to overcome this issue, you will need to buy more servers and install, configure, and maintain those physical servers. While in the case of excessive resources, you will be paying for idle resources and bear the extra cost. Therefore, AWS enables you to provide the required capacity on user's demand and manage the entire process of installation, configuration along with the maintenance of virtual server for clients.

AWS Compute domain encompasses a variety of compute services. Amazon EC2 or Elastic Compute Cloud is the fundamental compute service of Amazon with customizable cloud computing capacity. It offers complete control over computing resources and enables deployment on the cloud environment of Amazon. AWS provides serverless solutions such as Lambda. The flexibility of AWS compute services allows you to execute all the applications in the cloud virtually. It provides container services as well, which enable us to use Docker via Elastic Container Service (ECS) or Kubernetes via EKS. Then there are managed compute options in AWS, such as Amazon Lightsail. You can use this service for the desired compute capacity without knowing the provision or management of the underlying hardware. Furthermore, there are a few options that extend beyond raw server capacity. In this section, you will see some commonly used fundamental compute services from an exam point of view. Regarding compute services, the Amazon EC2, Lambda, and Elastic Beanstalk will be covered in this chapter.

AWS EC2 (ELASTIC CLOUD COMPUTE)

Introduction

EC2, abbreviated as *Elastic Compute Cloud*, is the first and commonly used Amazon web service, which provides empty servers. It is used to launch a virtual machine and run the application on it. Based on the user's criteria, all types of virtual machines are available here. It is a web-based service that allows developers to deliver secure and resizable cloud computing capacity easily. It makes it easy to scale up or scale down the user’s infrastructure based on the requirement. For instance, let us say that the web traffic requirement that comes to your website keeps changing. Therefore, behind the scenes, EC2 can enlarge its workspace to several instances, and when there is no load, it can reduce the workspace to one resource. It follows a pay-as-you-go pricing model, and almost all Amazon web services get well integrated with this EC2 service. To configure new instances, EC2 uses Amazon Machine Images. AMI’s are the software and application packages that you need to run your application. It provides the necessary information to configure an EC2 instance.

Now, we will look at the steps for an EC2 instance creation.

How to create it?

For creating an instance, you have to follow these seven steps:

- *AMI* Selection
- *Instance Type* Selection

- Instance Configuration
- Adding *Storage*
- Adding *Tags*
- *Security Group* Configuration
- Review

First of all, an AWS account is required, and then an EC2 instance for compute capacity. After that, you can follow the steps to create it. First, you have to create an AMI, which is the software and application packages required to run our application. It is a pattern that helps to generate new instances, new devices, new virtual machines based on user needs. The software, the operating system, and the additional applications get installed in the AMI. The second step is to select the hardware, which is called the instance type. The instance type is the hardware specification required to construct a machine. You will select the hardware according to the workload and hardware size according to the strength of the workload.

Now, you need to configure the instance. For that, you have to specify the instance number, buying options, the network type, the subnet, the IP address assignment, the IAM role, and the shutdown behavior. There are various purchasing options that are available under the configure instance, such as reserved instances, spot instances, or on-demand instances, and you can select according to your requirement. A Public IP address and an IAM role can be assigned to an EC2 instance when it is being created here. You can also choose the shutdown behavior of the EC2 instance, which will stop the instance when the device or instance is shut down from your desktop. So, as the name says, ‘Configure instance,’ there is a great deal of instance configuration options available here.

An instance can be bootstrapped with some scripts under the *‘advanced details’* or the *‘advanced’ tab* in the configuration instance. Now, bootstrap is the script that a user wants to execute in the instance until it comes online. Let us assume that you have to provision the instance for someone else. So you can create a bootstrap shell script and paste it in the console option in the configure instance, instead of you launching the instance by yourself, then logging in and running some commands and then passing it to the other individual. After that, Amazon will take those commands and execute them before passing them on to the user who requested them in the first place. Now it can be a different user or maybe just you; it will automatically start the software installation process in the instance that you are going to initialize. So, when configuring the instance, all this data goes in here. Note that the first three steps include the OS volume and the basic hardware.

Now in step four, it is time to add extra storage to the EC2 instance. Here, you have multiple storage options. One of them is ephemeral storage, which is free, and the other is external storage known as Elastic Block Storage, which is paid. Moreover, the EC2 instance can be integrated with S3 for its storage requirements. Free subscription for users is the best thing about storage. You can use 30 gigabit SSD or magnetic storage throughout the year, but you must mention size in gigabit and volume type here. The type of volume may be a provision volume, a general-purpose volume, or a magnet volume. You will also provide information about where the disc is being installed and whether this volume needs encryption or not. All these inputs are provided by the user under the 'adding storage' section.

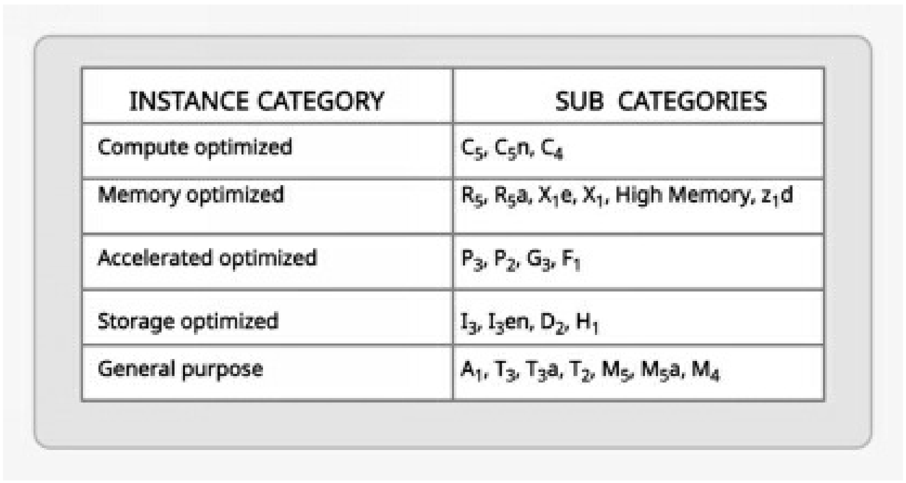
Next, you configure tags for an EC2 instance to specify them easily at upcoming stages. Tags are very useful in identifying a machine in such an environment where users run thousands of virtual machines. You give some meaningful terms to an instance so that you can specify the instance type, billing options, and the purpose of launching that instance. In an environment where 700 or 800 instances are running, it could be a full-time job to identify an instance, the purpose of its creation, and its ownership. Therefore, tagging helps in such cases.

After tagging, you configure the Firewall, a security group for an EC2 instance. Security groups are known as a firewall because they provide a shield to EC2 instance by protecting them from unwanted internal and external traffic. Here, the EC2 instance can be tuned and accessed based on port number and IP address. At this point, you will accept or reject external connections for this specific EC2 instance.

And then, at step seven, you Review the configurations that have been made and ensure that the configuration is satisfactory. Before the EC2 instance is launched, the Amazon console will show an option for creating a key pair. There are two types of key pairs: One is public, and the other is private. The user keeps the private key by downloading it, whereas Amazon uses the Public Key to confirm the user identity. Therefore, the user should download and save the private key. It is downloaded as a *.pem* file, which is a file format. Then, accessing the EC2 instance is our next step. If you have launched a Linux instance, then a tool called *PuTTY* is required to access the instance. PuTTY is used to access Linux instances from windows instances. It rarely happens that window instance lacks PuTTY; otherwise, it usually has PuTTY installed in it. So, in such scenarios, the PuTTY and PuTTY key generator can be downloaded. PuTTY key generator is a tool that can change the *.pem* file into a *.ppk* file. In this way, a user can launch EC2 instances successfully.

Instance types

Let us discuss the *Instance Types* and their applications. There are five broad categories, and then there are specialized subcategories/families of instances. These Instance types can be shown in the following table.



INSTANCE CATEGORY	SUB CATEGORIES
Compute optimized	C_5, C_5n, C_4
Memory optimized	R_5, R_5a, X_1e, X_1, High Memory, z_1d
Accelerated optimized	P_3, P_2, G_3, F_1
Storage optimized	I_3, I_3en, D_2, H_1
General purpose	A_1, T_3, T_3a, T_2, M_5, M_5a, M_4

The first one is *Compute Optimized*, which offers a lot of computing/processing power. So, if an application requires excessive processing power, then this family is perfect for such applications. Its use cases are game servers, ad server engines, and scientific modeling. The second category is *Memory Optimized*. It is perfect for applications that need in-memory caches. Some applications are compatible with caching. Those applications would generate a lot of data that a user wants to store in the cache for re-reading or lengthy processing. For such applications, this memory-optimized instance with an in-memory cache is an excellent use case. They are useful for data analytics, in-memory caching, and in-memory databases.

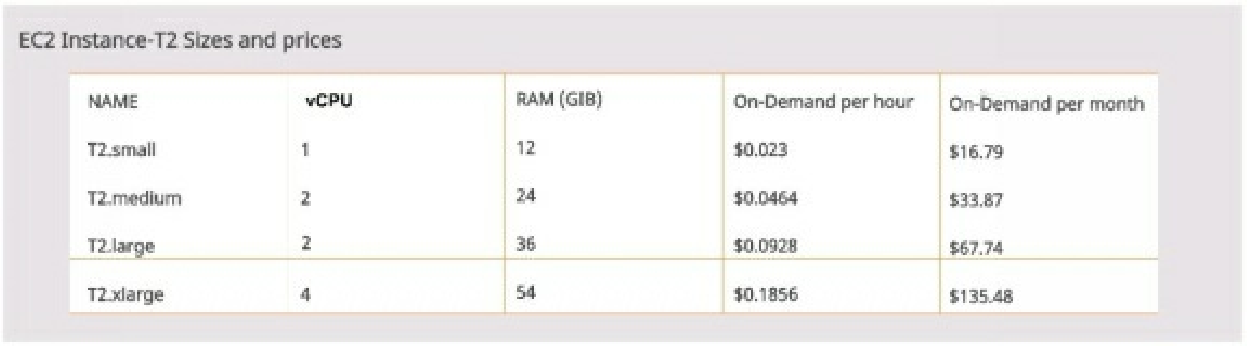
The third family is *Accelerated-Optimized*. Instances of this family use co-processors. They will be perfect for ML, computational finance, voice recognition, and pushover analysis. Moreover, Future technologies would make use of accelerated optimized instances. Then we have the *Storage Optimized*, which is the fourth category. As the name suggests, this family is more suitable for storage servers like NoSQL database, in-memory or transactional databases, and data warehousing. The fifth family is *General Purpose*, and it is used for general purpose applications. Normally, when users launch an EC2 instance, it will almost always be in the *T2* or *T3* general category. If you are not particularly concerned about the family, you could choose the general-purpose category because the services are evenly balanced. It offers a balance between virtual CPU, memory, storage, and network performance. In general purpose, all the components required for a computer are equally balanced. They are suitable for web servers and code repositories.

One key point to remember is that these five instance families are hardware-based, so they are static and cannot be changed. Moreover, users do not have much control over hardware because they purchase the hardware. Now, let us move onto the instance sizes, instance profiles, and other important terminologies to discuss them in a bit more detail.

Instance Sizes

There are different sizes for each subcategory or family of EC2 instances. The table below shows the example of a T2 type of instance that shows small, medium, large, and extra-large sizes. The instance sizing works in a way that a user usually gets double the previous one. So when you move from small to medium size, it doubles the RAM and the CPU. On the other hand, the RAM and the on-demand prices get doubled for medium and large sizes. So users get twice in size every time the price gets high. Therefore, the standard rule is to go with the next version of the instance types if the user wants an update or increment in instance size.

EC2 instance-T2 Sizes and prices



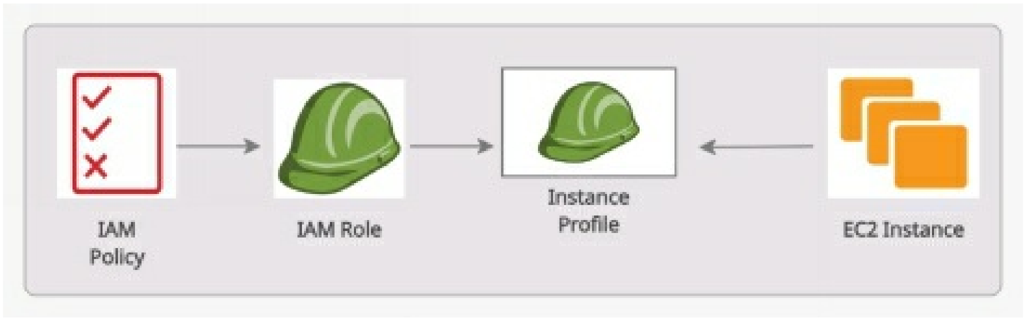
EC2 Instance-T2 Sizes and prices

NAME	vCPU	RAM (GIB)	On-Demand per hour	On-Demand per month
T2.small	1	12	$0.023	$16.79
T2.medium	2	24	$0.0464	$33.87
T2.large	2	36	$0.0928	$67.74
T2.xlarge	4	54	$0.1856	$135.48

Features of Amazon EC2

Instance Profiles

The *Instance Profile* is used to get permission for the EC2 instances, which is like a container for IAM roles. So it is used to attach a role to an instance instead of inserting your AWS credentials into your code. The basic idea is that there is an instance profile for the EC2 instance and the role that has permissions. So you should never insert AWS credentials whenever you are given the opportunity not to insert them. There are two ways to set up an instance profile, for instance. If you go through the wizard, you will see the IAM Role here. So, an IAM role needs to be created and then attached to the instance. As it is a kind of an invisible step, users often skip instance profile. However, if you use the console, it will automatically create the instance profile, but you would have to create it if you do this programmatically through Cloud Formation.



Placement Groups

We are going to take a look here at *Placement Groups***.** The placement groups are used to select the instances' logical placement to enhance communication, performance, and durability. It is a free and optional service. A user does not necessarily need to launch an instance in the placement group. However, if he does, there are some advantages depending on the use case.

Clusters

Let us discuss clusters now. *Clusters* are used to pack the instances inside the Availability Zones tightly. In the case of a tightly coupled node to node communication, clusters are ideal for low latency network performance. Furthermore, when the servers are close to each other, then communication becomes very fast. Clusters cannot be multi-AZ, but they are ideal for high-performance computing applications.

Partitions

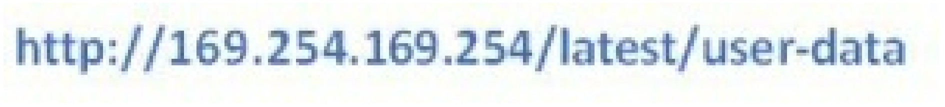
Partitions are used to spread the instances across the logical partitions. However, the underlying hardware is not shared among these partitions as each partition they are running on individual racks. They are ideal for widespread and repetitive workloads such as *Hadoop*, *Cassandra,* and *Kafka*. These technologies use logical partitions, and users have physical partitions, so they can use their instances very effectively by combining both.

Spread

When each instance is placed on a different rack, it is known as *Spread*. Spreads are used when there are critical instances that need to be isolated from each other. A user can spread a maximum of seven instances. Unlike clusters, the spread can be multi-AZ.

User Data

The *User Data* is a script that will automatically execute when EC2 is launched. It enables users to install packages or check updates before launching an instance. So, when you walk through the EC2 wizard, there is this advanced detailed step where you can provide your bash script. A specific URL is used to check the user data script performed on that EC2 instance on launch. If you use SSH protocol and CURL this particular URL, you can see the user data script within the EC2 instance.



```
http://169.254.169.254/latest/user-data
```

Metadata

It is the additional information about the EC2 instance that a user gets at run time. If a user *SSH* into the EC2 instance and runs this CURL command with the latest metadata, he will get all the information. The information users get, such as current public IP address, *AMI-ID*, and the instance type. So, you can use the bash script if you want to do it programmatically. *Metadata* and user data scripts can be combined to perform every type of AWS staging automation. Thus, it is suitable and very useful for debugging.

EC2 pricing

Now, let us take a look at the EC2 pricing model, and there are four ways you can pay with EC2. They are On-demand, spot, reserved, and dedicated, and we are going to go through each section.

On-Demand Instances

First, we will look at *On-Demand Pricing*. When a user launches an EC2 instance, by default, it will use an on-demand instance. There is no advance payment and long-term commitment for on-demand instances. Depending on the EC2 instance type, a user is charged only by the hour or minute. The use-case of the on-demand instance is the applications where the users require short duration, uncertain and spiky workloads. If an application is under development or if a user wants to experiment, then on-demand is a perfect choice.

Reserved Instances

Let us discuss *Reserved Instances (RI)* in detail. They are designed for applications that need reserved capacity or have a stable, predictable consumption pattern. So, you have to inform AWS about using reserved instances for a certain time period, and they are going to offer discounts. Reserved instances provide the most exceptional long-term discounts. These discounted prices depend on three variables, including *Term, class offerings,* and *payment options.* We will go through these variables to understand how they work.

Class offerings include *standard, convertible,* and *scheduled* methods**.** *Standard* offers the highest savings, with 75% reduced prices as compared to the on-demand instances. Attributes are like instance types; therefore, a user is not allowed to change the RI attribute. If you want more flexibility as you might need more space to grow in the future, you can choose convertible. Discounts in the *Convertible* are 54%, which are not as good as standard, but now you can change the instance size to a bigger size. Size cannot be reduced; it will always increase, but there is flexibility in the convertible. Then there is *a scheduled* method that is used when a user needs to reserve an instance for a specific time. There may be a case where you have a regular workload for a few hours every Friday. Therefore, by informing AWS that you will do it on schedule, they will provide you discounts.

The other two variables are *Term* and *payment options*. A Term is the time duration in which a user is willing to sign an agreement, like one year or three years. Terms operate in the principle of" *the greater the Term, the higher the savings."* In the payment options, a user has *all upfront, partial upfront,* and *no upfront* payments. No upfront is the most attractive option. For example, you might say that you will use the server for one year, so you only pay at the end of every month, which is a decent way of saving money.

The reserved instances can also be shared between multiple accounts within a single organization. You can sell unused RIs in the AWS marketplace. If you buy a reserved instance, there is always a resell option so that someone in need might use it.

Spot Instances

Let us look at the *Spot Instances*. As compared to on-demand instances, they offer the biggest discount of 90% to their users. AWS has some excess compute capacities, so they want the maximum use of their unused servers. It is the same as if a hotel offers a discount to fill its vacant rooms or when an airline offers tickets on discounted rates to fill its vacant seats. So, these unused instances are just the EC2 instances that reside here. Therefore, it is better to offer discounted rates than to waste such instances.

There is also a drawback when using spot instances. If another customer decides to pay a higher on-demand price for an instance that you are already using, then this instance can be stopped at any time. AWS must provide this capacity to the new on-demand user. Thus, instances can be terminated at any time by AWS under some conditions. When the instance is terminated by AWS, you do not pay for the partial hour of use, and if you terminate an instance, you will pay for the hour that it ran. The spot instances are designed for applications with adjustable timings or applications that are possible at meager computing costs.

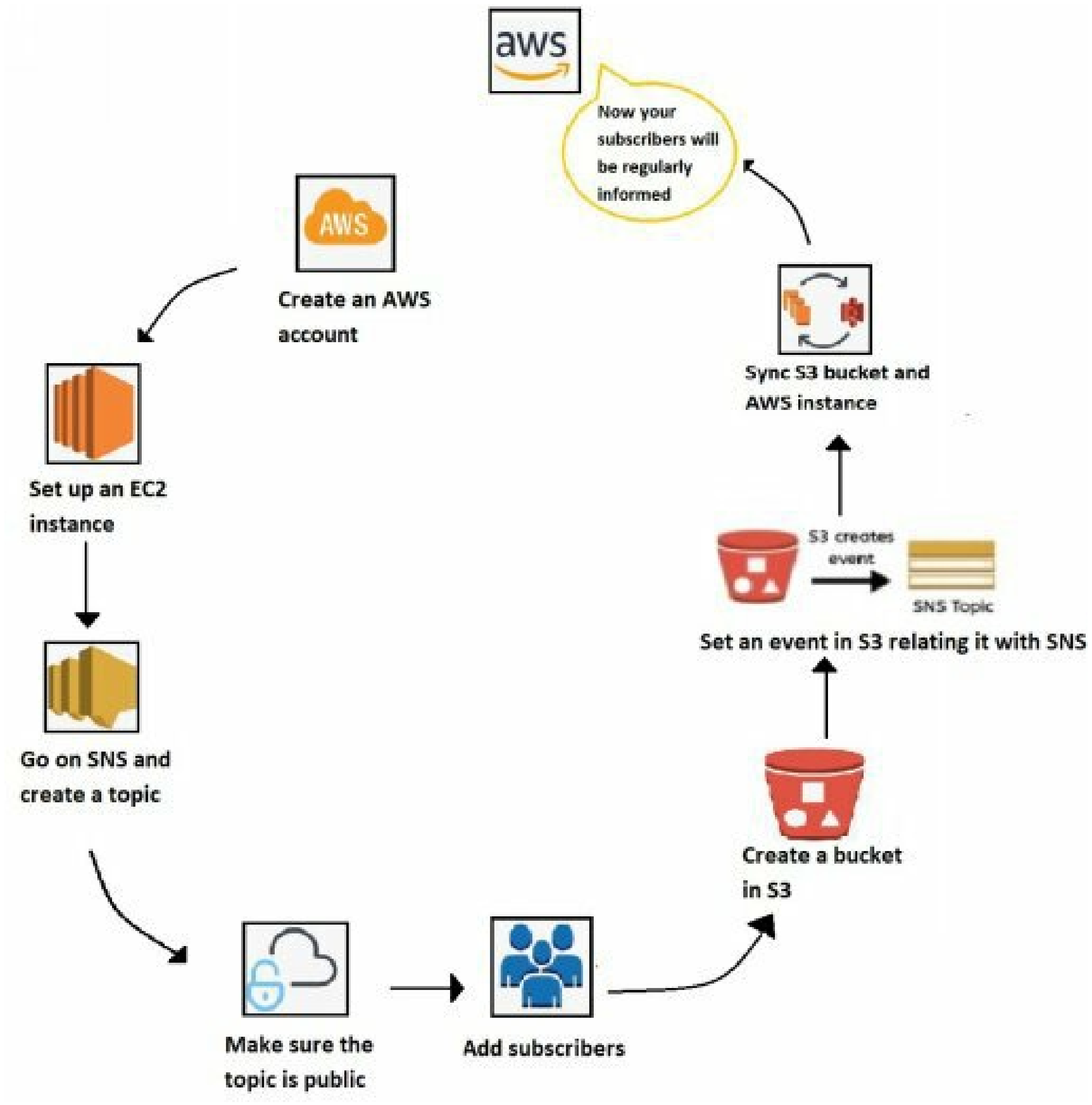
Dedicated Host Instances

The *Dedicated Host Instances* are the most expensive option in the EC2 pricing model and are meant to meet the regulatory criteria. The regulatory criteria are intended to be met where there is a strict server-bound licensing that does not allow multi-tenancy or cloud deployments. One must understand *Multi-Tenant* and *Single-Tenant* to understand a dedicated host instance. Whenever an EC2 instance is initiated, and any instance type other than the dedicated host is selected, it is multi-tenant. Multi-tenant means that you use the same hardware as other AWS customers, and the only distinction between you and other customers is through the software called *virtual isolation**.*** Single-Tenant means that a single user has unique hardware, and users are separated via *physical isolation**.*** For comparison, a multi-tenant is like an apartment for everyone, whereas a single-tenant is like a whole house for everyone. Large companies and businesses like to have their dedicated hard wares because they may have security issues or concerns for sharing the same hardware with other AWS customers. With dedicated hosts, it comes with an on-demand and reserved option. Therefore, a user can save up to 70% with this option, but a dedicated host is too expensive compared to other EC2 pricing options.

EC2-Use case

Let us discuss a use case in which a successful businessman has many users and a successful in-market product. Now, few more products are developed, which (he thinks) will be very successful in the future, and the customers will be satisfied with it. So, the businessman needs to figure out a practical marketing approach to get the customer's attention. This business use case can be accomplished using basic AWS services like EC2, SNS, S3 and then combining them all. Therefore, let us discuss how to get this environment started, essential measures to connect the environment, and make the applications start execution on it. In short, figure out the way to notify the users about a newsletter.

So, the team of a businessman can develop an architecture where whenever the company creates a newsletter, the group of users is informed about it. Here the components are AWS resources. Firstly, the architecture would need EC2 instance for computing, SNS for informing the users, and S3 for storage then, merge the EC2 and S3. That is all an infrastructure would require to inform a group of users about a new newsletter.


aws
Now your subscribers will be regularly informed
Create an AWS account
Set up an EC2 instance
Go on SNS and create a topic
Make sure the topic is public
Add subscribers
Create a bucket in S3
Set an event in S3 relating it with SNS
S3 creates event
SNS Topic
Sync S3 bucket and AWS instance

Let us discuss each step in detail now. So, first, his team needs to launch an EC2 instance. As discussed earlier, there are seven steps for launching an EC2 instance. By following these steps, they can successfully launch an EC2 instance. After that, they will move towards notifying the customers. *Simple Notification Service* is a service of Amazon which updates the customers through email. So, they need to search for SNS in the AWS account and create a *Topic* that will be used for public notifications. So, it will be made public, and the *Subscribe*r is also created. Now, these subscribers are the ones the team wants to send the newsletter information. Suppose that they already have an email database. So they can link this database with the subscriber list, and then new newsletters will be delivered to the subscribers when they publish the topic.

The next step is to create *a Bucket in S3*, where data will be stored. They will create an *Event* in that bucket, which will trigger and notify the *SNS*. In this way, the S3 bucket will be set up, and subscribers will get notifications if anything is added to the S3 bucket. S3 bucket will generate a notification event that will ensure that the notification is delivered to the end-users because they already subscribed to the topic as subscribers. Finally, the S3 will be attached to EC2 so that the bucket and EC2 instance will be synchronized. Thus, when the user (businessman) puts some content in the S3 bucket, the email system will inform the customer, and the customer will be able to go to the website hosted in EC2. Since S3 and EC2 are synchronized, the items that a user puts in S3 will also appear in EC2. When everything is connected, the subscribers will be updated regularly whenever new content is placed in the S3 bucket. Moreover, the same content will be available in the EC2 instance over the website.

AWS LAMBDA

Introduction

AWS Lambda is one of the services offered by AWS that fall within the compute domain. This also belongs to the broad category of AWS compute services. With AWS Lambda, users can execute code for all application types and backend services virtually. They only need to provide the code in one of the AWS Lambda supporting languages.

Let us discuss Lambda in detail. It automatically executes your code without needing you to set up or manage service. You just need to write down the program and upload it, and then Lambda is responsible for executing that code. Thus, you neither have to provision and manage the services nor require any server to run them. By executing the code, Lambda is able to scale an application and answers every trigger. The program runs side by side, and each trigger is handled separately and scaled concisely to the size of the workload. Currently, this service supports *C#, Java, Node.js*, *Python,* and *Go*. It can execute, call, and answer to some events from other services, and it can execute the functions based on these events. These functions may belong to *Node.js, Java, C#,* etc.

In Lambda, billing is measured in seconds. You will only pay for the duration in which the code executes, which ensures that you do not receive any service charges. You only pay for the time the code is computed for every hundred milliseconds. You pay for the code when it runs, and a couple of times, the code is triggered, so you do not pay anything when it is not running.

How does it work?

Let us see the working of Lambda and how easily and smoothly the complex function works in the backend. Consider a use case in which the clients send data to Lambda. Anyone sending requests to AWS Lambda is a client. These clients may be an application or some other Amazon web service that sends data to Lambda, and Lambda receives the request. It runs on the specified number of containers determined by the size, amount, or volume of the data. If there are just one or a few requests, it will operate on a single container. To handle these requests, they are passed to the container, which executes the code provided by the user to satisfy those requests.

Containers are required to understand Lambda. So they are lightweight, self-executable packages in software that contain all necessary components, such as codes, run times, system libraries, and tools. They are currently running on both Linux and windows. The software will continue its execution without affected by the environment because the container separates the software from the surroundings. For instance, a developing and staging environment can differ, which is a kind of isolation that helps to reduce tension between the teams, which are running the various software on the same infrastructure.

Furthermore, if there are few requests, Lambda delivers them to a single container. However, when the number of requests grows, it will create several containers and share multiple requests to the different containers. Depending on the volume, size, and number of sessions, more containers are given to manage those requests. Whereas, when the number of requests reduces, the number of containers also reduces, which helps to save expenses. Whenever the users are not using any resources, they will not pay for them. They do not pay for resources at all and are only charged when a function runs inside the container.

Lambda Pricing

Let us have an overview of *Lambda pricing*. The first million requests or the first functions a user execute on AWS Lambda each month are free. So, if there is a startup that does not achieve a million requests per month, it means that nothing is being paid for computing. Then it costs only *$0.20* for another million requests on Lambda. Therefore, first, *400,000GBs* of seconds are free, and after that, *$0.0000166667GBs* per second will be charged, which is a very low cost. This value will also change based on memory usage. The above-mentioned values are used for the lowest usage, which is 128 MBs, but usually, users do not exceed *512MBs*, which is very high.

For the instant clue of total pricing, let us assume that there is a Lambda function at *128MBs*, and there are *30 million executions* (*requests*) per month with a duration of *200 milliseconds* for those Lambda functions. So, only *$5.83* will be charged for the given scenario. It can be calculated as *memory * executions per month * time duration per invocation***.** It is clear from this example that AWS Lambda is very economical.

Defaults & Limits

Now let us discuss some *Default & Limits* for *AWS Lambda*, which is necessary for a user to understand. By default, you can have a thousand Lambdas running in parallel. If you need more of these, you can ask AWS support. You can store temporary files on Lambda while it operates, which has a limit of up to 500MBs. When you create a lambda, by default it runs in *No VPC*. In some use cases, you may have to change through VPC, and if you set Lambda to a VPC, it will lose internet access.

The *Timeout* can be set to a maximum of 15 minutes. So, if you want to go past 15 minutes, you can use *AWS Fargate*, which is identical to AWS Lambda. However, there is additional setup and processing in AWS Fargate, due to which, per second, is charged instead of milliseconds. If the need exceeds 50 milliseconds, Fargate is a good option. Lastly, you can adjust the memory between 128MBs to 3008MBs. Memory starts at 128MBs and increases up to 3008MBs by the increments of 64MBs. If you use more mega-bytes, the more costly, it will get to match with the time it takes.

Cold Starts

Now one of the most important concepts of AWS Lambda is *Cold Starts***.** AWS has pre-configured servers which are lying in the switch-off mode for the user runtime environment. These servers need to be switched on when Lambda is invoked, and the code needs to be copied over. So, when the function initially runs, there will be a delay during this time, which refers to as Cold Start. Assume that you have a Lambda function that gets triggered, and to run this function, there is no server. So, the server will get started here, and you will have to copy the code due to which delay occurs. However, if you want to invoke that function again, the code is available, and the server is active so, you will not face the delay again. The cold start will not happen at this time, and now your servers are warm, which refers to as *warm server*.

Cold Start is one of the drawbacks of using *serverless functions*. The serverless functions are inexpensive, but they also come with a tradeoff. A user may experience delays by using serverless functions cold starts. There are some strategies to overcome the cold start issue. One of them is called *Pre warming*, which allows the servers to run continuously. So you can invoke a function to start the servers prematurely, and as soon as someone starts using it, they will stay warm. The other approach for a cold start is that you can take a Lambda and assign more responsibilities to it so that more operations will process through it, making the *warm* more persistent. Thus, cloud providers are trying to find solutions to reduce these delays, making cold start less of an issue.

Use cases

There are many use cases for Lambda, which can be used in different ways, especially for the business world. There is a use case of AWS Lambda, which is used for image processing when the images are uploaded in the S3 bucket. Let us assume that an object is uploaded with the unexpected format in the S3 bucket, which means that the file needs configuration. So, it triggers AWS Lambda, and the bucket is filled with new objects. These images are managed and converted into thumbnails as per the device. The device that reads the data on the other end could be any device such as a personal computer, an Android phone, An Apple phone, or a tablet. Therefore, Lambda is triggered based on different devices and formats, while videos and other types of images are also converted into the desired format.

The data related to social media is also analyzed using AWS Lambda. Let us suppose that you are gathering the hashtag trending data. This data will be gathered and inserted into the Kinesis stream to be fed to the Amazon atmosphere. Thus, Lambda will be triggered, it will receive the data, and then it will be stored in the database. Businesses can also use this data for future analysis.

Let us suppose another scenario in which you need to set up temporary storage and a system for backup (near real-time backup) as soon as the data is uploaded. Now, near real-time backups are almost impossible and are not much useful. However, businesses demand near real-time manual backup, which is just a manual backup instead of near-real-time backup, and it is not effective at all. Even if you come up with the solution of manual backup that is close to near real-time backup, it will not be effective. So, by observing the size of data and the number of times this data is added to the source bucket, there is no possibility that you can do it manually and keep it as real as possible. However, if it is still a requirement, you can use AWS Lambda to set things up. So, AWS Lambda manually handles the backup.

In other words, Lambda comes to the rescue in such difficult and impossible situations. So, for this particular use case, two S3 buckets have to be created. One is a source bucket where the data will be stored, and the other is a destination bucket where the data will be backed up from the source bucket. Furthermore, an IAM role for the communication of these buckets and a Lambda function to copy the files from source bucket to destination bucket is also required. So, the Lambda function will be triggered whenever there is a change in the metadata for the bucket, which is then uploaded to the destination bucket. When everything is set, you can test it by adding the data in the source bucket, which will automatically replicate, or the data will be copied from the source bucket to the destination bucket automatically.

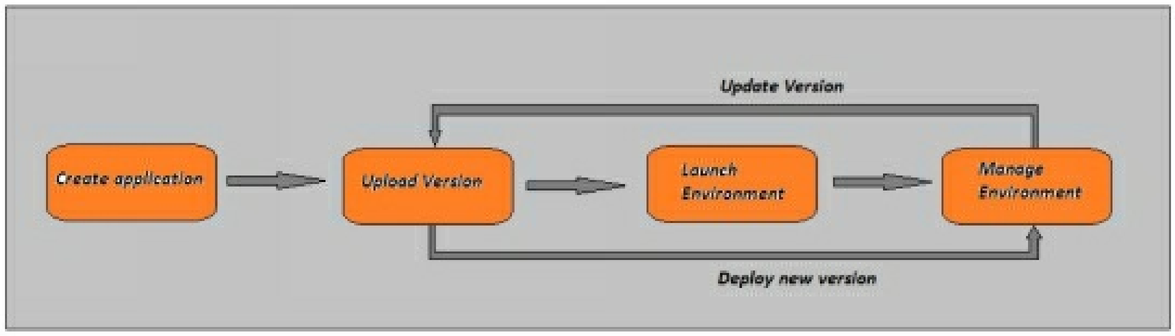
Let us understand the replication of data between two buckets, and for this purpose, there is a feature that comes along with AWS S3 known as *Cross-Region Replication*. You can use Lambda if the replication between two different buckets in two different accounts is required where data is inserted into the source bucket and replicated into the target bucket. In order to begin the process, there must be two buckets, an IAM role (that enables you to push the data from source bucket to target bucket), and then create *Lambda Events* or *Triggers* to look for the event in the source bucket. When new data is inserted, Lambda gets triggered, copying the data from the source bucket and pasting it in the destination bucket. Moreover, it uses the IAM role and policies for the required permissions. So, in a couple of clicks, a manual backup system is set, which runs smoothly without any manual interference. Furthermore, it can be as near-real-time as possible.

ELASTIC BEANSTALK

Introduction

Let us now discuss *AWS Elastic Beanstalk*. It is used as a *Platform-As-A-Service* that allows coders to deploy and manage their applications easily without knowing the infrastructure that runs those applications. Elastic Beanstalk minimizes management complications without limiting the control or choice. One simply needs to upload the application, and it will manage the load balancing, health monitoring, capacity provisioning, and scaling of applications. Without Elastic Beanstalk, you may be able to run applications on AWS, but it does take time to pick and bundle services from a wide variety of AWS ecosystem options. Elastic Beanstalk allows you to extract the infrastructure layer to make it easy to install and manage your applications on AWS, but you must be aware of the underlying infrastructure.

Elastic Beanstalk workflow:



For using Elastic Beanstalk, you need to develop an application and then upload its version in application source bundle format (which includes Java .war file) to Elastic Beanstalk and describe a few details about the application. When you launch this application, its selected and supported platform version is deployed by Elastic Beanstalk. Then, the AWS resources that are necessary for your code execution will be created and configured. Once you have deployed the environment, you are able to handle the environment and launch new versions of that application. It supports the applications built-in Java Node.js, Python, Go, .NET, Ruby, and PHP.

This service can set up the Auto-Scaling Group, database, EC2 instance, and Elastic Load Balancer pre-configured with your platform. You can also build a customized platform on Elastic Beanstalk and execute it. It can also run dockerized environments in the form of *Docker*s and *Multi-container Dockers*. The *Blue/Green* and *In-Place deployments* are a couple of deployment methods in Elastic Beanstalk. Now, it is in In-Place deployment by default, but blue/green deployment is also deployable. You can also get monitoring with the help of Elastic Beanstalk. It also contains interesting security features, such as it can rotate out passwords for you if an RDS is connected. It is not recommended by AWS for production applications. However, large enterprises and some startups are using Elastic Beanstalk. Amazon does not include any additional cost for using this service. You are only required to pay for those resources that are used to execute and store your applications.

Use Case Scenarios

We are going to discuss a couple of use cases for Elastic Beanstalk. Let us assume the first use case where the client demands a web application with no underlying infrastructure concerns. Thus, you can launch Elastic Beanstalk for such clients and offer multiple functionalities according to their needs that would normally require a lot of time for application configuration and maintenance. Your client will get an Elastic Beanstalk stack for a deployment process that has many AWS features in a single package.

- Log Rotation
- Load Balancer
- X-Ray
- Blue/ Green deployment
- Auto-scaling / auto-healing instances
- Security Groups

- Health checks/monitoring
- Networking / VPC Configuration
- Managed updates
- Integration with deployment tools such as Git and Visual Studio

Whenever a client demands a stable application deployed without any complication, then Elastic Beanstalk is the most suitable option in such cases.

For the second use case, let us assume that the client wants to manage a few infrastructure components, but he is not interested in learning the complete setup procedure of AWS. In such cases, Elastic Beanstalk allows the clients to manage some updates, such as managing deployment, the size of Auto Scaling Group, and switching over to the latest version without the need to access or train on all the other AWS systems. You can use this service to deploy and make changes from one place in multiple locations traditionally. For instance, a client can launch the latest version of the code, update the ASG min/max and instance size from a single screen.

Benefits

- It provides a complete stack of applications in few minutes.
- The developer may concentrate on the application with no infrastructure concerns.
- Dynamic and Automated scaling on the basis of usage/load.
- Automatic updates in the platform.
- You can integrate it with other AWS services.
- Accessibility control is role-based (IAM) across all AWS services.
- Audit logging and security of cross-service APIs via (CloudTrail),
- Maintain complete control of resources and frequent accessibility.

Limitations

- Any subsequent deployment or modification in the Elastic Beanstalk stack will bypass any changes made to the resources outside of Elastic Beanstalk.

- It is unable to copy stacks from one to make another stack. To make another stack, you need to use the Elastic Beanstalk CLI.
- Scaling and launch issues are not simple to resolve here because of the limited information provided.
- The worker processes are not supported by Elastic Beanstalk.
- There is a limited set of software targets for deployment.

CHAPTER 3: STORAGE FUNDAMENTALS

Overview

Storage is one of the basic building blocks of *Infrastructure-as-a-service* (*IaaS*). AWS is undoubtedly leading the cloud computing market, which also includes the cloud storage market. In this module, the storage services offered by Amazon will be explained in-depth, including their key features and the reasons to use these services in your environment. This chapter presents an outline and introduction to the various AWS storage services and gives an insight into the process of data transfer from and to AWS along with an understanding of choosing the storage service that best serves your needs.

Amazon web services provide a variety of storage options, and five major storage options are available to the users. The first and most frequently used AWS cloud storage service is the *S3-Simple Storage Service*, which stores and collects data from the internet in any quantity. Then there is *Elastic Block Storage*, the Solid-State Drive (like C drive, D drive, or the E drive in your computer) connected to the EC2 instance. Then we have *Elastic File System* in the AWS storage options. The underlying technology of EFS and EBS is quite the same, but these services differ from each other in one way. EBS (D and E volumes) can only be accessed through the instances attached to it. On the other hand, the EFS is a shared file system, and multiple file systems can access it. EFS can be accessed from inside the Amazon environment as well as from on-premises.

We also have *AWS Storage Gateways*, which are used to shift the data from the user's local environment to the cloud environment securely and keep the local copy of data. So, the data can be accessed from on-premises as well as from the cloud. In the end, we have *Snowball* and *Snowmobile*. Snowball is a system for importing and exporting user's data. It is hardware sent to your on-premises where you can copy the data in it and then send it back to Amazon, and then Amazon will copy this data to the location which you have mentioned in your account.

Furthermore, if the size of the data is too big, then Snowmobile can be used, which is a data center on a Truck. Amazon would send a truck loaded with a data center in a container with high compute capacity, lots of storage space, and much more. So, when the truck arrives, it will be parked near the client's data center and is connected to it through cables. In this way, you can transfer the data into it and send it back to Amazon, where they would copy that data into your account and in any other storage that you notify them. So, if the data size is too big, you would call Snowmobile. However, it is not available in every region.

In the case of cloud storage, these popular services operate differently and deliver varying performance, availability, cost, and scalability levels. We will discuss these storage options with the help of use cases and compare their costs, accessibility to store data and performance. There is a common benefit of using these services that the users do not need to manage their servers independently, but they will be required to make a selection. This chapter explains these services in a simplified form, which will help select your desired storage service.

Now let us move on and discuss the AWS storage in more detail.

AWS SIMPLE STORAGE SERVICE

Introduction

AWS S3 stands for *Simple Storage Service,* an *Object Storage Service* in which data is stored and recovered from the internet, but a user cannot install anything. There are two types of storage in AWS that are Object storage and block storage. In Object Storage, you can store and recover the data but cannot install anything. It can be accessed directly from the internet. On the other hand, the Block Storage needs to be connected to an EC2 instance as you cannot access it directly but can install the software in it.

AWS S3 is designed to store and collect data from the internet in any quantity. It is also accessible through the web interface. There are different ways to access this storage service. One method is to drag and drop the content directly into S3, and the other way of recovering the data is to open a browser and click on the download button. In this way, you will be able to download any content with a size up to 5TB. You can download thousands of files like that, but a file can have a maximum of 5TBs. S3 is designed for developers where instead of storing the data locally in the server, they can push or retrieve the data from S3 anytime they want. You can also use S3 as a code repository to save your code so that the applications will read the code from that code repository. Furthermore, you can share the code with another user with full security and encryption using S3. Amazon also provides a service known as Import/Export to transfer huge data volumes that will allow you to upload and download the data in S3.

The Amazon S3 offers 99.999999999% *durability* and 99.99% *availability*. The term Durability means that the data gets lost if it is stored somewhere else, whereas it will not be lost with AWS due to its 99.999999999% durability. The other term, availability, means that data is available to users at any time they request it. In AWS, you can access your data with a 99.99%

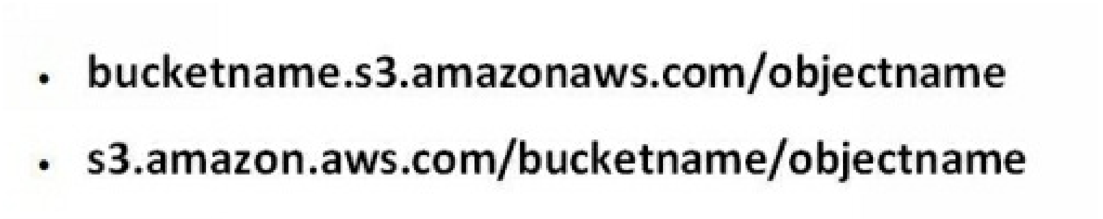
availability rate. So, whenever you request the data, it will be available to you without taking a lot of time. Amazon offers its S3 services in US East (N. Virginia), US West (Oregon), and Asia Pacific (Mumbai, Sydney & Tokyo).

Now let us move onto discussing the working and basic building blocks of AWS S3.

How does it work?

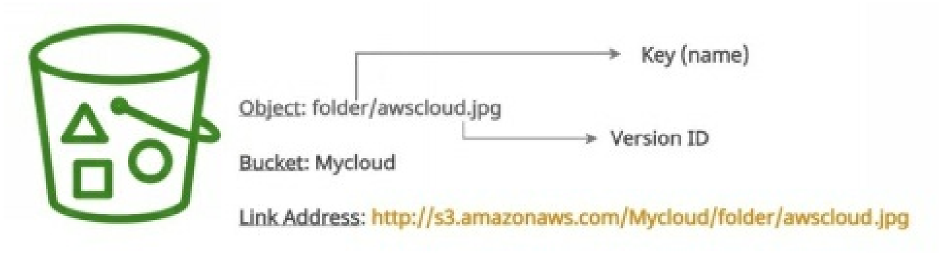
Let us take a look at the terms Buckets and objects in Amazon S3 before we learn about how it works. *Buckets and Objects* are known as the S3's basic building blocks. An Object is the original data with some additional information (a reference to the data) such as file type JPEG file or PNG file, name of the file, and the time it was added. This additional information is also called metadata. Therefore, we can say that an object is a combination of original data and metadata. A bucket is a container that receives the data and stores it securely. The name of the bucket must be unique in all Amazon accounts because the name of the bucket is a piece of the URL. There is a concept of the folder in S3 with the management console, which refers to the grouping of several objects. You cannot create a bucket inside the other bucket, but it is possible to have a folder inside a bucket.

You can access the object via a public URL when it is uploaded. The Two URL types available in S3 to use are:



- bucketname.s3.amazonaws.com/objectname
- s3.amazon.aws.com/bucketname/objectname

The key refers to an object's distinctive identifier within the S3 bucket. You can have a separate key for every object in a bucket. These objects have an Access Control List with the help of which you can figure out that if you can share the object over the internet. A specific version ID is created and assigned to the object by S3 whenever you insert the data into a bucket to identify it in the future easily.


Key (name)
Object: folder/awscloud.jpg
Version ID
Bucket: Mycloud
Link Address: http://s3.amazonaws.com/Mycloud/folder/awscloud.jpg

Let us discuss the working of the S3 bucket. Buckets are created in the S3 service to store files (based on objects), which act as folders. Whenever a bucket is created, the region must be specified to deploy that bucket. The expected latency, security policies, and service usage costs are generally kept in mind before selecting any region. You can store Objects in an S3 bucket. Amazon S3 objects are accessible and managed with the help of a web interface by default. All the objects that are uploaded to the S3 bucket are independent with respect to their characteristics and related permissions, meaning that who can and who cannot access the files. When you upload a single file or set of files to the bucket, the S3 storage type must be specified for those specific objects.

Whenever you store a file as an object in S3, by default, it is stored in several places at the same time, which includes availability zones, data centers, and on various disks. The S3 service will check control hash sums to monitor data consistency regularly. In the case of detecting any data corruption, S3 will recover the object with the help of the redundant data. The S3 Lifecycle Management feature allows you to create lifecycle policies that will assist in moving the objects from one storage type to the other storage type after the specified time. You can define these policies depending on the most recent access for the object or the object's initial creation.

Let us now discuss different storage classes and features in Amazon S3.

S3 Storage Classes

To understand these classes, consider a use case in which there are different data types in a school with different features and validity. Let us take this use case and discuss S3 Standard first.

S3 Standard

It is the default storage class of Amazon S3 and good for use cases having low latency. Let us say if you want to access the data of student’s attendance, you can easily retrieve it within no time with the *S3 Standard*. So, it is a suitable use case for data storage in S3 standard. Thus, the right candidate file in our use case is the data that is frequently accessed and retrieved, such as a student's attendance report or attendance sheet, which is accessed daily. It can be immediately accessed and retrieved by you whenever needed.

S3 Standard-IA

The other S3 storage option is *Infrequent Access* or *Infrequent Data Access*. As the name suggests, it is used for data that is not accessed frequently. The candidate data for infrequent data access includes student's academic records, which are not required daily. However, you can use S3 Standard-IA when requested. Therefore, the data is not required daily, but when you want to access it, you can quickly access it.

Amazon Glacier

Amazon Glacier is an archival solution in the cloud. It provides a cheaper storage service in the cloud environment to store the data that requires a long retrieval time. It was basically built to Archive cold or static data that is not intended to be modified or accessed for longer periods, such as for several months, years, or even decades. It is not appropriate to store the active data (which changes frequently) in Amazon Glacier because the archived data in this storage tier is static. It means that you cannot move, copy, or apply changes to this storage data; however, you are still able to download or delete it.

The *Archive* is a basic storage unit in Glacier. It represents a single file or combination of multiple files which can be uploaded as a single archive. There is a distinctive system-generated identifier in the form of a lengthy string of numbers and letters for each Archive in the Glacier. You can use this identifier as a key to recover an archive in the future. Generally, the archives in Glacier are not directly accessible. You can download or upload the data from/to the Glacier by using Amazon S3 because it maps the system generated identifier in the Glacier with the user-defined object name in S3.

File backup to Amazon Glacier-Amazon S3 is used for file backup to the Glacier (not directly to it). When you copy the files to S3, Amazon will move the files from S3 to the Glacier, which depends on your specified S3 lifecycle policy. When an object is archived in Glacier, its S3 storage class will be changed to 'GLACIER,' which shows that this storage class's content is moved to Glacier. However, if you want to retrieve, delete, or update an archive in the future, you can use the *Index Entry* stored in S3. For long-term Glacier storage, database backups can be moved from tape storage media to the cloud as well.

Files restore from Amazon Glacier- Given that Glacier offers an inexpensive storage solution to upload unlimited data, retrieving data from Glacier is a much slower and complicated process. There is a retrieval delay of three to five hours for every restore request to the Glacier before downloading the archives. The process of data retrieval from Glacier involves two steps. The first step involves requesting Glacier to store a copy of data in Amazon S3 temporarily, whereas, in the second step, you can download the data from S3 to the targeted location.

The candidate data that can go into the Amazon Glacier is the student's admission fee, which is not critical data. Whenever you are required to access this data, you can always wait to retrieve it because high performance is not required for archives. In other words, the retrieval of data takes time when you request it. A student's old record is the perfect candidate data for storing and retrieving from Amazon Glacier in our use case.

S3 One Zone-IA

In this Storage Class, the data is not accessed frequently and stored in a single availability zone. Amazon stores data in multiple availability zones by default and charge the user for that storage. If you demand to keep the data costs lower so in that case, you can choose One Zone IA Storage Class as it keeps data in one AZ. The candidate data for S3 One Zone-IA is the student’s report card.

Standard Reduced Redundancy

This storage is used for the use cases in which data can reproduce quickly, and it is not urgent such as a copy of the PDF Library book. You would have a source PDF, and you can produce copies of it so that it will be available for readers to read.

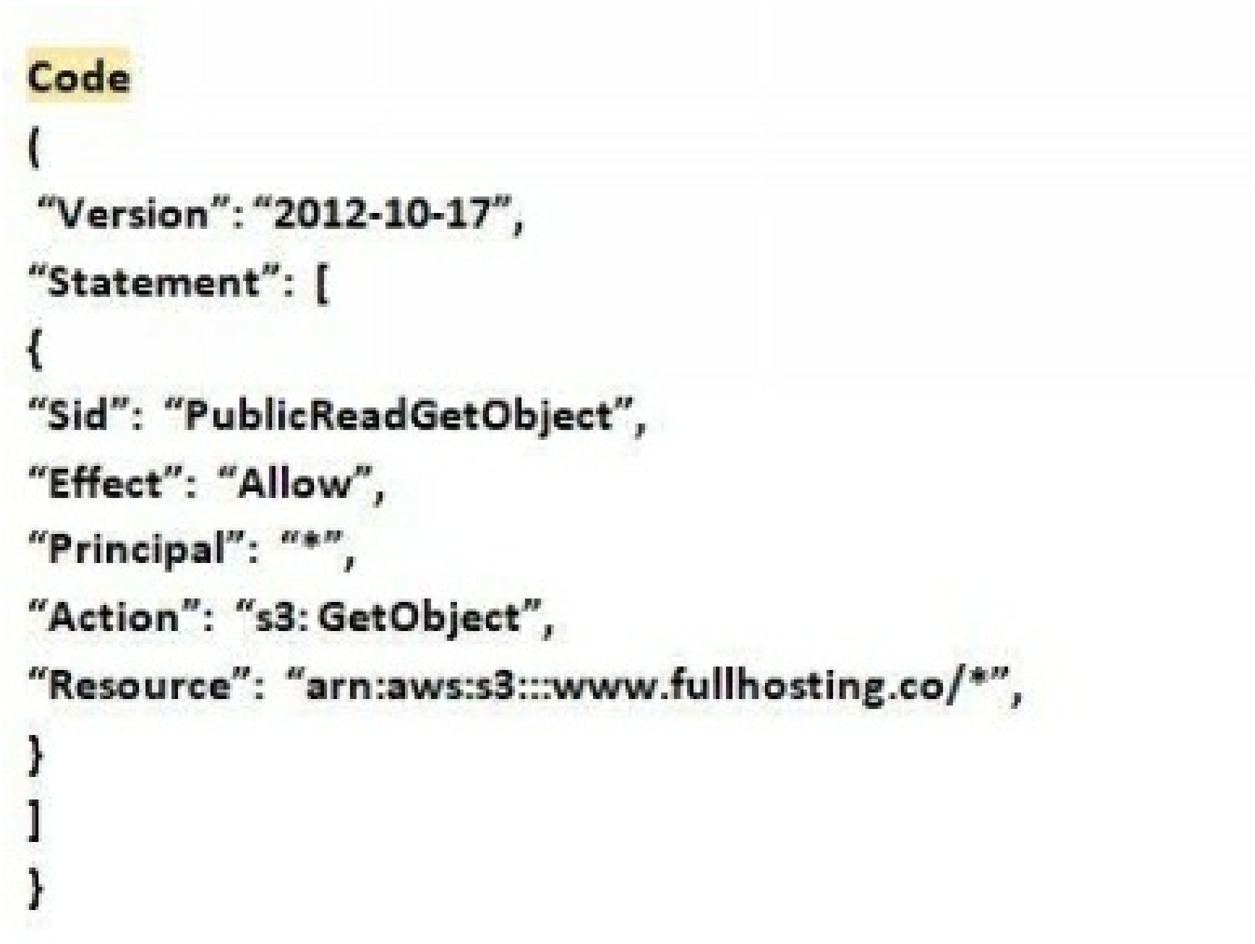
Features of Amazon S3

Let us discuss the features of Amazon S3 in detail.

Security

Let us look at *S3 Security*. When a bucket is created, it is private by default because AWS is very concerned about keeping the bucket private. They have already changed the interface several times. They send emails to the users to inform them about the public buckets because it is a severe weakness to AWS, as people usually leave these buckets open. When a new bucket is created, all public access is denied, and if a user wants to have public access, he needs to check either for his ACLs or bucket policies.

In S3, you can enable *Logging* for every request to get detailed information about the objects, such as the minor detail of objects that were accessed, uploaded, or deleted. When the log files are created, they will be saved in a different bucket instead of the same bucket. You now have two choices for managing access to your objects and buckets. One is *Bucket Policies*, and the other is *Access Control List (ACL*). ACLs are used for implementing fine-grained control over specific bucket objects, whereas Bucket policies can be used for applying global control, such as making a whole bucket public. In S3, the first authorizing method was Access Control List. But now, the newer method used by nearly all AWS services is bucket policies. ACLs are simple but outdated, while policies can enforce very complicated rules and permissions. Therefore, you might prefer ACLs over bucket policies in some use cases because it is easy to grant access through ACLs. For granting access, you can list the objects, write them, or just read and write the permissions by right-clicking a bucket or an object. Bucket policies are a little bit more complicated because a user needs to write a JSON document policy. You can see an example of JSON code down below.



Code

```
{
 “Version”: “2012-10-17”,
“Statement”: [
{
“Sid”:  “PublicReadGetObject”,
“Effect”:  “Allow”,
“Principal”:  “*”,
“Action”:  “s3: GetObject”,
“Resource”:  “arn:aws:s3:::www.fullhosting.co/*”,
}
]
}
```

Consider a use case in which a user has static S3 hosting so that he will use bucket policies. Here, Fullhosting.co is a website where individuals are allowed to read access for the **‘**Getobjects’ to the bucket. That is why it is used in more complex setups. Though bucket policies are complicated than ACLs, they are more common than ACLs.

Encryption

The major characteristic of security is *Encryption.* When you upload files to S3, it uses *SSL* or *TLS* by default. You will have encryption in transit, which means that SSL or TLS will manage the traffic between a local host and S3. *Server-Side (SSE)* and *Client-Side Encryption* are the two encryption types. Original data remains static for SSE, and Amazon will help you encrypt the object data. There are few options for the S3 managed keys in SSE, such as *SSE-AES, SSE-KMS***,** and *SSE-C*. The first option is just an algorithm for encryption, which means that when it uses encryption, there will be 256 bytes in length or characters in length. S3 manages the whole encryption as it is doing all the work for you. There is another service called the KMS (Key Management Service), which uses envelop encryption for data. The key is encrypted with another key, and with KMS, these keys are managed by either AWS or by you. In the last option, ‘C’ stands for customer-provided, where keys are provided by the customer. Here, you will provide the keys yourself. There is an interface for it, which is a little bit more complicated. On the other side, there is client-side encryption. There is no interface in client-side encryption, and the user is encrypting the files and uploading them to S3.

Data consistency

Now, we will discuss the S3 data consistency model. In this model, S3 provides consistency of Reading-After-Writing for the PUTS operations of new objects, which means that reading an object will always be successful after writing it. All new objects created in multiple AZ's are going to be reproduced before success is returned. With the help of Read-after-Write consistency, a new object will be visible to every client instantly. In addition, it is more consistent than the eventual consistency. S3 service will provide eventual consistency for Read-after-Write while making a GET or HEAD request to the key name before creating an object. The request is made to figure out if the objects exist or not.

There is a difference in consistency between when you write data to S3 (writing new objects) and overriding files or deleting objects. When the new data is sent to S3 as a new object, it will be Read-After-Write consistency. It means that as soon as the data is uploaded, it can be read instantly. Likewise, when you *PUT* an object, you will be able to get the object as soon as you get the success response. The S3 always takes time to duplicate an object to all Availability zones for overwriting and deleting an object because of its eventual consistency for overwriting *PUTS* and *DELETES*. As the name implies, overwrite PUTS shows that an object already exists with a similar name for the new object you will create. On the other hand, overwrite DELETE implies that the existing object in the S3 is being deleted. An eventual consistent system means that when there are no latest updates, then the system's queries will eventually return similar outcomes.

If you instantly read the data, the modifications may or may not be visible to you. If you have to read the data immediately, then S3 might return an old copy to you. It only takes a few seconds to update, yet it might be unlucky for you in some use cases. Therefore, you must be aware of this possibility.

Cross-Region Replication

Let us discuss cross-region replication. This S3 feature allows the asynchronous and automatic copy of data from one destination bucket to another situated in any other AWS Regions. It was created over the existing versioning facility of S3 used to copy the files from different geographical locations. When you enable this feature, it will automatically duplicate the object that was uploaded to a certain S3 bucket into the destination bucket, situated at different AWS locations. Any Access Control lists or metadata that is associated with the object will also be copied through the replication process, which you can enable and manage by using S3 API.

Bucket replication cannot be performed in a single region. You have to enable S3 versioning for both the origin and target buckets in order to use *Cross-Region Replication*. Lifecycle policy rules can be set up inside a destination bucket for deleting the previous versions of the data or storing them in Amazon Glacier. Likewise, if you want to execute these actions at the source bucket, you must configure the same lifecycle policies on the source bucket. When an object is copied from one bucket to the other, the data will be encrypted with the help of SSL in order to secure it from attacks. If the source bucket's data is already present in the destination bucket, it will not be replicated even if the process is carried out. In this way, the existing objects will not be replicated, and the redundant data will not be stored. The Cross-Region Replication will replicate all the future uploads of all the objects to the other buckets.

It provides higher durability in case of a disaster. It must be enabled, and the destination bucket with the alternate region must be defined. Then, those objects from the region of origin will be automatically replicated to the targeted region. An object can also be replicated to another AWS account bucket. To avail of this feature, the versioning must be turned on in both types of buckets.

Versioning

Versioning helps you to keep several objects forms in the same S3 bucket. This feature is used to fetch, restore, and secure each object version in the S3 bucket. For instance, there are two objects in a bucket with the same key but separate version IDs, such as picture.jpg (version ID is 12) and image.jpg (version ID is 11).

You can set the bucket versioning in S3. Thus, versioning allows you to set a version for objects and helps to prevent data loss by keeping a record of versions. You can recover the objects from overwriting or deletion with the help of versioning enabled buckets. It has two main objectives:

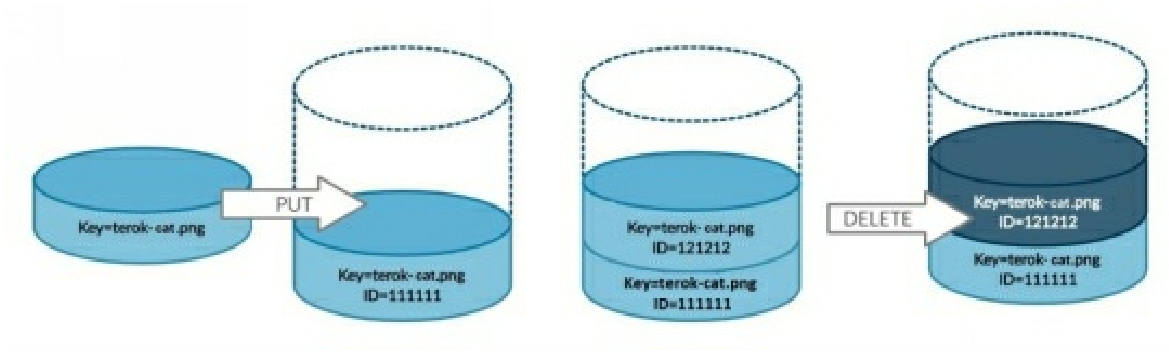
- On deleting an object, it will not get deleted permanently. Versioning will create a delete marker which will become an object's current version.
- In case of overwriting an object, it will restore the object's older version and create a new version for it.

You can apply a versioning state to every object in the bucket. When the versioning state is turned on, every object in the bucket will become versioned, and they will have a separate version ID. A couple of necessary points are as follows:

- When the versioning state is turned off, the objects' version ID will be set to null, and the existing objects will not be affected.
- The owner of the bucket has the authority to stop the object versions by suspending the versioning. Suspending the versioning will not affect the existing objects.

Let us discuss an example of versioning. If you have a file with an image called cat.png (a name is the same thing as a key), then the image will have a version ID. Consider a use case where the version ID for this image file is *111111*. When you add a new file with a similar key, the enabled versioning feature will create a new version of this file by assigning a new ID. Let us suppose it is 121212 in this case. So, when you need to access this object, it will always take the one from the top, which is 121212 here.

On the other hand, for deleting the object, it will access the previous file. So, versioning is a proper way to secure data loss. Moreover, if you want to access a previous version, you can do it, as it only requires specifying the version ID. S3 versioning cannot be disabled once it is turned on, but it can only be suspended. So, when turned on, you are not allowed to remove versioning from existing files; all you can do is suspend them. Furthermore, all the base files will be available in one version.



Lifecycle Management

The *Lifecycle Management* in S3 automates the process of transferring bucket objects into various storage classes and removing those objects altogether once it is no longer required. This feature helps you to check your S3 Bucket objects automatically and transfer those objects to Glacier or remove them from S3. It can be used for security purposes, legislative compliance, internal policy compliance, or general housekeeping. Implementing good lifecycle policies, which are set up at the Bucket level can improve data security. Each bucket will have up to 1000 policies. By using object 'prefixes,' various policies may be set up within the same bucket affecting different objects. There is no need to launch the policies manually since they are checked and run automatically.

Consider a use case where you have an S3 bucket, and you created a lifecycle policy. In this lifecycle policy, you have specified that move the data to the Glacier after seven days because you will not be using it for at least a year. At the same time, you keep the data around for compliance and expect a cheaper cost for storing it, so that is possible with the lifecycle policy. After a year, you will create another lifecycle policy to transfer the data into any required storage and delete the older version. Life cycle management can be used with versioning and is applicable to both current and previous versions.

Transfer Acceleration

Let us look at *Transfer Acceleration* in S3. This feature allows end-users and S3 buckets to share files quickly and securely between them over long distances. So, when you upload a file, you want it to reach S3 as soon as possible. So, instead of uploading it to S3, you can send it to a distinct URL for an edge location. An edge location is the nearest data center to the user. When the data reaches an edge location, it will accelerate uploading to the S3 bucket by using the AWS backbone network, an optimized network path.

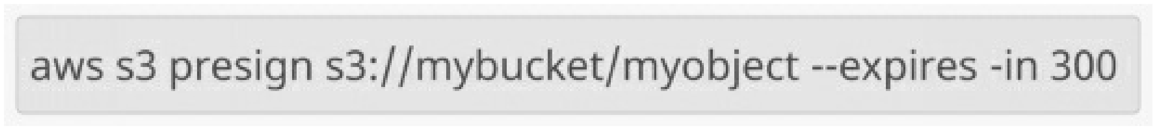
Pre-signed URL

It is a URL that is provided to the users to grant temporary access to a specific S3 object. Pre-signed URLs are generally used for accessing private objects. You can Read or Write an object using this URL (or update an existing object). A pre-signed URL uses three specific parameters that are set by your application. They are bucket, key, and expiry.

- The object is stored in the bucket
- Key is the object’s name
- Expire refers to how long the URL is valid.

You are unable to communicate with the listed object after the expiry time has passed. You can generate this URL that will allow you to temporarily access an object for uploading or downloading object data to the endpoint.

Moreover, the CLI or SDK can be used to generate the pre-signed URLs. In the CLI, you can identify the actual object that will be accessible for a short time (assume it expires after 300 seconds). So, it will generate a lengthy URL with an access key that will provide temporary authentication so that you can work with the object.



```
aws s3 presign s3://mybucket/myobject --expires -in 300
```

Pre-signed URLs are used while building a web application. Consider a very common use case in which you want to allow users to download files from a part of the web application that is protected by a password as those files in S3 are private. Thus, a pre-signed URL is going to generate, and the users need to download the file within 5 seconds, as it is accessible for 5 seconds.

MFA Delete

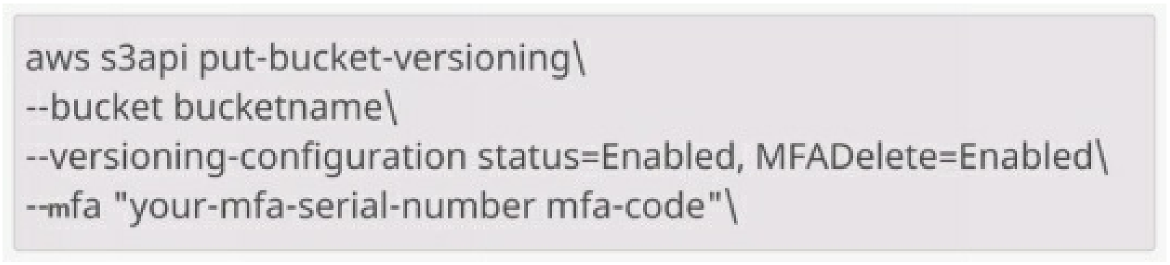
The accidental deletion of objects is one of the key concerns of the users. So, to avoid such situations, AWS offers a secure feature known as *Multi-Factor Authentication*, which is implemented on the S3 bucket. MFA forms a security layer for the following reasons:

- Permanent deletion of versioned objects.
- To modify the versioning state of the objects.

A two-level authentication is required while setting up MFA Delete on a bucket. They are:

- Using security credentials.
- Six-digit code from a verified authentication device, which includes Google authenticator.

MFA Delete is an excellent way to ensure that files do not get deleted when users access them, but it has some limitations. You cannot use MFA delete feature until versioning is turned on for the bucket. This feature can only be enabled or disabled by the bucket owner who is logged in as the root user. The owner of the bucket is also able to delete the object from the bucket. The AWS CLI can be used to turn MFA delete on to check the configuration for versioning. The example of MFA Delete is shown down below:



```
aws s3api put-bucket-versioning\
--bucket bucketname\
--versioning-configuration status=Enabled, MFADelete=Enabled\
--mfa "your-mfa-serial-number mfa-code"\
```

Benefits of S3

Let us discuss the benefits of AWS S3. As we discussed earlier, that AWS S3 is a very durable and highly available storage service because it provides 99.999999999% stability and 99.99% availability. Highly availability means that the S3 storage is not the regional service, so it does not depend on AZs. If an AZ goes down within Amazon, it will not affect the user's capability to access the S3 storage. It is very scalable, which means that it is not necessary to pre-provision storage capacity. If lots of S3 storage capacity is required, you can easily get it. Moreover, once you are done with usage, you can remove some data to pay less in that specific month.

There are always different storage options for you to store the data in S3. You can use any AZ in the S3 console to store data if you want more stability in the region. When various regions are available for you to move the data, you can store it in any one of them. This storage service is also very affordable, making it the inexpensive alternative among all Amazon's storage options. It is very flexible when it comes to cost and storage. S3 itself is a low-cost service, so there are many pricing levels within S3 in terms of cost, and most of them depend upon durability.

It is effortless to transfer the data with S3 by just browsing the bucket and uploading the data; thus, it will be uploaded. S3 is a secure service because it provides different security features such as *Encryption*, *Bucket Policies*, and *MFA Delete*. It introduces a secure layer of protection over the data stored in S3.

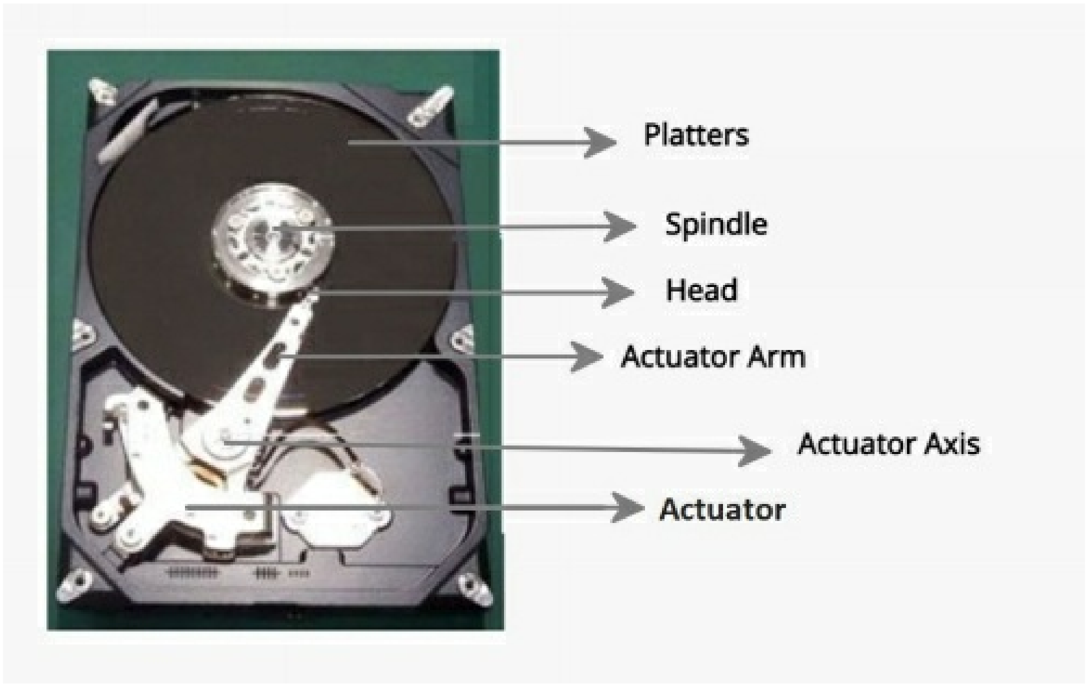
General Storage devices

Before diving deep into the Elastic Block Storage, let us first recall the different storage volumes in general. The three types of storage volumes are:

- Hard Disk Drives,
- Solid State Drives,
- Magnetic Tapes.

Hard Disk Drive

The first one is the *Hard Disk Drive* which is magnetic storage with rotating *Platters* on an *Actuator Arm* and a *Magnetic Head*. It is a round device (basically like a recorder) having an arm and a little needle (head) at the end of it. The HDD is used for writing continuous data. It is good at writing a bunch of data, but when the user has many small reads and writes, then there will be more physical movement involved. So, when the device's arm goes down, it means the user has a lot of data. In case of many read and write operations, the arm needs to lift up and move to where it needs to go for a read operation, then go down for write operation, and then lift up again, which is going to affect its ability to have high IO. So it got restrictions because of its arm. Hard disk drives are suitable for throughput because they write continuous amounts of data, which means that a fast amount of data is written constantly. However, the limitation here is that it involves the physical movement of parts.



Solid State Drive

The next storage volume is *Solid State Drives (SSDs).* They do not have physically moving parts. They use integrated circuits to transport the data and store it on devices such as *flash memory*. Besides, SSDs are resistant to physical shocks. They are silent as there are no moving parts; moreover, they have fast access times and lower latency. They are best for frequent reads and writes, having high IO and best throughputs. Therefore, when the user requirement is of high IO, then SSD is a good option.

Magnetic Tapes

Let us look at the last storage volume. There is a magnetic tape in old computers that looks like a film that consists of big reels. Even today, these magnetic tapes are in use due to their durability. They are extremely cheap to produce and last for decades. However, they are not in use like they were in the past in the form of big reels, but now they are used in modified shapes like *Tape Drives***.** By inserting a *cassette* into the drive, which contains the magnetic tape, users can use it easily.

Now, let us discuss the Elastic Block Store and its volume types in detail.

ELASTIC BLOCK STORE

Introduction

Elastic Block store generates new volumes connected to EC2 instances, backed up via *snapshots*, and provides easy encryption. It is a virtual hard drive in the cloud. It is highly available and a durable solution for attaching continued block storage volumes to EC2 instances. To keep volumes safe and protected, they are automatically replicated within their AZ. The data will be stored on AWS EBS servers even when the EC2 instances are disabled. It provides the same low-latency performance and high availability within the chosen AZ, enabling you to scale storage capacity at a low pricing model. Like a physical block storage drive, you can also attach, scale, and detach data volumes with any EC2 instance dynamically. Being a highly dependable cloud service, the EBS provides 99.999% guaranteed availability.

The EBS volumes are attached to a single EC2 instance to work as a single device, due to which they act as raw and unformatted block devices. One must keep in mind that EBS volumes are independent of the instances on which they are installed and can be used as a hard drive. A single instance can have 20 EBS volumes where each volume's size ranges from 1GB to 1TB. In an EC2 instance, your data is stored in local storage that is accessible until the instance is running. However, the data will be lost if you shut down the instance. So, when you add the file to an EC2 instance, it is suggested that you must save it on Amazon EBS as the EBS volumes will always be accessible to read. EBS volumes, as primary storage are suitable for databases, file systems, and applications that demand granular modifications. For instance, on top of the EBS volumes, you can build a file system and modify the configuration of that volume anytime.

Let us discuss some basic knowledge about storage mediums that will help you understand the EBS Volume types and their use cases. So the first term is

IOPS which stands for Input/Output per Second. The speed at which read and write operations are performed separately on a storage medium is known as *IOPS*. So whenever the term *high IO* is used, it means that the medium can do lots of minor and fast read and write operations. Another term of *throughput* is a data transfer rate from one storage medium to another in Mbps. Moreover, there is Bandwidth, which is the measurement of the overall speed at which data can travel through a network. To understand these two terms, think of a pipe and water. The Bandwidth is considered as pipe and throughput as water. In this way, it is easy to understand the difference between these two terms.

EBS Volume Types

Let us now understand the different types of EBS volumes and their use cases. There are five EBS volume types:

- General Purpose SSD,
- Provisioned IOPS SSD,
- Throughput Optimized HDD,
- Cold HDD,
- EBS-Magnetic

Let us discuss the Solid-State Drives first. The *General Purpose (SSD)* is used for general purposes without certain requirements or demands. It can be used for maximum workloads, such as web applications. It has a fine balance between price and performance for its actual attributes. It can have a volume size between 1GB to 16TBs and a maximum *IOPS* of 16,000 per second.

Provisioned IOPS (SSD) is suitable for the high throughput or high SSD performance for *Mission-critical low-latency.* It is used for fast input and output operations. It is not just *IOPS* but also high throughput, and it is great for large databases such as RDS. If you surpass 16,000 IOPS, which is the limit for general-purpose, you should start using Provisioned IOPS. Its volume size can be between 4GBs and 16TBs and have maximum IOPS of 64,000.

Then we have *Throughput Optimized* and Cold *HDD*, which are the hard disk drives. The Throughput Optimized HDD is an inexpensive volume type designed to frequently access data and heavy throughput workloads. It is suitable for data warehousing, big data, and log processing, where there is plenty of data. By default, its volume size is huge, ranging from 500GBs to 15TBs, and the maximum IOPS is 500. On the other hand, Cold HDD is used for the workloads that occur less frequently, and it is the cheapest hard drive available. This storage option is best for keeping backups and file storage for a user, having the same volume size as the throughput optimized HDD, with a maximum IOPS of 250. Lastly, we have *EBS Magnetic*, which is very low-cost storage for long-term *archival storage.* There are 500GBs and 15TBs and a maximum of 40 to 200 IOPS. It normally uses previous models of hard drives to give you a low cost.

Features of EBS

AWS EBS has strong features that facilitate the automated and reliable storage of persistent data and optimize cloud storage cost investment. These features are discussed one by one.

Snapshot Backups

Snapshot backups are the most valuable feature of EBS while still being the most difficult to comprehend. You can capture the snapshot of EBS volumes whenever required. This feature creates a copy of the EBS volume's data to S3, where it is stored redundantly in several AZ (like all data in S3). There is a bit more effort involved in moving volumes across regions. It begins with the same process by creating a snapshot and generating AMI from that snapshot. However, you must copy the AMI to get into another region and then launch the EC2 instance. The Snapshots are considered incremental backups that can be extended, copied, transferred, exchanged, updated, handled, and organized within all AZ. They are accessed through the EBS API rather than being stored as user-accessible objects. You can store them behind the Amazon Machine Image, which contains necessary information about the data recovery and instance launch in the cloud. They can be used for disaster recovery, data backups, and collection of production data for development and testing purposes.

EBS-Optimized Instances

For storage workloads that involve short and intensive periods of high I/O activities, EBS Optimized Instances provide a burst of performance improvements. Depending on the instance type, the throughput efficiency for EBS-optimized instances lies between 4250 and 14,000 Mbps. With this capability, the low spec instances can replicate the high performance of larger instances for a short time during a day. This feature enables you to scale your instances to meet EBS demand spikes properly. This optimizes EBS volumes for a number of storage use cases, and customer experience is unaffected by demand spikes.

Encrypted Root Volumes

Let us discuss the process of *encrypting* the *Root Volumes*. The *launch wizard of the EC2 instance* enables you to encrypt root volume while creating an instance**.** The volume type is considered as your root here. Earlier, it was not possible to encrypt a volume on creation, but now it is achievable. In order to apply encryption to an unencrypted volume that you created, the snapshot of the unencrypted volume can be taken and then copied to generate another snapshot. In this way, you will be able to encrypt the volume with this snapshot. For an encrypted snapshot, you must encrypt the copy and spin a new instance with this encrypted AMI. Afterward, a new EC2 instance will initiate from that AMI, and the root volume will be encrypted.

Instance-Backed Block Storages

When an EC2 instance is launched, two types of volumes (block-level storages) can get attached to it. One is the *EBS volumes*, and the other is *Instance Store Volumes.* The users use EBS volumes 99.9% of the time, except for some scenarios where they need to use instance store volumes. EBS volume is an external block device connected to a single instance. In contrast, the Instance Store volume is temporary storage mounted on disks. Instance Store volume is physically attached to its host machine, while EBS volume is not. It means one is long-lasting and durable, whereas the other is short-term and temporary. Ephemeral is another term used, for instance store volumes, which means ‘lasting for a very short time.’ The EBS volumes are created from EBS snapshots, while Instance Store Volumes are made from stored templates in S3.

The way EBS volumes are used can also affect the instance behavior. When you start or stop an EC2 instance, the data in its attached EBS volume persists on starting back up again. On the other hand, an instance cannot be stopped with instance store volume because it is temporary storage, and you will lose the data by terminating the instance. It can be rebooted, but you would not be able to stop the volume. Thus, terminating and rebooting are the only two options available, for instance store volumes.

Moreover, status checks are performed when an EC2 instance is launched, and anything passing through it is like a host check. Therefore, in case of failure, you can lose the data in it. However, while spinning up an EC2 instance, there is no need to worry about any data.

If we talk about use cases, EBS volumes are perfect where preserving data is the requirement. Mostly, EBS backed volumes are used for the EC2 instances. For Instance store volume, it is perfect for a temporary backup to store an application's cache, logs, or random data. In this way, when servers are out of order, you can move or preserve the data. Thus, the above discussion unfolds the difference between two instance-backed volume types.

Use Cases

- **For NoSQL database**-It provides the low-latency performance for NoSQL databases and the required dependability for best performance.
- **For Relational database**- For the fulfillment of changing storage requirements, EBS is able to scale up or down, which makes it a more suitable option to deploy databases such as MySQL, MS SQL Server, PostgreSQL, or Oracle.
- **Testing and Development**- Provision, scaling, replication, and archiving of production, testing and production environments are achievable with EBS.
- **Business consistency**- Copy the AMIs and EBS Snapshots to execute the applications in various regions of AWS. It will accelerate the recovery time and reduce the data loss by providing backup of the data and log files across geographies on a regular basis.
- **Enterprise-wide application-** With the help of block storage, it

can fulfill a number of business computing requirements, which support the essential user applications, including Oracle, MS SharePoint, or MS Exchange.

ELASTIC FILE SYSTEM

Introduction

Elastic File System (EFS) is a shared file storage service for EC2 instances where a single file system can be connected to many EC2 instances and read or write files stored in it. It is a scalable cloud-native Network File System used for Linux-based applications and workloads that you may use in combination with on-premise resources and AWS cloud services. It requires no provisioning of storage, so that the lack and handling of disc space are less of a problem. In this file system, storage capacity grows up to petabytes and shrinks automatically depending on the user's data storage**.** Its file system structure uses the NFSv4 protocol, which reflects a standard on-premise structure and simplifies the files' access and transfer. Its pay-as-you-go model allows you to scale services upon demand which cost around $0.3/GB monthly. Its simplified interface makes it easy to create and setup your file systems. It is similar to EBS, except that with EBS, you can only install the virtual disk to one instance, while with EFS, you can share an EFS volume with two instances.

You can access the file system with the help of multiple instances, so EFS is a great storage solution for applications that run on multiple instances and need parallel data access. It provides an edge to its users that they can have many EC2 instances in the same VPC mount to a single EFS volume. They all share the same drive in a single VPC. To mount the user's EC2 instances with EFS, it needs to install the *NFSv4.1 client.* EFS will produce multiple targets in all VPC subnets to allow you to mount in different subnets or different AZ's. The EFS File System is a regional service, so; all application deployments across different AZ's are able to access the same file systems, ensuring that the storage layer of the application is highly available.

Use Cases

- **Big data Analytics** -It is able to execute the applications of big data that require a lot of lower-latency file accessing, read-after-write operations, and node throughput.
- **For web server and content management support**- This file system can enable web serving applications and content management systems which include websites, blogs, and archives.
- **For Lift-and-shift applications**- It is a scalable, elastic, and available service with which one can move business applications rapidly and easily without any re-architecture.
- **Application development and testing**-In order to facilitate the auto-scaling workloads, a shared file system for sharing files and codes over several compute resources is only offered by the EFS.
- **For Hybrid environments:** The on-premises instances can use AWS EFS to execute a hybrid cloud environment, which greatly extends the possible EFS use cases.

STORAGE GATEWAYS

Introduction

AWS Storage Gateways are used to extend and back up on-premise storages to the cloud. This service allows you to integrate your company's on-premise IT environment with AWS's storage infrastructure very smoothly and securely. You can store data to the AWS cloud in a secure manner as it is scalable and affordable. Virtual machine images are used in the storage gateway to make it easier on your on-premise systems. AWS Storage Gateway supports both *VMware ESXi* and *Microsoft Hyper-V*. When it is set up and activated, the AWS console will be used for creating a gateway. There are two components to link these two elements, which are on-premises and the clock.

Types of Storage Gateways

Let us look at the three storage gateway types. The first one is the *File gateway*, which uses *NFS or SMB*, and stores user files in S3.
Secondly, *Volume gateway* makes the use of *iSCSI*, which is recognized as a backup solution. The last one is the *Tape gateway*, which is for backing up the user's virtual tape library. Let us discuss each of them one by one.

File Gateway

File Gateway enables you to utilize either Network File System or Server Message Block protocol to generate a *mount point* to treat S3 as a local hard drive or local file system. File gateway always seems to extend your local storage onto S3. *Ownership, permissions,* and *time-stamps* are kept in S3 metadata for the objects related to the file. A file can be handled as a native S3 object once it is moved to S3. *Bucket Policies, Lifecycle Management*, *Versioning,* and *Cross-Region Replication* directly apply to the object accumulated in the bucket. You can obtain all the benefits of S3 and get to use it like a local file system or hard drive.

Volume Gateway

The second type of storage gateway is the *Volume Gateway*. It provides disk volumes for your application using the *Internet Small Computer Systems Interface (iSCSI) block protocol.* If you have local storage volume, you can interact with S3 and store a backup of your storage volume as an EBS Snapshot using this protocol through the storage gateway. Since there are two different types, it depends on the usage for the type of volume gateway you select. Thus, data is written to the volumes, and you can have an asynchronous backup for this data as a point-in-time snapshot. You can also save it in the cloud as an EBS snapshot.

There are two different types of volume gateways. The volume gateway for *Stored Volumes* is the first type in which primary data gets stored. This primary data is stored locally when the data is backed up to AWS asynchronously. You can then get backups of your data on AWS. Furthermore, it offers applications running on-premises with low bandwidth access to entire databases and off-site backups that are durable. It creates volumes for storage on your on-premise servers and installs them as iSCSI devices. The stored volumes range lies 1GB to 16 TBs in size. Therefore the stored volume data is stored on the user's on-premise storage hardware.

The second type is *Cached Volumes*. The difference between storage volumes and cache volumes is that the primary data is stored on AWS because the most frequently accessed files are being cached. It is important to remember the location of primary data in stored volumes and cache volumes because it reduces the necessity of scaling on-premise storage infrastructure while providing the applications with low latency data access. It generates up to 32TBs storage volumes and attaches them as iSCSI devices from on-premise servers. The Gateway will store the data you will write in S3 to these volumes and hold the recently read data in your local storage. You can cache the most frequent files, gateway caches, and upload buffer storages. Cache volumes range from 1GB to 32GBs in size.

Tape Gateway (VTL)

The *Tape Gateway* is used for backing up *Virtual Tape Libraries (VTL)* to

AWS. It is the third type of storage gateway and the solution to archive user's data in AWS, which is durable and affordable. You can leverage the current tape-based backup application setup. The data gets stored on virtual tape cartridges created on a tape gateway. Every tape gateway is pre-configured with tape drives and a media changer which is available as iSCSI devices for current user backup applications. Based on the requirement, tape cartridges are added to Archive the data because it supports the different tape services: *Veeam, Backup Exec,* and *NetBackup.* In short, you can store virtual tape libraries on S3 and use S3 Glacier due to its prolonged storage.

AWS SNOW FAMILY

AWS Snowball and AWS Snowmobile are the members of the Snow Family, which we are going to discuss next.

AWS Snowball

AWS Snowball utilizes physical storage devices to move huge volumes of data at a faster-than-internet speed between S3 and on-premise storage sites. It can be sent directly to your company to upload your data volumes to AWS servers. This service offers user-friendly interfaces to create jobs, track your data, and monitor your job progress. It includes an integrated E Ink shipping label (a Kindle screen) that automatically sets itself to ship to the client and then ship back to AWS until the data is completely loaded. AWS KMS protects these physically rugged devices, which secure the data in transit. The Snowball has a maximum data transfer rate of 250 to 400Mbps, which is equal to 2-5 to 3GB per link.

The speed of reads can limit you from your own network, but multiple import tasks can be performed at the same time that will enable you to saturate it. You can avail of 50 TB and 80 TB models of Snowball in the US region; however, only 50TB Snowball models are available in other regions of AWS. If the datasets to be transferred are greater than 50TB, you can order more than one Snowball from Amazon. It will cost $250 for 80TB and $200 for 50TB data transfers, which is quite affordable (one-fifth of the total cost) as compared to using internet pipes to send data to S3. If you buy and upload the data using your own hard drives, it will cost more than getting 50TB of storage for $200 because an 8TB hard drive costs so much. So, Snowball becomes a reasonable choice to transfer the data for various businesses who want to move their datasets to the AWS cloud.

By using Snowball client and protected devices, AWS Snowball accelerates the data transfer at a petabyte-scale to and from an AWS service. The process is very simple, where the console is used to build multiple jobs to order multiple snowball machines depending on the size of data that needs to be transferred. Then the Snowball client is downloaded and installed onto your system. Snowball devices are connected to your local network upon arrival, and the IP addresses are configured either via DHCP or manually. Then the Snowball Client is used to find out which directories you want to copy. Following that, the migration takes place at fast speeds over a safe network (not on the internet). When the transfer is completed, the data is automatically encrypted and copied by the Client into the snowball device, and then an alert is sent to you. For Snowball shipment, Amazon uses an E Ink label, but once the migration is over and you are able to give the devices back, the label switches to reflect the right address for the return shipment. You can monitor the stats of the job via text messages or directly by using console or AWS SNS.

AWS Snow Mobile

It is an Exabyte data transmission service used to transmit a huge amount of data volumes to AWS. This is a truck-pulled, shipping-container that can transfer up to 100PB’s of data per Snowmobile. You can transfer huge data volumes very easily to the cloud, which includes entire data centers, video libraries, and image repositories. It is more secure, affordable, and fast to transfer data with Snowmobile.

Amazon sends a Snowmobile to your on-premises/data center and sets it up as a network storage center for you. After that, multiple fiber channels are used to connect the device with your local network, resulting in a bandwidth of over 1Tbps. The super fast Snowmobile’s network switch is wired into your local area network so that data can be sent from your data center with high speed to the Snowmobile. When the transmission of data is completed, the container will be returned to AWS, which will load the data into S3 or Glacier. A number of security layers are used to secure the data, including GPS monitoring, alarm monitoring, a dedicated security staff, 24/7 video monitoring, and an additional safety vehicle during the data transfer process. The data will be encrypted and secured by using the 256-bit encryption keys controlled by AWS KMS.

CHAPTER 4: DATABASE FUNDAMENTALS

Overview

AWS has a broad range of *Databases* that can satisfy the user data requirements. AWS databases will satisfy the demands, regardless of whether users need a relation, a time series, a graph, a document, in-memory, or a key-value database. All of these databases are able to handle applications that receive hundreds and thousands of requests per second. Amazon can automate every major database operation, including maintenance, upgrades, restoration, and backup. In this chapter, various types of database services are going to be discussed that are available in the AWS cloud platform. At the end of this chapter, you will be able to choose and identify the database that best suits a particular use case.

Before moving into the family of available AWS database services, the dissimilarities between a *relational* and a *non-relational database* will be addressed. After that, we will cover the broad topic of Amazon RDS in detail. It is a fully managed database service where users can choose among common database engines and allow AWS to handle their databases. Then we will discuss the two major Amazon SQL databases, such as Amazon Redshift and Amazon Aurora, along with their use cases in the context of a scenario. Then, we will look at the non-relational databases, which include DynamoDB
, and Elasticache and the use case scenarios for using these types of database services. Let us start the chapter now.

Relational and non-relational databases (NoSQL) are the two groups of AWS database services. You will see an overview of these two groups of Amazon database services in this section.

AWS Relational Database Services

The data in *Relational Databases* is stored in tables in *rows* and *columns* where you can apply queries using *Structured Query Language*. Columns in such databases show the attributes, whereas the rows in the tables represent the records. Every field is a data value in the table. Amazon RDS, Redshift, and Amazon Aurora are the primary services that offer relational database services.

The use-cases for Amazon relational databases are:

- Transactions
- CRM applications
- Data warehousing
- Finance data
- Enterprise Resource Planning (ERP)apps

AWS NoSQL Database Services

The relational databases are not good for those use cases which require high speed or adaptive scalability in particular. *NoSQL databases* provide the non-tabular flexible database schemas to allow more efficient distribution and processing of data. These databases are generally required to deal with big data – a large volume of semi-structured and unstructured data. They are available in various forms on the basis of their data model. *Amazon Dynamo DB* and *ElastiCache* are the two types of managed NoSQL databases. Amazon Dynamo DB refers to a *Key-value Database* used to store the data as a series of key-value pairs with a key that serves as an ID. Different data types, such as plain and compound objects, can be stored in them. Customer preferences, e-commerce shopping carts, Real-time binding, and Product catalogs are the use cases of Dynamo DB. Based on your application, it is a managed non-relational database, which can be a more suitable solution than a traditional SQL-based database. Amazon ElastiCache, on the other side, is an *In-memory database type* in which the data is stored in the memory to ensure low-latency access. These stores can be used as caches, message brokers, databases, or queues. The ElastiCache can be used for the following applications:

- Caching
- Session stores
- Real-time streaming
- Leaderboards
- Gaming
- Pub/sub messaging
- Geospatial services

The other services for AWS databases are *AWS DMS* for database transfers and migrations, *Amazon Neptune* for graph databases, and *MongoDB* for document databases.

Let us proceed towards our first database now, which is known as Amazon RDS.

AMAZON RDS

Introduction

RDS stands for Relational Database Service, a cloud computing solution in AWS to provide help in configuration, launch, and scaling of a relational database instance. RDS is the managed AWS solution for relational databases that supports several SQL engines**.** Amazon RDS manages many time-consuming database management tasks, including migration, patching and backup, and recovery, as a completely managed service. It also provides security and backup for the user databases. It can also backup RDS instances automatically, take a regular snapshot, and save the transaction logs for enabling recovery point-in-time. RDS also patches the software of the database engine automatically. It allows replication for high availability of production workloads and increased reliability. The automatic failover in multiple AZ's can also be enabled with synchronous replication of data. You can control the Amazon RDS through AWS Management Console, AWS CLI, or its own APIs. So there are currently six relational database options for AWS. They are *MariaDB, Aurora, Oracle, PostgreSQL, Microsoft SQL, and MySQL.*

Being a database administrator, you can create, manage, delete, and launch an RDS instance. RDS instance is a cloud-based database environment that uses storage and compute resources. Based on the database engine, you can spin up several schemas or databases. Each customer can use a total of 40 Amazon RDS instances per account. In the case of SQL Server and Oracle instances, you can have only ten of each.

Features of Amazon RDS

Availability

You need to access the hosted data in the cloud at the time or the location of your choice. With the help of the Multi-AZ feature, RDS can provide high availability to maintain the redundant copy of your data in a different location. This Multi-AZ service level agreement will guarantee the database uptime of at least 99.95 percent a month.

Scalability

Scaling a self-hosted in-house database can become very challenging, but Amazon RDS has made it very simple. It provides two types of automated scaling: Horizontal scaling (introducing more machines into your infrastructure) and Vertical Scaling (introducing more power to your existing machines). In case the database gets overburdened with the requests, this service provides a load balancer that distributes the requests equally.

Performance

With the help of the Performance Insight dashboard in Amazon RDS, you can easily troubleshoot and analyze relational databases' efficiency.

Encryption

All RDS engines can be encrypted at rest. By turning on the encryption, automatic backups, snapshots, and read replicas associated with the database will also be encrypted. AWS KMS is used for encryption where the default key or KMS key is used to turn it on. However, enabling encryption in older versions of certain engines is not possible yet.

RDS Backups

Let us look at the *Backups* of *RDS*. There are two solutions to perform RDS backups in AWS. They are automated backups and manual snapshots.
In *Automated Backups*, the backup value can be set to 7 to enable it and set it to 0 to disable it. The retention period of 1 to 35 days must be selected in automated backups. The transaction logs are stored here the whole day, and all the data will be stored in S3. No extra cost will be charged for these backups. You can define your required backup through a backup window. In a backup window, you have to specify the starting time, UTC, and then the storage. The time cannot exceed half an hour (0.5). During backups, the IO and the storage may be blocked so that you may face some problems during this time. So you might want to choose that time carefully.

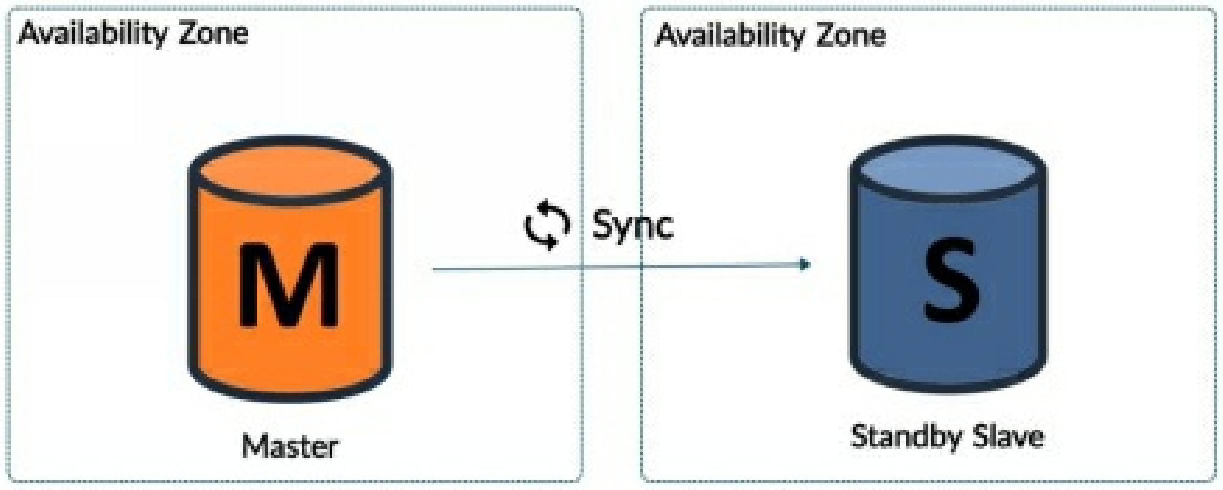
On the other hand, the *Manual Snapshot* creation is a manual process in which you can drop down actions and take a snapshot. You can still have these snapshots even if the primary database or the RDS instance is deleted. So if you leave the RDS system, they do not disappear. It is also possible for you to restore the previous version of the snapshot.

Restoring Backups

Let us now understand the method to *restore* a *backup*. It is the simple step of selecting restore to a single *point in time* from the drop-down actions menu. AWS then takes the recent daily backup while restoring it and applies the transaction log data related to that day. As a result, the point in time recovery will be reduced to a second during the retention period. Data backups are never restored over an existing EC2 resource. Once an automatic backup or manual snapshot is restored, a new instance must be created for the recovered database. So, a new DNS endpoint is required when you create a new instance. You have to do it manually because you need to uninstall the previous instance and use this new endpoint for your applications.

Multi-AZ deployment

Let us now discuss the *Multi-AZ deployment*. It guarantees the availability of your database if another AZ is not available. This allows users to copy the database in another AZ, and AWS directly synchronizes all changes from the master DB to the standby DB automatically. The master DB is the primary database, and standby is a slave database that does not receive real-time traffic. In the case where AZ goes down, standby is simply a backup to replace the master database.



Assume that you have enabled automatic failover protection, and it will occur when our AZ goes down. Thus, it will point to the slave database, and the slave will be upgraded to the master database. Therefore, this database is now the master database.

Read Replicas

Read Replicas can be used to run several copies of your database. The write operations can not be performed to these copies, but you can read them. Read replicas are designed to reduce the primary database workload in order to maximize performance. You must enable automatic backups in order to use read replicas. So creating a read replica is very simple; you drop down actions and click "create read replica'. The process of replication between read replicas and the master database is asynchronous. It allows you to have five replicas of your single database. Each read replica has a specific DNS endpoint. There is no automatic failover for read replicas, so if the primary replica fails, you need to change URLs manually to point at copy.

You can have the *Multi-AZ Replicas* and *Cross-Region Replicas.* You can even create replicas of replicas. You can promote read replicas to your master database, but this will break the replication. The read replica acts as a database instance, allowing only read links so that applications can interact with these replicas, just like any DB instance connection. Once the replica becomes its own master, it does not receive updates from its old master. Promoting one read replica does not affect the relationship between its old master and other read replicas.

Comparison between Multi-AZ & Read Replicas

Now let us differentiate the Amazon RDS Multi-AZ deployment and Read-replica.

- For replication, Multi-AZ has synchronous replication, and Read-replicas have asynchronous replication.
- The primary instance database is going to be active. The primary instance will be the instance currently active on Multi-AZ. Therefore the standby does nothing. It just becomes the primary instance in case the primary instance goes down, which uses all the replicas, read-replicas, and primary instances.

- You can have automatic backups taken from the standby database for multi-AZ, whereas no backups are enabled by default for read replicas.
- Multi-AZ is followed by two AZs within a single region. Read-replicas are available in one AZ but can be multi-AZ, Cross-AZ, or Cross-Region**.**
- When upgrades occur in multi-AZ deployment, they will be in a primary database. On the other hand, when upgrades occur in read replicas, they will be independent of the source instance.
- The automatic failover can occur on standby in multi-AZ deployment. DB instance Failover here is completely automatic and does not require administrative action. You do not have this automated failover for reading replicas. Each of these replicas must be manually promoted to become the standalone database instance.

AMAZON AURORA

Introduction

Aurora is a completely maintained Postgres or MySQL compatible database service. It is designed to scale according to the workload and is very fast by default. High-end database availability and speed are combined with the open-source database's convenience and cost flexibility in Aurora. It can run on engines that are compatible with MySQL or Postgres. The advantage of using Aurora over a standard RDS Postgres or MySQL engine is that it is fine-tuned to deliver the best performance. If you use MySQL, Aurora is five times faster than the traditional MySQL, and its Postgres version is three times more efficient than the traditional Postgres. The cost is one of the biggest advantages, as it is a tenth of the price of other database solutions that provide similar performance and availability.

Aurora is a corporate-level database service that can scale up to 64 TBs in terms of performance and availability. It has a special feature called *Multi-AZ deployment* that allows it to replicate data across different availability zones. Depending on the use cases, you can select from various types of hardware specifications for your instances. *Serverless mode* is another feature of Aurora which provides a complete on-demand experience in which the database scales down automatically for the lower loads and scale-up for the higher loads. In this mode, you are only charged for the time in which the database is operational. However, a small delay will be there in responding to requests when the database is fully scaled down.

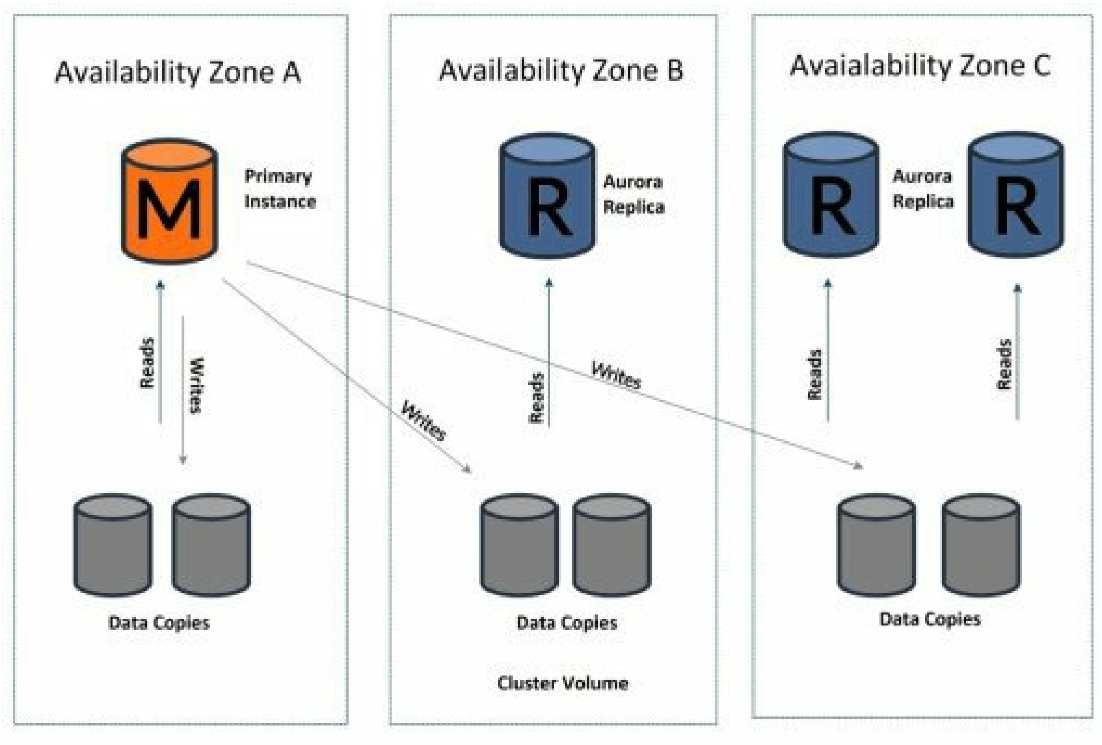
Aurora architecture's working depends on a volume cluster that handles all DB instance data in that specific cluster. A volume cluster is essentially virtual DB storage that is shared across several AZ's. The underlying storage volume is located on top of several cluster nodes distributed around various AZ. Apart from this, the Aurora database may also provide Read Replicas.

Generally, only a single instance works as a primary instance that supports read and write operations. In contrast, the remaining instances act as read-replicas, and the user is responsible for load balancing. It differs from the Multi-AZ deployment, in which the instances are located in the AZs and support automated failovers.

Features of Aurora

Availability

Aurora is a highly available database service because it replicates your data six times across three AZ’s, each with two copies. If you lose two copies of your data, it will not affect the write availability. If you lose three copies of your data, it will not affect read availability**.** To avoid storage failure and for safety purposes, AWS Aurora provides the data backup on a constant rhythm.



Higher Security

Aurora provides the database with several layers of security to make it better than the other services. Data in the underlying storage is encrypted on an encrypted Amazon Aurora instance. The management is done via the AWS KMS, and SSL is used to encrypt the data in transit. In addition, replicas, automatic backups, and snapshots are available in the same cluster.

High Scalability and Performance

Aurora Scaling is a managed feature that starts with 10GBs of storage initially and can scale up to 64TBs with increments of 10GBs. Auto-scaling is enabled as far as storage is concerned, which automatically scales up or scales down the storage. The computing resources can scale up to *32 vCPUs* with 244GBs of memory for its computing power.

Aurora allows both horizontal and vertical scaling. You can perform vertical scaling by upgrading the instance types, and there is a minimum downtime associated with it in the case of Multi-AZ deployment. If this is not possible, then you can schedule the scaling during the database maintenance time. On the other hand, horizontal scaling is performed through the read replicas where the Aurora database can have a maximum of 15 read replicas simultaneously. There is a difference between storage scaling and Aurora compute scaling, and the one mentioned above is compute scaling. The storage scaling in Aurora is performed by modifying the maximum allotted storage space or hardware storage type such as HDD or SSD.

Aurora offers five times the throughput of the standard MySQL database. Such performance is good for enterprise databases at the rate of 1/10th price. As your needs change, you can simply scale your database preparation up to and down from smaller to larger instance options.

MySQL and PostgreSQL Compatibility

Aurora engine is completely compatible with the PostgreSQL and MySQL databases. Compatibility for the latest versions of the database is introduced on a regular basis, and you can also move the data from PostgreSQL/MySQL to Aurora using simple import/export tools. Your current applications that use PostgreSQL or MySQL can also be used with Aurora without requiring any code modifications.

Fault Tolerance & Durability

Aurora automatically manages backups and failovers when it comes to *Fault Tolerance* and *Durability*. Snapshots can be shared whenever data has to be exchanged with another AWS account. Aurora also comes with self-recovering storage, so data blocks and disks are regularly checked and repaired automatically.

Replicas

There are two types of replicas available in Aurora, including *Amazon Aurora replicas* and *MySQL Read Replicas.* You can have five read replicas for MySQL, and there is a high-performance impact on the primary database. It does not have automatic failover, yet it supports user-defined replication delay and various data/schema against the primary. On the other hand, you can have 15 *Aurora replicas*, and there is a low-performance impact on the primary database. It supports automatic failover but does not support any user-defined replication delay or different data/schema.

Moreover, the Aurora database can span into multiple regions via *Aurora Global Database***.** Thus, you need to select the one that is more suitable for your enterprise. From an exam point of view, one must know the two different Aurora replicas types.

Aurora Serverless

Aurora is very expensive if it is not used for high production applications. However, if using Aurora is still a requirement, Amazon has a feature known as *Aurora Serverless.* It is just another Aurora mode that only runs when the user needs it to run and can scale up and down, depending on the application's requirement. The capacity settings are required when the serverless option is set in the database features, and the capacity for *Aurora Capacity Units* (*ACU*) can be set to a minimum and maximum. It lies between 2 and 384 ACUs, which only charge you when it is consumed. Aurora Serverless is used for low-volume blog sites such as for a chatbot or to build an MVP that is out to clients. It is not frequently used, but you can plan to use Aurora in upcoming times.

Use Cases

It is designed for various use cases which can manage virtually any scenario in which there is a need for relational databases.

Enterprise Application

Aurora has enough robustness to deal with any business application that uses a relational database. It reduces the cost by 90% compared to traditional organizational databases and enhances the availability and reliability of a database.

SaaS Applications

SaaS applications are often designed to deal with multi-tenant users. So, Aurora is perfect for this use case as it is highly reliable and provides high performance along with storage scaling.

Web and Mobile Gaming

For the large mobile games and web applications designed to work on a larger scale, Aurora is the most suitable option. This database service meets the requirement of modern mobile and web games, which include huge storage scalability, high availability, and higher throughput.

When to use Aurora?

- Getting rid of the administrative tasks required for database management while staying with the query layer compatible with MySQL/Postgres.
- Using conventional databases such as Postgres and MySQL when you need better read performance at the cost of slower updates and lower writes.
- Your storage requirement is only in the TBs (not in 100sTBs data in the near future).
- There is a requirement of an online use case of transactional processing with immediate results and fewer data.
- The analytical workloads of your OLTP operation are not disrupted.
- When there is no need to process multiple rows of data by the analytical workload.

REDSHIFT

Introduction

Amazon Redshift is a completely managed data warehouse of *Petabyte-size*, which is used to analyze the huge data size using complex SQL queries. It is necessary to understand the term data warehouse to comprehend the process of Redshift because Amazon Redshift is a columnar store database.

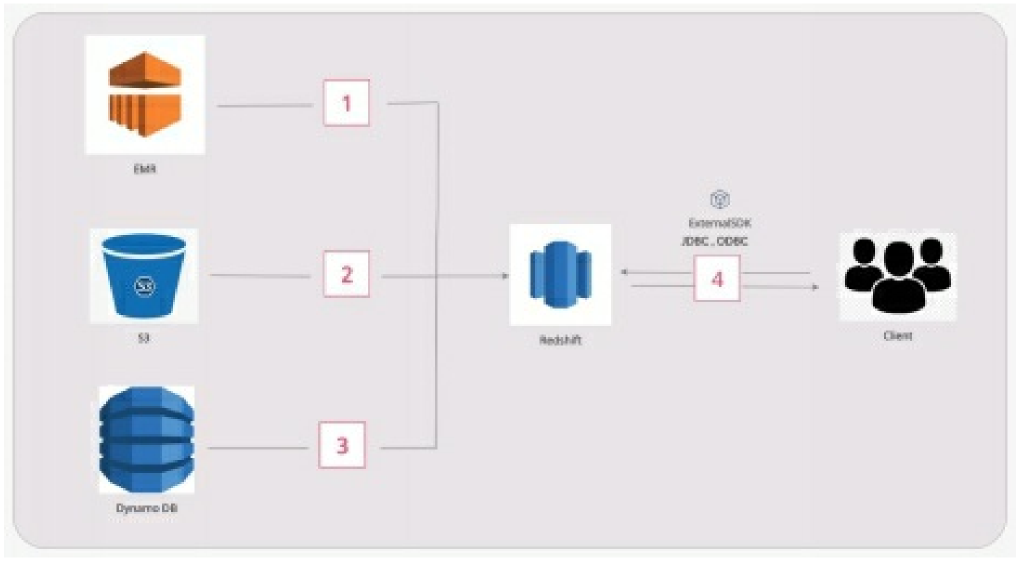
Let us compare the term data warehouse with the database to understand it better. Therefore, you need to have some basic knowledge and understanding of database transactions. It is defined as: 'A transaction represents a unit of work done within a database management system.**'** For instance, reads and writes are an example of a transaction. Transactions are treated differently for databases and data warehouses. *OLTP* is an online transaction processing system for a database in which transaction is short, which means small and simple queries that focus on writes. In OLTP, you can create a database for storing existing transactions and enabling fast access to some transactions for ongoing businesses.

Consider a web application that needs to be very responsive regarding reads and writes for the current users. Read and write operations may include adding an item to a shopping list, signing up, or any operation in the web application. Generally, these operations are backed by a single database, Postgres, or might be running on RDS. The Data warehouse runs on an Online Analytical Processing System (*OLAP*). OLAP involves lengthy transactions, such as long and complex SQL queries that focus on read operations. Therefore, a data warehouse is built to store a huge amount of historical data that enables fast and complicated queries across all data. It is used for generating reports and Business intelligence tools. A data warehouse is not a single source because it takes data from multiple sources. The data is combined from DynamoDB, EMR, S3, and Postgres in one place to run complex queries.

You will prefer Redshift because the pricing starts at 25 cents per hour, with no capital costs or commitments. It scales up to petabytes of data for $1000 per terabyte a year. Redshift price is less than 1/10th the cost of most similar services. It is used for business intelligence, and it uses OLAP. It is a columnar storage database that's crucial for improving the performance of analytic queries. The size of the data to load from the disk and the overall disk IO needs instantly reduces. Redshift is fast due to columnar storage. Let us discuss it in more detail in the following use case.

Use Case

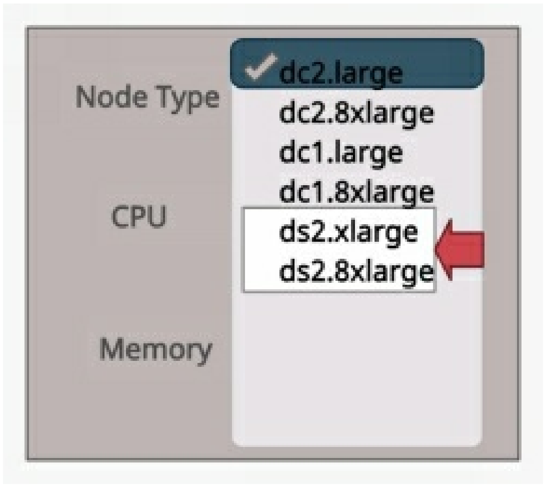
Let us discuss a use case to understand the process of Redshift. Consider that you want to build a business intelligence tool. You have many different sources, and the data is coming from EMR, S3, and DynamoDB, which you will copy into Redshift by using the copy command.



You can normally interact and access Redshift data using AWS SDK as most of the services do. However, in this case, you do not use any AWS SDK because you may need to make a generic SQL connection to Redshift. Moreover, if you are using Redshift, you must be using java. Therefore, you would use *JDBC* or *ODBC,* third-party libraries, to connect and query Redshift data. As mentioned earlier, columnar storage is very important to Redshift performance. So, let us understand its concept here. *Reading via the rows* is normally used with a database, whereas in an OLAP, it is *reading via* *the columns* because it is better to look at columns rather than looking at huge data and breaking it. OLAP applications look at multiple records at the same time. You can save memory because you only fetch the columns of data you need instead of whole rows. If you read columns, the data can be stored as the same data type, which allows easy compression. It means that you can load the data more quickly. Furthermore, you always look at large volumes of data simultaneously, so only columns needed in bulk can be retrieved. Thus, it gives much faster performance for a use case, such as the business intelligence tool.

Redshift configuration

Redshift configurations can be set up in two different cluster types. They are single node and multi-node. A single node is a suitable approach to start on Redshift. You can launch a single node of 160GB, or you can simply launch a multi-node. While launching a multi-node, you can have a leader node and 128 compute nodes. So, there is huge computing power for a user. When you spin Redshift multi-node, you can have a maximum set of 32 nodes, while previously, it is mentioned that anyone can have 128 nodes. So, if you need more nodes than 32, you can ask AWS for a service limit increase of 128 nodes. Apart from different cluster types, there are two different types of nodes as well. They are *Dense Compute* (*DC*) and *Dense Storage* (*DS*). As the names indicate, DC is optimized for computing power, and DS is optimized for storage. So, you can choose the node type according to the desired use case.



The image shows that there are no small or micro nodes. It can only be started with a large node because you only deal with a large amount of data with Redshift.

Redshift uses multiple *compression techniques* to achieve large compression that is relative to traditional data stores. Compression is very important in terms of speed. Identical data is stored sequentially on disk, so it does not require any indexes or materialized views. It saves greater space compared to the traditional systems. The data is sampled when it is loaded to an empty table, and the most suitable scheme is selected automatically.

Redshift Processing

Redshift uses *Massively Parallel Processing* abbreviated as *MPP*. It automatically distributes the data and query loads among all nodes and allows you to easily add nodes in the data warehouse while maintaining fast query performance. So, it is easy to add compute-power on demand. Then there are Redshift backups that are enabled by default with one day retention time. This retention time can extend up to 35 days. Redshift always tries to preserve at least three of your data copies, out of which the first is original, the second is the replica on compute nodes, and the third is the backup copy in S3. Redshift can replicate your snapshots asynchronously to S3. So, you also have the option of moving data from region to region.

Availability

The availability of Redshift is in *Single-AZ* rather than *Multi-AZ*. To run Redshift in multi-AZ, you need to run multiple Redshift clusters in a different AZ with the same input. Therefore, you have to manually run a clone because there is no managed or automated way to do multi-AZ. In case of any malfunction, snapshots can be restored to another AZ.

Billing

There are *compute node hours* for Redshift billing in which the total number of hours run across all nodes in the billing period. It is charged as one unit per node an hour rather than charging you for the leader's node per hour. As you have only one compute node and one leader node, you will pay for the compute node only when you spin up a cluster. The Redshift backups are stored on S3 so, you are charged for the S3 storage fees. Data transfer is only charged for transfers within a VPN. For Redshift security, there is *Data-in-*

transit encryption using SSL, and then there is *Data-at-rest*, where you can encrypt data by using AES-256 encryption. Therefore, Database encryption can be applied using KMS or *CloudHSM*, which is as easy as applying it.

DYNAMODB

Introduction

Let us look at *DynamoDB*, which is a *key/value* in the document database. It is a *NoSQL database* service used by applications requiring stable latency (in milliseconds) at any scale. It can guarantee consistent reads and writes at any scale. The NoSQL database is a type of database that is neither relational nor uses SQL for querying the data. This database service uses key-value store, and document store.

A key/value store is based on the key-value model in which you have a key and a value, and they can only search by the key, whereas the document store is a document model where your data is more structured than KV models. The whole data is a single value in the database. Thus, the DynamoDB is a NoSQL key/value in the document database with many features for internet-scale applications. This database service is multi-feature, multi-region, durable, and multi-master with built-in protection, memory caching, recovery, and restoration.

DynamoDB provides developers with minimum configuration and management while offering high efficiency and scalability. It serves as a key-value-store database for NoSQL, which reflects that the data (JSON objects) is not stored in a relational or structured mapping, but it is stored in the format of simple key-value. In order to provide high data durability and availability, the data in DynamoDB is stored on SSD's and replicated synchronously in several AZ's in a region. Moreover, DynamoDB is a user's choice because it satisfies all the user requirements. For example, if you need a hundred reads per second or a hundred writes per second, you will surely get it using DynamoDB. It depends on the payment, so scaling this database is not an issue. Regarding durability, DynamoDB stores its data across three regions, and you can have fast reads and writes as it uses SSD drives.

How does it work?

DynamoDB stores the data in tabular form, just like other databases. There is a set of Items in each table, and there are many Attributes in each item. The DynamoDB table must contain a primary key in every table item, which is a single attribute or two of them, such as sort-key and partition-key. Using the primary key or making your indexes and using keys from those indexes, you can reference certain items in a table.

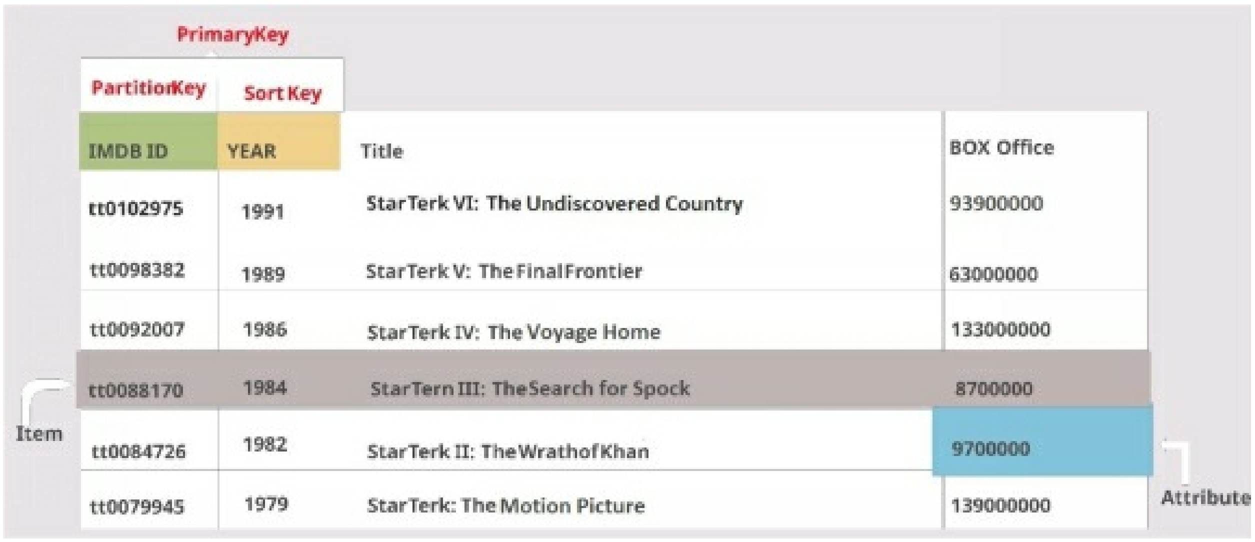
In the DynamoDB table, your data is shared over several devices instead of having a single hosting server, so you are guaranteed high performance and scalability. It also means that you cannot link directly to a database host and query the data. The DynamoDB HTTP API is used to write and read items from and to a DynamoDB table directly or via AWS SDK/CLI. You can also batch reads and writes to DynamoDB tables, even through various tables at once. It supports cross-region replication, transaction, and automatic backups.

Table Structure

It is important to understand the *Table Structure* of DynamoDB because it does not use the same terminologies as a relational database. In DynamoDB, there are three basic data model units, *Tables*, *Items*, and *Attributes*. The term *Item* is used instead of a *Role* and Attribute instead of a *Column/Cell*. Then we have the *primary key*, which is made up of two keys:

- Partition Key
- Sort Key

These table elements of DynamoDB can be shown in the following diagram.


PrimaryKey
PartitionKey
Sort Key
Item
Attribute

IMDB ID	YEAR	Title	BOX Office
tt0102975	1991	StarTerk VI: The Undiscovered Country	93900000
tt0098382	1989	StarTerk V: TheFinalFrontier	63000000
tt0092007	1986	StarTerk IV: The Voyage Home	133000000
tt0088170	1984	StarTern III: TheSearch for Spock	8700000
tt0084726	1982	StarTerk II: TheWrathofKhan	9700000
tt0079945	1979	StarTerk: The Motion Picture	139000000

Consistent Reads

Consistency is important while using DynamoDB because when data is written in the database, it is copied to the other regions. Reading your copies of data may lead to inconsistency. If someone is reading for region C when the update occurs, there is a chance that you are reading it before it is updated (written). DynamoDB gives a couple of options for consistent reads based on the user requirement. The first option is *Eventual Consistent Reads*, which is the default functionality. When copies are being updated, you can read and return an inconsistent copy. Eventually, all copies of data will become consistent within a second. The drawback here is that the reads are fast without any guarantee of consistency.

The other one is *Strongly Consistent Reads*. In this option, all copies are updated, and then you will be able to read them. It will not return a result until all copies are consistent. You have a guarantee of consistency, and the only drawback is a high level of latency. There will be slower reads, but the reads will be as slow as a second because all copies of data will be consistent within a second. Users can wait up to a second in the case of a write operation. If they can tolerate something consistently if it is not important, then those are the two options.

Features of Amazon DynamoDB

Scalable-In order to fulfill your throughput and storage requirements, DynamoDB is designed to scale the table’s resources to hundreds and thousands of servers spread over several AZ’s. You can store the data in a table without any pre-defined limits.

Durable and Highly Available- With Dynamo DB, the data is copied to at least three data centers, which allows the system to continue its operation and serve the data even in a complicated scenario.

Flexible- Dynamo DB does not force you into a single data format or consistency model due to its exceptional flexibility. There is no fixed schema in the Dynamo DB table, and you can have the desired number of attributes in every table item, such as multi-valued attributes.

Fast- With Dynamo DB, you can have high performance and throughput with very low latency. It is built on SSDs to help optimize higher performance and high scalability. Furthermore, the read and write operations cost is minimized when the index is not created for all attributes. Only the primary key index is updated in the write operation, which results in the reduced latency of both reads and writes.

Storage -Its storage charges are $1/GB a month, and the request price depends on the amount of capacity reserved. So the cost for every ten units of writing capacity is $0.01 per hour and $0.01 for every 50 units of reading capacity.

Use Cases of DynamoDB

Let us go through some of the common scenarios in various domains in which DynamoDB seems to be an effective approach.

Use cases in Ad Tech

- Metadata stores for assets
- storage of user-profiles in Ad targeting and Real-Time Bidding
- Clickstream, impression data store, and user events
- Caches for common items

Use cases in Finance and Banking

- Detecting frauds

- User transactions
- Change data capture and mainframe offloading
- Transaction processing based on Events

Use cases in Gaming

- Player datastore
- Leaderboard
- Game states
- Player session history data stores

Use cases in Retail

- Workflow engines
- Customer accounts and profiles
- Shopping carts
- Inventory management and delivery

Use cases in Software and Internet

- Metadata caches
- User content metadata stores
- Driver, user, and vehicle data stores
- Relationship graph data stores
- User vocabulary data stores
- Ride-tracking data stores

Use cases in Media and Entertainment

- User data stores
- Media metadata stores
- Digital rights management data stores.

AWS DATABASE MIGRATION SERVICE

When you have selected the most suitable service for your requirements, you need to determine the data type and its process of migration. This process involves creating a backup of your current database and transferring it to the latest database service. It may need some re-scheming or reformatting of data. To make data migration simpler and easier, AWS Database Migration Service is available for AWS users. In the first six months, this service is free of cost for several services, which include Dynamo DB, Redshift, and Aurora. This service helps you to keep the databases running during migration that ensures minimum downtime. It also reduces productivity disruptions and revenue loss. To make the migration easier, AWS DMS supports most open-source and commercial databases. It includes both homogenous transfers and mixed transfers such as SQL to SQL Server and Oracle to Amazon Aurora. It can be used to constantly merge and replicate databases to a data center using Amazon Redshift and S3.

CHAPTER 5: MANAGEMENT AND MONITORING SERVICES

Overview

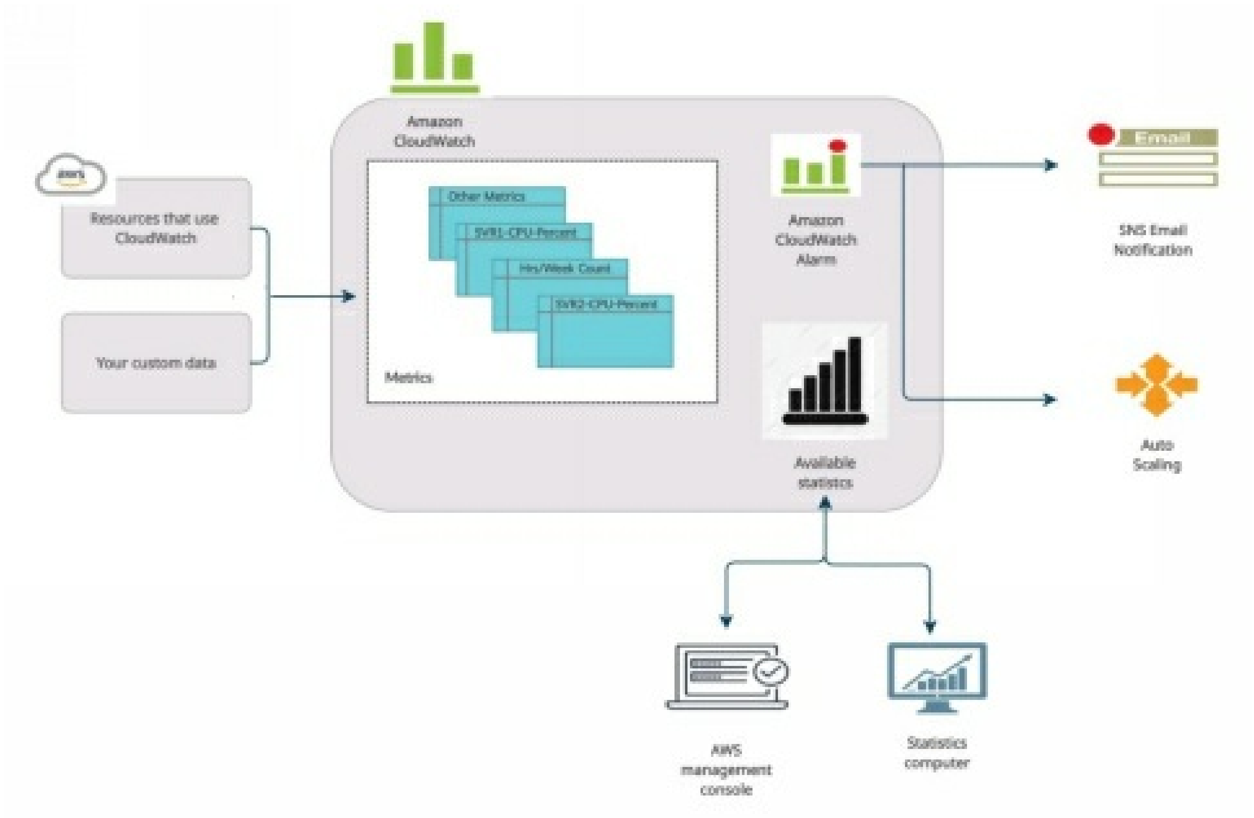
AWS Cloud Monitoring Services use various automated and manual tools. These tools are used to analyze, manage, and observe the cloud computing architecture, cloud framework, and Amazon web services. It integrates the complete cloud management strategy, which allows cloud administrators to monitor cloud resource status. The three basic Amazon Management and Monitoring services will be our focus for this chapter. These services include CloudWatch, Cloud Trail, and Cloud Formation. Let us discuss CloudWatch in detail first.

CLOUDWATCH

Introduction

Amazon CloudWatch enables its users to monitor different applications and cloud resources running on Amazon cloud infrastructure. It enables you to observe EC2 instance metrics and track them, such as EBS volumes and RDS DB instances. Besides, it can perform different functions, such as setting up alarms, storing log files, viewing graphs and statistics. Moreover, it can also be used to monitor changes in AWS resources and respond to them. Amazon CloudWatch provides an overview of the system's health and performance, which is eventually used to enhance an application's functionality. An advantage of this monitoring solution is that you do not need to install any additional software. Let us observe the working of Amazon CloudWatch.

The following figure reflects the concept of monitoring in AWS CloudWatch.



For AWS applications and resources, CloudWatch provides full platform visibility. It monitors the files of resources and creates key metrics depending on the log files of the applications. Disk storage, processor latency, processor usage, and network traffic are all key metrics. CloudWatch displays the actual summary of every resource along with system activities, depending on the metrics. CloudWatch also offers insight into the AWS architecture, allowing it to monitor the performance of applications, notice trends, and fix operational issues. Let us suppose that the AWS environment undergoes unpredictable operational changes. So, in that case, it sets up alarms and sends proper notifications.

CloudWatch is a monitoring service that allows its users to respond, log, and observe the log data. Therefore, you have to understand that CloudWatch includes multiple services under one title. It is not a single service. These services include CloudWatch dashboards, CloudWatch alarms, CloudWatch metrics, CloudWatch Events, and CloudWatch logs.

As we have gone through Amazon CloudWatch concepts and its operations, let us discuss some of its services now.

CloudWatch Dashboards

CloudWatch Dashboards offer insight into the use of resources, operational health, and an application's performance. Therefore, customized dashboards are created to display multiple metrics and access images and texts. One can create multiple dashboards along with global dashboards that are used to fetch data from multiple regions. The AWS CLI or Management console is used to create them and then incorporate them into *Infrastructure as a Code* tool such as AWS CloudFormation.

Depending on the customer's preference, texts and graph widgets are introduced to the dashboard to display the metrics. For example, a graph widget is created to reveal CPU utilization for specific EC2 instances. Both new and existing customers can create three dashboards using up to fifty metrics without any extra cost. If it exceeds the limit, you need to pay for each additional dashboard monthly.

CloudWatch Logs

Using *CloudWatch Logs* help in accessing, monitoring, and storing AWS resources’ log files such as Route 53, EC2 instances, Amazon CloudTrail, and few others. The three basic concepts of CloudWatch Logs are discussed below.

Log Events: It is an activity record registered by the applications or the monitored resource.

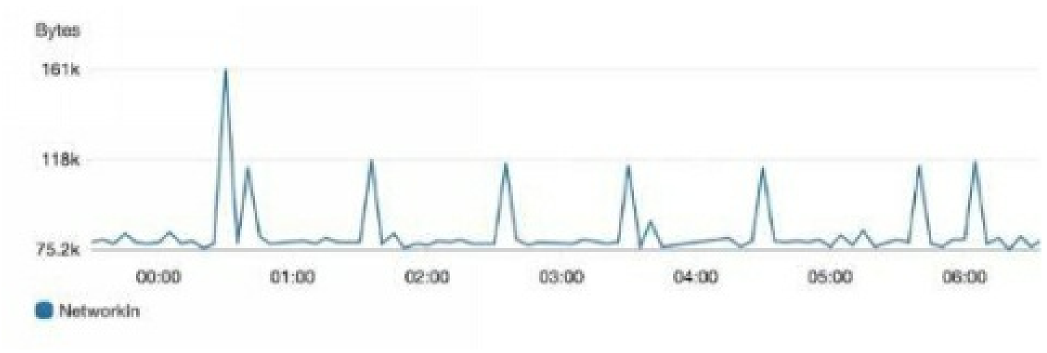
Log Streams: It shows the series of log events from the instance of an application. Thus, log streams are the series of log events with the same resource.

Log Groups: The log group comprises a set of log streams that share similar monitoring settings, access control, and retention.

Moreover, you can use CloudWatch logs to try and fix machine bugs. The log files are maintained and stored automatically. One can set the alarm so that it sends a notification if there is a bug in the system log. CloudWatch log stores the original data. Therefore, by viewing the original files, you can remove the errors in minutes.

CloudWatch Metrics

Let us discuss *CloudWatch Metrics* that is built on top of CloudWatch logs. CloudWatch Metrics represents a sequenced set of data points. It is also considered a variable for monitoring. Therefore, you need to extract the data from logs as data points and graph it. Consider a use case in which network traffic '*NetworkIN*' reaches an EC2 instance. You select that specific metric and observe it by seeing it.

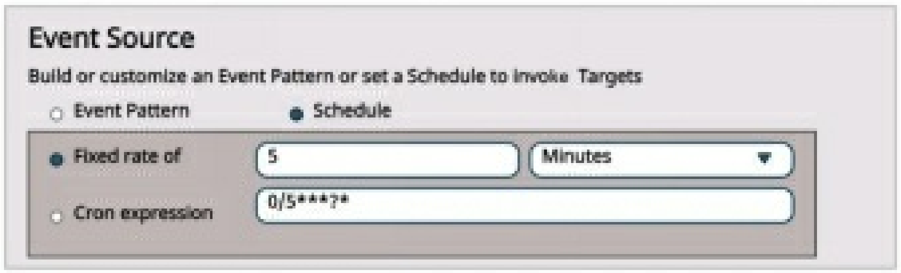


The above figure shows the CloudWatch metrics. These metrics are pre-defined, so you do not need to do anything. To support this, you must enable logs on some services so that when the data arrives, these metrics are available.

CloudWatch Events

They offer real-time delivery of system events from AWS resources to targeted categories such as Lambda, SQS queues, and SNS Topics. They allow you to create a set of rules that are used to match these events, and then these events can be redirected to one or more targets. Let us suppose the same example of the AWS environment undergoes unpredictable operational changes. In this case, CloudWatch Events will record these changes and carry out corrective measures by sending notifications and triggering Lambda functions.

CloudWatch Event depends on CloudWatch metrics and logs. It allows you to respond to your data and act on it. You can also specify an *Event Source* depending upon an event pattern or a schedule that will trigger the selected function in the target.



Event Source

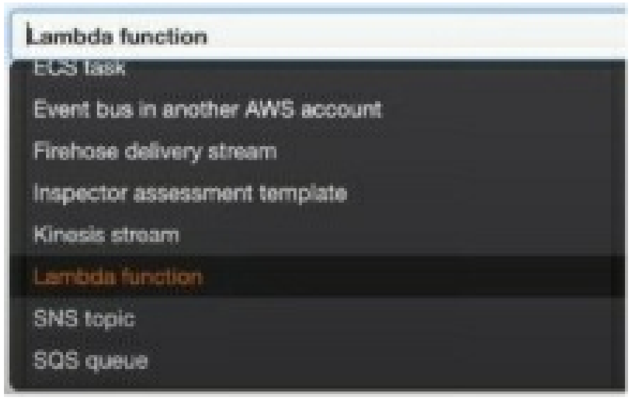
Build or customize an Event Pattern or set a Schedule to invoke Targets

○ Event Pattern ● Schedule

● Fixed rate of: 5 Minutes

○ Cron expression: 0/5***?*

Consider a use case where you schedule an event that you usually do in a Chrome tab. You might need to back up the server once a day. Therefore, you would trigger it, and then an EBS snapshot is created. So, this concept is used in CloudWatch Events. Moreover, depending on a pattern or a timeframe, one can trigger events. Moreover, there are several inputs here that are not necessary to discuss. However, EBS snapshot and Lambda are the most common among all.



Custom Metrics

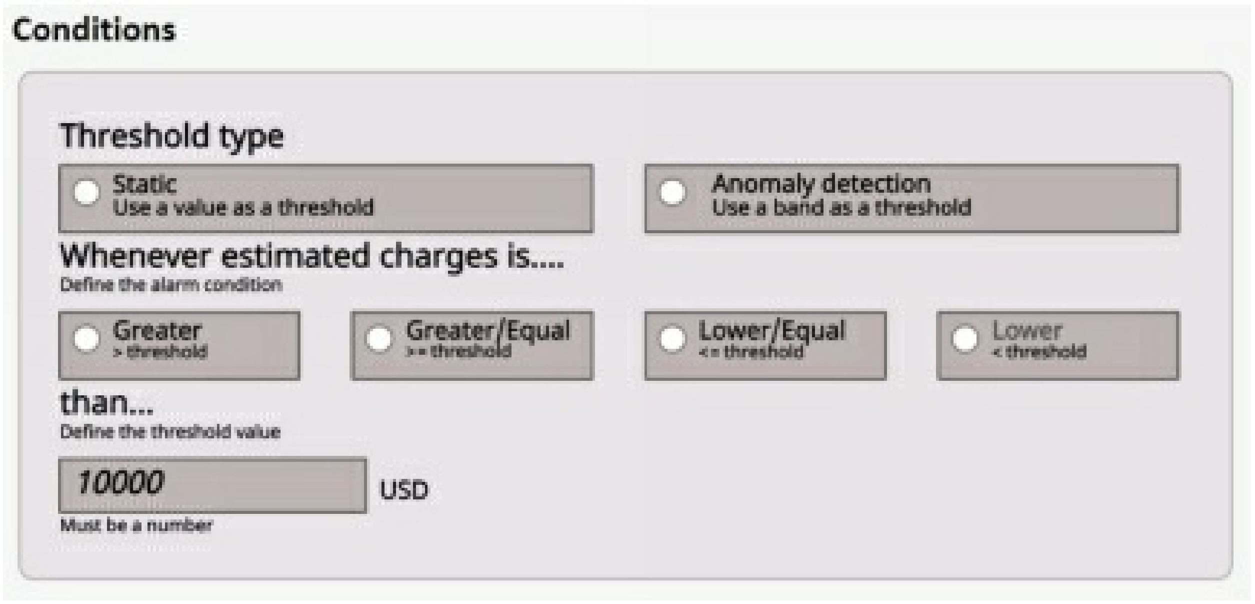
There are plenty of *Pre-defined CloudWatch Custom Metrics*. Let us say that you want to make your own custom metrics. Therefore, you need to use the Software-Development-Kit or a Command-Line-Interface to do so. For custom metrics, you can send data programmatically. Thus, you can send any data and publish it to CloudWatch metrics.

Another interesting characteristic of custom metrics is that it provides the opportunity to obtain high-resolution metrics. It is only achievable through custom metrics. So, when you want to edit the data with high-resolution metrics in less than one minute, you can go as down as one second. There are different intervals, such as one second, five seconds, ten seconds, and thirty seconds. However, if you enable it, you possibly want to go as low as possible because one must keep in mind that the higher the frequency, the more will be the cost. Thus, the only way to obtain high-resolution metrics is through custom metrics.

CloudWatch Alarms

CloudWatch Alarms send a notification to the users based on metrics when the defined threshold is exceeded. The most common use case is Billy alarms. It is one of the very first steps while setting up an AWS account. There are different options while setting the alarm. In the conditions tab, you need to mention if it is static or anomaly detection.

Furthermore, you need to mention the time to trigger the metrics, such as when it is greater than, equal to, or less than the threshold and the amount of limit. Let us say that you are watching for a thousand dollars, and you mentioned that it is fine while it is under a thousand dollars. However, send me an email if it exceeds a thousand dollars. That is the utility of CloudWatch Alarms.


Conditions

Threshold type

Static — Use a value as a threshold

Anomaly detection — Use a band as a threshold

Whenever estimated charges is....

Define the alarm condition

Greater — > threshold

Greater/Equal — >= threshold

Lower/Equal — <= threshold

Lower — < threshold

than...

Define the threshold value

10000 USD

Must be a number

Availability

It is important to understand how often CloudWatch updates the available metrics to you as availability varies for different services. When you use an EC2 instance, CloudWatch monitors at an interval of five minutes by default. If you want to get down to one minute, you have to turn on detailed monitoring that costs money. It will be between a minute and five minutes for all other services. There may be few other services with detailed monitoring, but EC2 is not one of them by default. The majority of services are by default one minute, whereas the EC2 is five minutes. You can get one minute by turning on detailed monitoring for EC2.

Agent & Host Level Metrics

Normally, CloudWatch does not track everything for an EC2 instance. So, it does not track memory utilization and will not inform about space left on the server by default because these are *Host level metrics*. These are detailed metrics. So, you will have to install the *CloudWatch Agent* to collect this information. The CloudWatch agent is a script that you can install through the systems manager's *Run command*. It probably comes pre-installed on *Amazon Linux 1* and *Amazon Linux 2*. Thus, Installation is necessary for detailed metrics such as memory and disk space. The already available detailed metrics are disk usage, network usage, and CPU usage. However, the disk usage here is limited.

CloudWatch Use Cases

Infrastructure Monitoring and Troubleshooting

CloudWatch monitors the key metrics and logs, generates alarms, checks applications, and compares the logs and metrics to sort out the operational challenges in AWS resources. It includes monitoring the user container environment through Kubernetes, AWS Fargate, Amazon EKS, and ECS.

Mean-Time-To-Resolution Development

The CloudWatch helps in comparing, evaluating, and visualizing logs and metrics, allowing you with instant response time to figure out the problems and incorporate them with the trace information from AWS X-Ray for end-to-end observation. The user's request can also be determined to accelerate the problem solving (debugging) and decrease the total mean-time-to-resolution or MTTR.

Proactive Resource Optimization

When you determine the metric values, CloudWatch alarms will evaluate those metric values against the threshold, and CloudWatch monitors the abnormal behavior of resources by using machine learning designs. When generated, an alarm can automatically select an action to perform EC2 Auto Scaling or stop an instance for presentation. It means that the resource planning and compute capacity are automatically available to you.

Application Monitoring

The CloudWatch will monitor the applications that work on-premises or on AWS architecture. The data is collected from all the layers of the performance stack, including metrics and logs on automated dashboards.

Log Analytics

The CloudWatch will analyze, visualize, and investigate the logs to improve the application performance by reporting operational issues. Inquiries can also be carried out to help you in an efficient and immediate response to the functional problems. Whenever you come across a problem, you can initiate querying by using the purpose-built query language for instant diagnosis of the root cause.

CLOUDTRAIL

Introduction

Cloud trail is used for monitoring API calls between AWS services and actions made on the AWS account. It helps in identifying the individual responsible for actions. AWS account holders can use CloudTrail to record and write each API call made to AWS resources in their account. The API call is made when:

- You make a REST API call to an AWS resource.
- You run an AWS CLI command.
- You access a resource from the AWS console.

Such activities may come from another AWS service, an application, or human users such as:

- When the Lambda function writes to an S3 bucket (Another AWS service)
- When an AWS CLI command is called by a bash script (An application)
- When you spin an EC2 instance from the console (Human users)

CloudTrail preserves the API events in an immutable and secure manner which you can use to analyze later.

Let us consider a record to understand the working of CloudTrail in simple words. This record can identify the person responsible when something goes wrong by providing five W's, i.e., *what, where, when, why,* and *who.* The 'where' part provides the account ID's details, such as the account where it happened and the person's IP address who created the request. The 'when' part tells the time at which it happened. The ‘who’ part tells about the user agent, which is the operating system, the language, the process of making the API call, or the user itself. Finally, the ‘what’ part provides information about the service and the region from where the request is generated. So, CloudTrail works in this way. This level of visibility can be helpful for proactive account vulnerability management, post-breach investigations, and ensuring compliance standards are met.

From an examination point of view, it is important to remember that when keywords such as governance, compliance, operational audit, or risk audit are mentioned in the exam, it is probably referring to AWS CloudTrail.

Features of CloudTrail

This governance and monitoring tool has various features such as

- Multi-region configuration
- CloudTrail Insights, Data events, and management events.
- Validation and encryption of Logfile integrity
- Event History to allow you to observe changes.
- CloudTrail is ‘Always On,’ allowing the data visibility for the last 90 days.

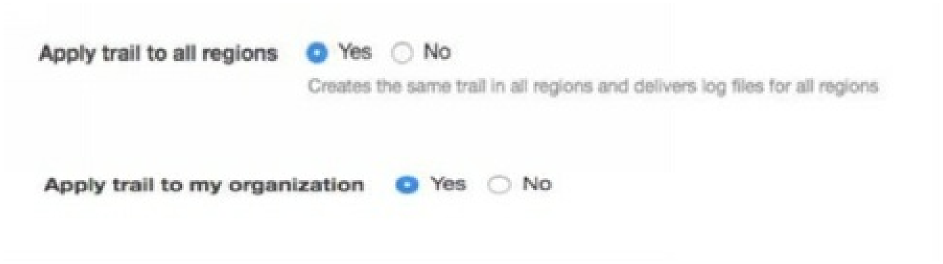
Let us discuss some features in detail.

Event History

CloudTrail monitoring logs already exist in the AWS account, which collects records for the last 90 days under the *Event History*. There is an interface here from which the events can be filtered out. Now, if you need logging beyond 90 days, which is a very common use case, you have to create a custom trail. The only drawback is that it does not have a GUI like the event history, due to which some manual work is required to visualize the information. A very common method is to use Amazon Athena. Thus, CloudTrail enables you to search the CloudTrail logs with the help of Amazon Athena.

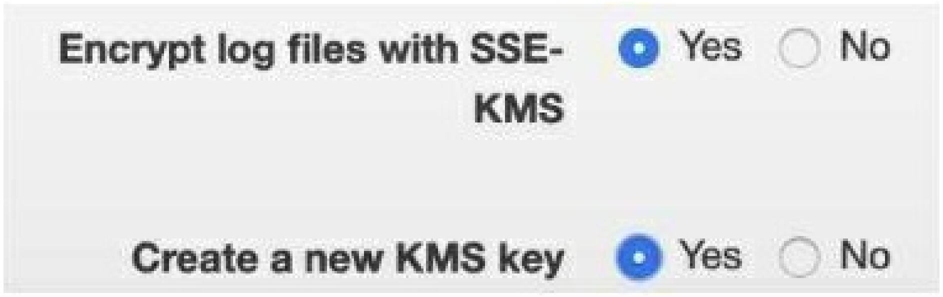
CloudTrail Options

There are plenty of *CloudTrail Options* which are important to understand. The first option is that a trail can be set to log in to all the regions. So, you can select yes to not miss any region. If you own an organization, you must have multiple AWS accounts, and you need the coverage over all those accounts then, you can check the box of *‘apply trail to my organization*.' You can also *‘Encrypt the CloudTrail Logs*,' required for KMS server-side encryption. The *‘Log File Validation’* also needs to be enabled because CloudTrail will inform you if someone has accessed your log files. It does not prevent someone from messing with your log, but at least it will inform you about relying on the log.



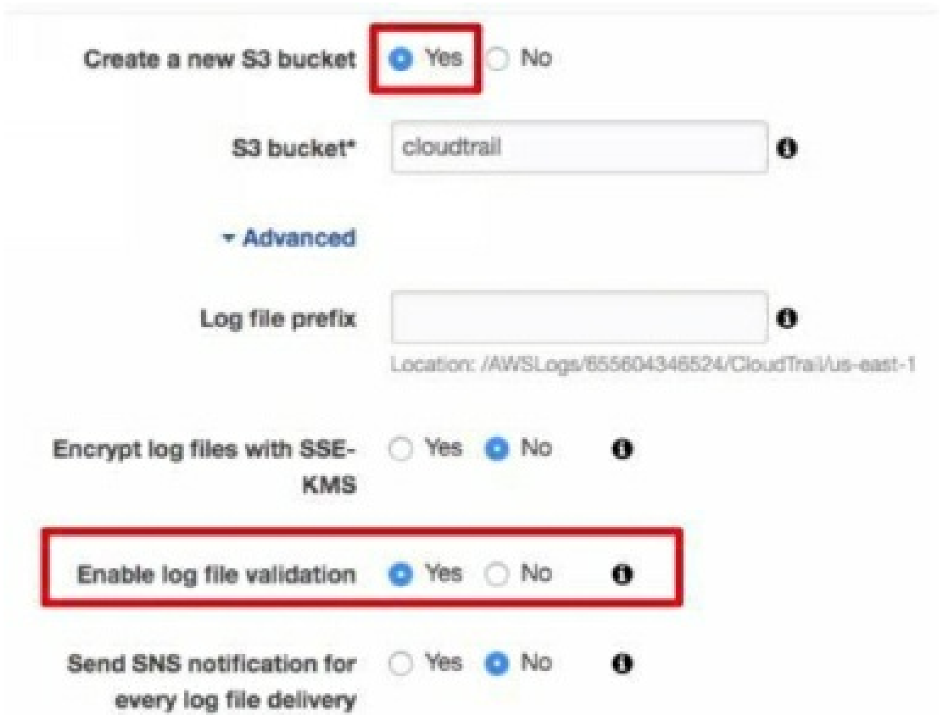
Apply trail to all regions [✓ Yes] No
Creates the same trail in all regions and delivers log files for all regions

Apply trail to my organization [✓ Yes] No



Encrypt log files with SSE-KMS [✓ Yes] No

Create a new KMS key [✓ Yes] No



Create a new S3 bucket [✓ Yes] No

S3 bucket* cloudtrail

Advanced

Log file prefix

Location: /AWSLogs/655604346524/CloudTrail/us-east-1

Encrypt log files with SSE-KMS Yes [✓ No]

Enable log file validation [✓ Yes] No

Send SNS notification for every log file delivery Yes [✓ No]

CloudTrail Events

There are various types of events in the cloud trail, including *Management Events* and *Data Events*. It is impossible to list all events as there are too many of them, but we can discuss the general idea for these events. So, there are four categories of events: *configuring security*, *configuring rules for data routing*, *registering devices*, and *setting up logging*. In the cloud trail, 90% of events are related to management, and management events are enabled by default. The data events are currently available for two services only: one for S3 and another for Lambda. These two are turned off by default as they possess high volume and occur regularly. The data event tracks S3 events such as *GetObject*, *PutObject*, and *DeleteObject* in further detail, whereas, for Lambda, it tracks Lambda every time it is invoked, which is why they are disabled by default. Thus, all other API calls are categorized as management events.

CloudTrail to CloudWatch

CloudTrail can also deliver its events to cloud watch. It helps in sending its insights of data, events, and data management to CloudWatch Logs. When a CloudTrail is being created, you can set up and forward events to CloudWatch logs, but only those events are sent that fit your trail setting.

Use Cases

Financial Issue Troubleshooting

Operational challenges can be solved by using the call history of AWS API developed by CloudTrail. You can observe the recent changes to the resources, such as adding, removing, or changing AWS resources in your environment.

Compliance Aid

In order to verify that your environment is compliant at any specified time, CloudTrail checks the history of every activity. Its event log files can be used by the IT auditors as a compliance aid. AWS resource modification log, log integrity verification and log analysis for unauthorized access are all part of a compliance audit activity's workflow.

Data Exfiltration

Data exfiltration can be detected using CloudTrail recorded object-level API events to collect activity data on S3 objects. When activity data has been collected, you can use additional resources (like Lambda or CloudWatch events) to provoke the response.

Abnormal Activity Detection

If you enable the CloudTrail insights, you can easily detect the abnormal activities in your AWS account. You immediately become alert and act upon the functional issues such as any service reaching its limit or faulty spikes in provisioning resources.

Analyzing Security

The integration of CloudTrail events with your log analytics and management solutions will identify client behavior patterns and run security analysis.

CLOUDFORMATION

Introduction

Let us discuss the concept of *CloudFormation* in detail. *CloudFormation* is a service that offers a convenient way for users to build and manage AWS resources in a very organized and predictable manner by creating and updating them. It is a service that lets you set up the AWS resources to concentrate more on the running application instead of wasting time on managing resources. You have to form a template for any EC2 or RDS instance and then upload it to the CloudFormation portal. Then, CloudFormation will build and manage the resources using a user-defined template. There is no need to worry about figuring out the dependencies and complexities between the applications. When a template is checked and verified, cloud formation manages all the applications' dependencies in an organized and consistent manner. When the template properly runs, it creates and makes those resources available for use. In short, it designs and creates the infrastructure and applications without performing manual operations. Take an example of a company, 'Expedia' that entirely manages and executes its front-end and back-end resources very easily running on AWS CloudFront. They spend less time on infrastructure management and more time on core business and applications.

How does it work?

With Cloud Formation, you can manage your complete infrastructure or AWS resources in a text file. This file is known as the *Template*, which is either in a JSON or a YAML format. The collection of resources that a template provides is known as a *Stack*. The stack will run the resources, which can be updated as well. Stacks are not limited, which means that the stack can be updated without creating another resource. Consider a use case in which you want to add two more servers to connect to other applications. As you are expanding the environment by introducing a new feature in your system, you can easily merge it into the stack and update it, clarifying that the stacks are updatable.

Now, let us discuss the functions and features of a template. With templates, you can add almost all applications and the resources that you might need. One can even make a template by provisioning EC2 instance and applications running on top of it. Moreover, these templates are portable, which means users can use them in multiple regions. If you use the same template in two regions, it can build a similar environment in both regions. It can also be shared with customers, and when the customer executes it, a portable template environment will be executed in his system from his AWS account with security guaranteed. You can build a resource, architecture, or an environment similar to other resources and the environment with the help of templates in CloudFormation. These templates can be reused by using some sections of the template. By reusing the parameter, mapping, and condition sections within the template, you can make a template reusable.

Now, let us understand how a template becomes infrastructure. Let us say you need a JSON format template that will be created based on various factors. These factors include user requirements, number of EC2 instances, time at which users want a load balancer, window server, a database server, and the applications running on top of it. All of this can be templatized into the code by you. Therefore, one creates a code and either put it in an S3 bucket or saves it locally from where the CloudFormation will call the template and provide the resources. Therefore, CloudFormation is used to create the stack or the resources defined in the template. The CloudFormation will analyze the templates first, validate them, and detect the dependencies between them. After that, it starts providing the resources gradually, one after the other. So, if you need five servers in a VPC, it will create the VPC first, then servers, and then the stack.

The infrastructure provisioning and updates are automated by CloudFormation in a managed and secure way. The *Rollback Triggers* can be used to configure CloudWatch alarms for CloudFormation to track while the stack creation and update are in process. The entire stack will be rolled back to the earlier state by the CloudFormation if any alarm is breached.

ChangeSet

With CloudFormation changesets, you will be able to see the effects of upcoming stack changes on your resources before they are implemented. If updating a stack is required, then simply running an update on the stack can solve the problem, and AWS will later replace the necessary resources. With the help of changesets, you are able to see the effect of these changes before they are applied.

Components of Cloud Formation

The two Key components in cloud formation are the *Template* and the *Stack*. Together, they complete the cloud formation. With the help of these components, the resources are easily provisioned and managed by the users. This resource management simplifies the job of an administrator who is managing the environment. Let us look at the template first.

Template

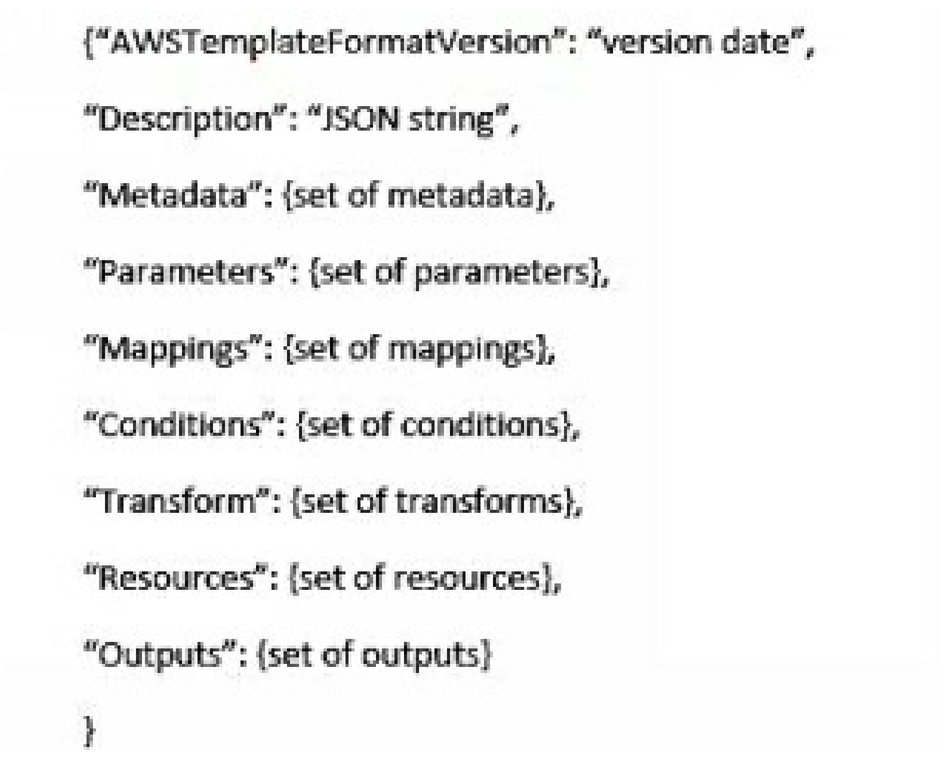
A *Template* is a formatted text file in *JSON* or *YAML* language. The core of the template describes the model of infrastructure when provided. To create a new template or modify it, there is a *Cloud Formation Designer tool* in the console. Any available text editing tool can be used here. There are nine main objects to build these templates. The first object is the *Format version*. It identifies the template's capabilities, which are determined by the version number. The *Description* is the second object in the template. It helps make random remarks about the template, such as the template's purpose or reasons to build a template.

Metadata is the third object in a template that includes the template's details and resources in the template. Then the fourth one is *the Parameters section*. It is an optional object which is used to customize a template. Whenever a stack is created or updated, it allows you to enter custom values into the template. *Mapping* is the fifth object used for key mapping onto the relevant named values’ set. Consider a use case where you want to set a value depending on a specific region. For that, you have to create a mapping that uses the name of the region as the key and then specify the value for each region. Then we have another optional object called *Conditions*. It allows you to include the statements to define a resource creation or property definition.

Consider a use case where you want to compare two values to check whether they are equal or not. Depending on the result of that comparison, you can conditionally create additional resources. The seventh object is known as *Transform,* which is also optional. This section explains one or more transforms that you can use in the cloud formation template. AWS Cloud Formation transform helps simplify the template's creation by compressing the expression of the AWS *Infrastructure as a Code* of the template. Moreover, it allows the reuse of template competence. It allows you to compress multiple lines of resources and declare them in a single line in the template.

Then the next object is known as *Resources***.** It is an important section that declares the AWS resource that you want to include in the stack. For example, when you want to introduce an EC2 instance and the S3 bucket, you use a resource object in the stack. The last object is known as *output*, which is also an optional object. It helps to declare the importable outputs into other stacks, or you can show them on the cloud formation console. Consider a use case where you want the output or the S3 bucket's name created in response to the cloud formation template. Thus, you can have that output in the output section of the cloud formation console. Therefore, it is easy to identify the resources provided by the template. That is how a complete cloud formation template will look like if you put all these objects together, including both optional and mandatory objects.

Sample JSON Template



```
{"AWSTemplateFormatVersion": "version date",
"Description": "JSON string",
"Metadata": {set of metadata},
"Parameters": {set of parameters},
"Mappings": {set of mappings},
"Conditions": {set of conditions},
"Transform": {set of transforms},
"Resources": {set of resources},
"Outputs": {set of outputs}
}
```

Resource Attributes

Let us now look at the *Template Resource Attributes* in clod formation. In this particular section, you can add attributes to a resource. It helps users to control additional behavior and relations between the templates. Thus, there are five types of resource attributes which we are discussing below:

Creation Policy

The *Create Policy Attribute* can delay the resource configuration actions before the stack creation proceeds further. It means that when you associate this attribute with a resource, the attribute stack creation is postponed until the success signals arrive back to the cloud formation. It is a good use case for auto-scaling. Moreover, the Creation Policy is compatible with a limited number of services, including auto-scaling and EC2 instance.

Consider a use case where you want to install a software application on an EC2 instance and want it to run before it proceeds further. In this case, you can attach the creation policy attribute to the instance. After installing and configuring the application, you will send the success signals that allow the cloud formation to move to the next stage.

Deletion Policy

Deletion Policy preserves the backup of resources when the stack gets deleted. Your data will be deleted if you have no deletion policy by default. However, if you use a deletion policy, you can preserve the data by taking snapshots, or you can select specific resources in the stack and delete them. Moreover, you have to specify the deletion policy attribute for the specific resource that you want to control. The cloud formation will delete those resources for which the deletion policy is not mentioned because it is the default behavior. Therefore, if you want to keep a resource, you need to mention *Retain* in the deletion policy because it keeps the resource from being deleted.

Depends On

Depends On attribute allows the users to create a sequence for deleting the resources. With the Depends On attribute, you can specify that the specific resource creation follows another resource. When you associate *Depends On* to a resource, that resource will create after the other resource, determined by the *Depends On.* It means that when you map one resource to another and specify that the X resource *Depends On* the Y resource, then the Y resource gets created before the X resource. Let us consider a use case in which EC2 Depends On S3. Therefore, when the cloud formation starts its executions, it will create the S3 bucket first and then the EC2 instance. That is because the codes for the EC2 instance might be stored in the S3 bucket.

Metadata Attribute

The next template resource attribute is known as *Metadata*. It helps in associating your resources with structured data. You can describe the data in JSON or YAML language by including this attribute in your template or resource.

Update Policy

Update Policy Attributes in the cloud formation template allow a user to manage and replace the instance's updates in an auto-scaling group. During an update, *WillReplace* determines the auto-scaling group and its instances being replaced.

Cloud Formation Access Control

With IAM, one can specify the list of users who can have access to the cloud formation template and create, update, or delete the stacks. Therefore, IAM helps you by providing a layer of control over the cloud formation service. You can use *Service Roles* that enable AWS cloud formation to call a resource in a stack on the user's behalf. You can also have Stack Policies, which are applied to the users having access to the cloud formation service already. These policies help them to upgrade the stack.

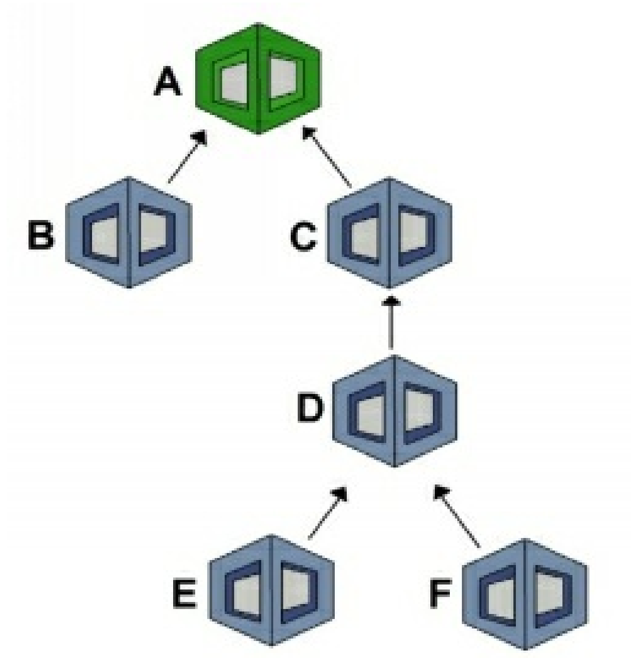
Stack

After the templates, let us discuss *AWS* CloudFormation *Stacks*. Now, the Stacks are the real resources created by running a template. It is a group of AWS resources that is run as a single entity. A stack can be created, updated, or deleted, which leads to the creation, deletion, and updating of the collection of resources in the stack. The cloud formation template defines all the resources in the stack.

Furthermore, a stack holds all those necessary resources for running web applications such as a database or web server and a few networking rules embedded into a stack. Let us assume that you want to eliminate all the resources generated in a separate stack by the template. You can delete the stack that will, in turn, delete all the resources in that stack. Thus, it helps you to manage all the resources as a single unit. The cloud formation template defines a stack capable of creating, updating, and deleting resources in a very predictable way. It includes all the resources necessary to operate a web application or database applications, including a database server, web server, VPC, the networking rules, and the security configuration in a VPC.

Nested Stack

In CloudFormation, one can *Nest* a stack with another stack leading to create the hierarchy of stacks. Let us take a use case that perfectly explains the nesting of stacks in a hierarchal manner. Assume that stack A is a root stack that could be a parent stack for B and C stacks. Similarly, stack C can be a parent stack for another stack D that can be a parent stack for other stacks called E and F stacks. Thus, nested stacks are possible in cloud formation.



Windows Stack

CloudFormation allows you to create a *Microsoft Windows Stack* based on Amazon EC2 Windows AMI. In this way, you can use a remote desktop to access the stack and install software and update its configurations. These stacks can run on windows server instances. There are various pre-configured templates available on the AWS website to use. Few available templates include a template for SharePoint, Elastic Beanstalk templates, and some other sample applications running on Windows-Server-2008-R2. These are responsible for installing a single server of an active directory running on a windows server.

Stack Sets

Stack Sets are the updated form of stack-level implementation. You can define a stack set by using the AWS cloud formation template. It allows you to create a stack in AWS account across the globe by using a single template. When a stack is created, you can specify the target account and the regions to create, update, and delete it. By doing so, the stack's functionality increases, and you can perform such functions in various regions and accounts within a single operation. You can manage and control the AWS cloud formation template using the administrator account. You can also use it to provide stacks in a few targeted accounts across a specific region.

Use Cases

CloudFormation is a well-suited mechanism for the provisioning of a wide range of AWS resources. It can support the infrastructure requirements of several application types, which include legacy applications, applications designed with AWS resources and container-based technologies, and current business applications. Different enterprises and companies use the feature of CloudFormation, known as 'single-click deploy' for their websites worldwide. It is used to eliminate the need for creating and launching any applications repeatedly because multiple backups for the same application are already created to be hosted on different locations.

Let us go through the use case scenarios of AWS CloudFormation. The most common use case scenario involves simple and small applications that are launched using EBS that require only a few resource modifications. For such use cases, CloudFormation templates must be used, as they will allow you to monitor all of the changes made to the infrastructure in comparison to CloudFormation templates. Assume another use case in which there are several web applications, and they require separate configurations and resources. Then you will need an environment where you can deploy these applications, each with their own set of resources. In that environment, you will also perform Blue-Green deployments. So in such a use case, the most suitable approach is to write the unique CloudFormation templates in order to create EBS applications for all of your applications.

Moreover, you can list the associated resources and their configurations in the same template. To upgrade your EBS application, you can write a new template with a new EBS application and resources after the templates have been implemented and the environment is ready. You can apply the proxy changes when the updated version application is examined so that you may route the traffic from the old application to the latest one.

CHAPTER 6: NETWORK & CONTENT DELIVERY SERVICES

Overview

Content distribution networks (CDNs) have started to emerge rapidly, utilizing cloud resources such as compute and storage. Cloud-based CDNs use geographical availability and the PAYG billing model, which is different from our conventional CDNs hosted in private data centers. The CCDNs support the AWS cloud model of content distribution as a service.

In this chapter, the AWS networking components will be discussed to design any environment using VPCs and content delivery solutions via AWS Global network, especially Edge Locations. Moreover, this chapter gives a general idea of the AWS VPC and the networking components. AWS networking services offer a vast spectrum of database and networking options that are scalable, on-demand, and available with a few mouse clicks. Let us look quickly at the *AWS Global infrastructure* before we get started with this segment.

AWS Global infrastructure

The cloud infrastructure of AWS is designed around the *Availability-Zones* and *AWS Regions*. In the *AWS Global infrastructure* environment, a physical location in the world in which there are several availability zones is known as Region. Each AWS Region comprises multiple AZ's that are physically separated. They ensure the high availability of an application. There are two or more AZ's available within a region. Normally, users keep their machines in separate AZ's while designing the AWS architecture. If one of the AZ goes down, their machine will be running in another AZ, and the application will have high availability.

On the other hand, an Availability Zone (AZ) has multiple distinct data centers kept in separate locations with redundant networking, connectivity, and power facilities. These Availability Zones allow you to run highly accessible, fault-tolerant, and scalable databases and production applications from a single data center. The AWS Cloud works over 25 geographic regions and 80 Availability-Zones across the world, and it has declared plans for more Regions and Availability Zones.

Now, let us start with the first and most basic networking service known as Amazon VPC.

AMAZON VPC

Introduction

Amazon VPC stands for *Virtual Private Cloud*, which enables its users to build a virtual networking system inside the cloud in their own logically isolated area. The EC2 instances can be launched into the VPC subnets. A VPC is like a standard network system that you can have in your data center. Every Single VPC is entirely customizable and logically separated from the rest of the cloud's virtual networks. You can select a number of IP addresses, set up network gateways, configure route tables, and create subnets. As a prerequisite, you must create an IPv4 CIDR Block. In addition, the VPC's IPv6 CIDR Block address is also available.

Moreover, you can configure security settings with the help of security groups and NACLs. VPCs are region-oriented, so they do not span regions. One can create five VPCs per region where each region has a default VPC. You can have 200 Subnets per VPC, which is a large number. DNS hostnames are not enabled by default while creating a VPC. When an EC2 instance runs with DNS hostnames enabled, it will give you a public IP. However, it will only get a public DNS that is a domain name, such as an address.

When you deploy an instance in the AWS account, it will have a private and public IP and will be deployed into the default VPC by default. The IP addresses that are not accessible over the internet are known as private IPs, which allow inter-instance communication in the same network. On the other hand, public IP addresses are used for intra-instance-communication on the internet. You can use these IPs to communicate between your and others instances on the web. Then there are Elastic IP addresses which are persistent/static public IPs that come with your account. If an instance or software fails, you can immediately remap it to another instance by using an elastic IP. Furthermore, the default IP address contains all valid IP addresses and is used for giving access to user’s public resources from the internet. It is represented as '0.0.0.0/0'. When you interact with AWS networking, you can use this address for sending traffic to the internet to get an internet gateway route. Therefore, you need to set this IP to enable any internet traffic to access your public resources using *SG inbound rules*.

When you create a VPC, there is no cost involved for using Route Tables, Internet Gateway, Security Groups, Subnets, and VPC Peering. However, there are resources in VPC that cost money. They are VPC Endpoints, NAT Gateway, Customer Gateways, and VPN Gateways. You mostly use those resources that do not cost anything. Therefore, you should not worry about getting over-billed.

Launch instance into a VPC

While launching an instance and specifying a VPC, the first thing is to define a subnet where you want to deploy the instance. If you do not mention a subnet, then AWS will choose the default subnet in the selected VPC. The next step after identifying the subnet is to identify an AZ, which can be selected by you, or AWS can pick one.

A primary private IP is allocated to the instance’s default network interface from the IPv4 subnet range whenever an instance is created within a VPC. In order to resolve an instance’s private IP address, the private DNS hostname must be provided to each instance. In case you do not specify a primary private IP, AWS selects it for you from the available subnet range.

Whenever you launch an instance in a subnet with an enabled public IP address attribute, the primary NIC generated for an instance will be assigned a public IP. Thus, the primary private IP is mapped to the primary public IP with the help of NAT.

Now let us look at the two types of VPCs.

Types of VPC

Default VPC and non-default (*Custom*) VPCs are the two types of VPCs. In Default VPC, Amazon creates it by default, and one can instantly launch an instance into it. The Default VPC is available for every region. You can instantly launch the EC2 instances without considering any network setup. A default VPC comes along with specific configurations such as:

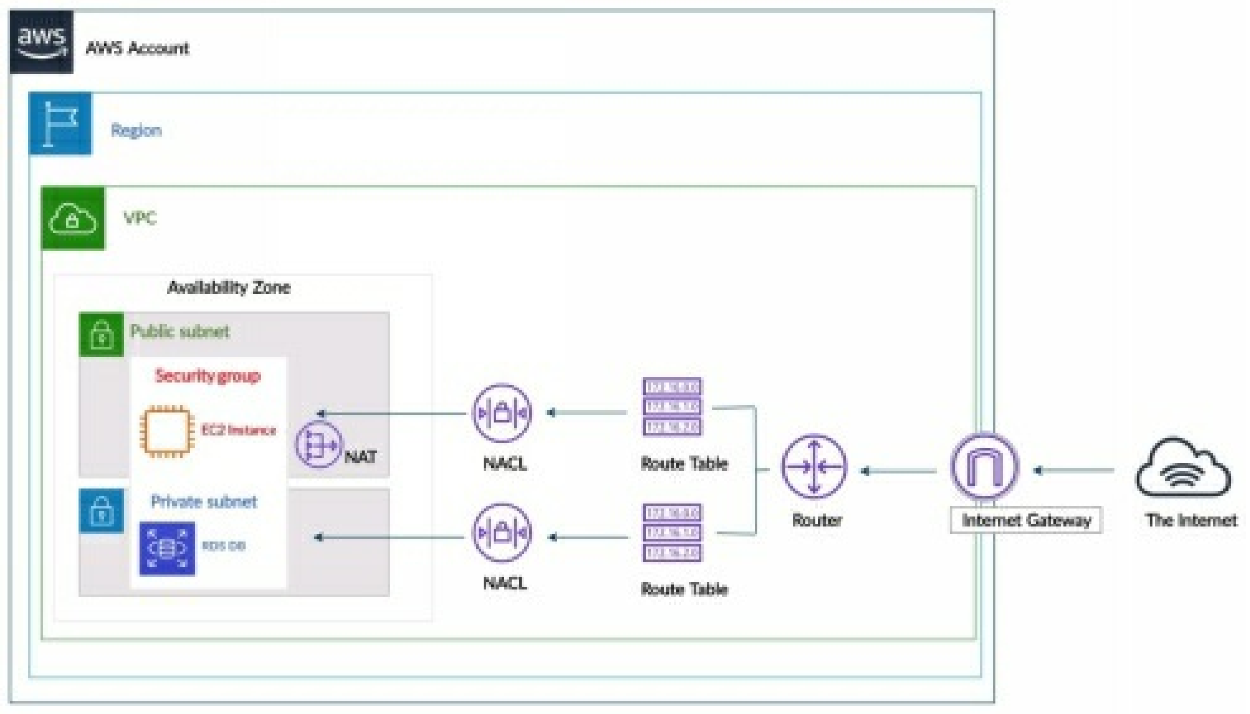
- It creates a VPC of CIDR Block size /16 with attached default subnet (size /20) for each Availability Zone.
- The main Route Table is provided with a rule that transfers all IPv4 traffic to the IGW.
- An IGW is created and linked with the default VPC so that the EC2 instances have internet connectivity.
- Default SG is associated with the VPC. Running an instance will automatically enable the SG unless you override it.
- It also integrates with default NACL and DHCP options.

In contrast, custom VPCs can be created using out-of-the-box configuration for hosting the production workloads in default VPCs. AWS suggests that you need to customize your VPC when its environment becomes complicated. For instance, a default VPC implies that every EC2 instance will reside in internet accessible public subnet, which may not satisfy a company's security requirements. Thus, you need to generate a Custom VPC and configure it as you want. Creating a custom VPC enables its users to:

- Making resources secure
- Modify the virtual network to identify the user's IP address range.
- Create private and public subnets
- Reinforce security settings.

Core Components

VPC provides logically separated sections of the AWS cloud to its users, where the user initiates AWS resources in the virtual network. The easiest way to understand VPC is by considering it as a personal data center because it offers complete control over your virtual networking environment. Let us look at the architectural diagram of VPC with multiple networking components in it.



We have a Region, and it has VPC in it. Then VPC has multiple Availability Zones in it. So when the internet flows into the network, it enters from the *Internet Gateway* and passes through a *Router*. The Router sends the traffic to a *Route table*, which will pass it to the *NACL*. After that, NACL directs the traffic to the *Private* and *Public Subnets* in the user's VPC. Then there is a *Security Group* that contains all the resources within VPC. This is just an overview of how network traffic flows into the VPC from the internet. Now let us look at different networking components which are part of the Amazon VPC.

Subnet

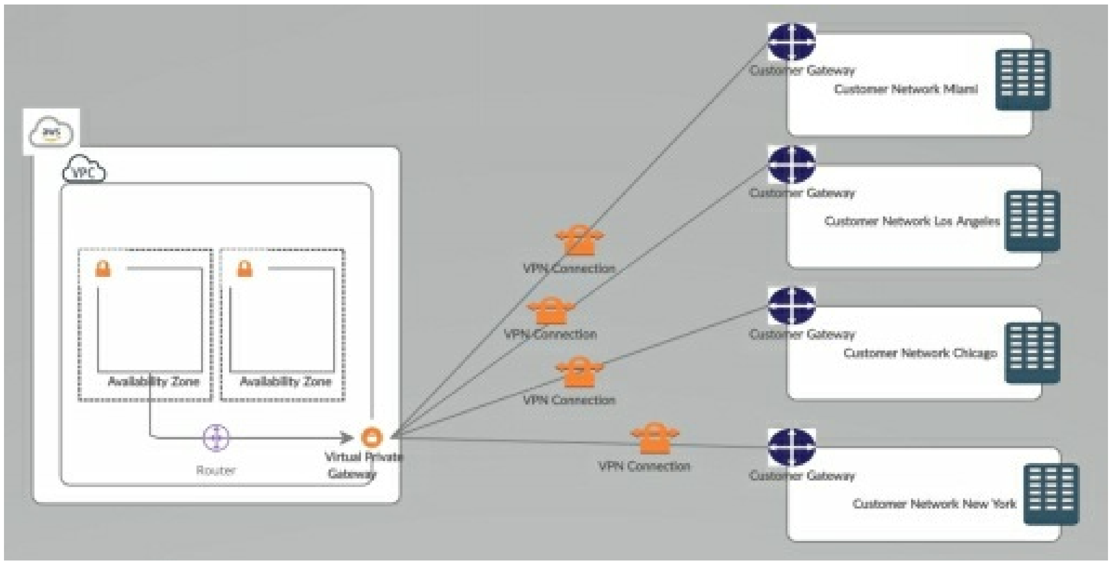
It is basically a set of IP addresses in Virtual Private Cloud within the AWS environment. A VPC consists of multiple availability zones, but the *subnet* lies in each AZ. Therefore, an instance cannot be launched until the subnets are present in the VPC. However, you can launch AWS resources in a specific subnet. Furthermore, the resources connected to the internet use public subnets. In contrast, the resources not connected to the internet use private subnets. The *Netmask* for a default subnet is 20, with a maximum of 4,096 addresses for all subnets, of which AWS keeps only a few.

Private & Public Subnets

Let us look at the two types of VPC subnets. The resources that are linked with the internet use *Public Subnet*, such as web servers. The main Route Table delivers subnet traffic intended for the web to the internet gateway, so it is made public. On the other hand, the resources that do not need an internet connection use *Private Subnet*. One might use them to secure their resources from the internet, such as database instances.

Virtual Private Network

By default, instances launched in the Amazon VPC cannot communicate with the user’s network. Users can link VPCs to an existing data center using *VPN* access. By doing this, they can create a Hybrid environment that will expand the data center into the cloud. They have to set up a virtual private gateway for this purpose.

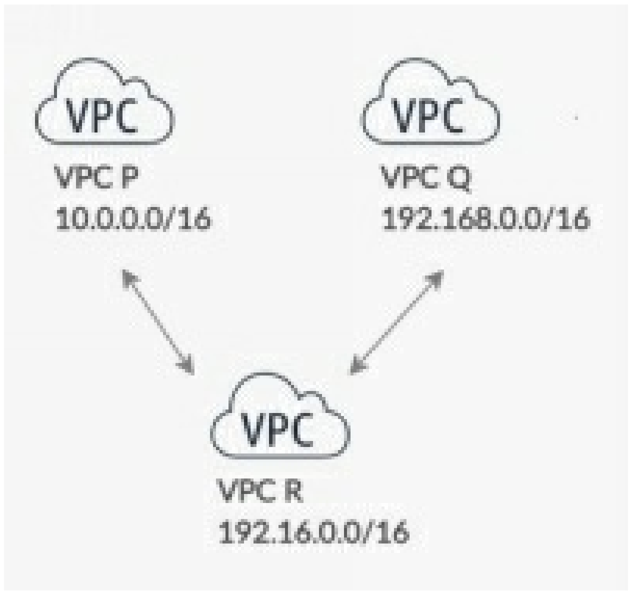


In this diagram, there is a VPN concentrator on the provider side of the VPN link. You will need the customer gateway for your data center. It is either a Software-application or a Physical-device that resides on the consumer side in the VPN link. When a VPN link is made, the traffic from the customer's side connection creates a VPN tunnel.

VPC Peering

VPC Peering enables its users to link one VPC to another across a direct network route using private IP addresses. A peering link can be established between two VPCs in your AWS account if they are in the same region. Let us say you have instances in VPC A, and you want it to communicate with

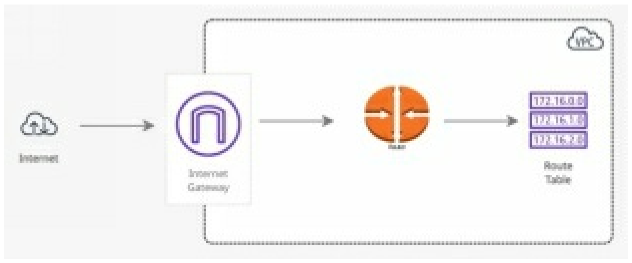
VPC B or C. In that case, you must create a peering link. Peering is a one-to-one relationship that means a VPC can be attached to other VPCs through multiple peering connections, but it does not allow transitive peering. In the following figure, VPC P can connect with VPC Q and R. In contrast, VPC Q cannot communicate with R until it gets paired directly. In addition, VPCs cannot pair if they have overlapping CIDRs. As seen in the figure, all VPCs have different IP ranges. They could not pair if they had the same IP ranges.



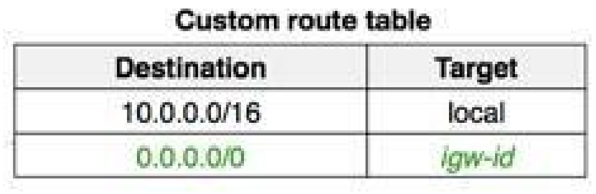
Now that we have learned how VPC operates and how instance launch works in it, let us expand this chapter by exploring further networking components.

Internet Gateway

This service allows internet access to the VPC and performs two activities. It creates a *Target* in the Route Table for internet traffic routing. It also performs Network Address Translation on instances with public IPv4 addresses. It allows instances in the VPCs of users to connect with the internet. This ensures that the network traffic does not face any bandwidth limitations or availability risks. To allow the VPC to connect to the internet, one must connect to an internet gateway. Moreover, only a single internet gateway is allowed to attach per VPC. So connecting an Internet gateway is the first step in allowing internet access to instances in your VPC.



Let us look at the IGW working. When you want an IGW to access the web and route the traffic from EC2 instances, you have to pass through a Route table to get to a router. Thus, you will be generating a new Route in the Route-table for the Internet Gateway. For that, you have to provide the information for the target as ***igw-id*** and destination as ***0.0 0.0/0*.**

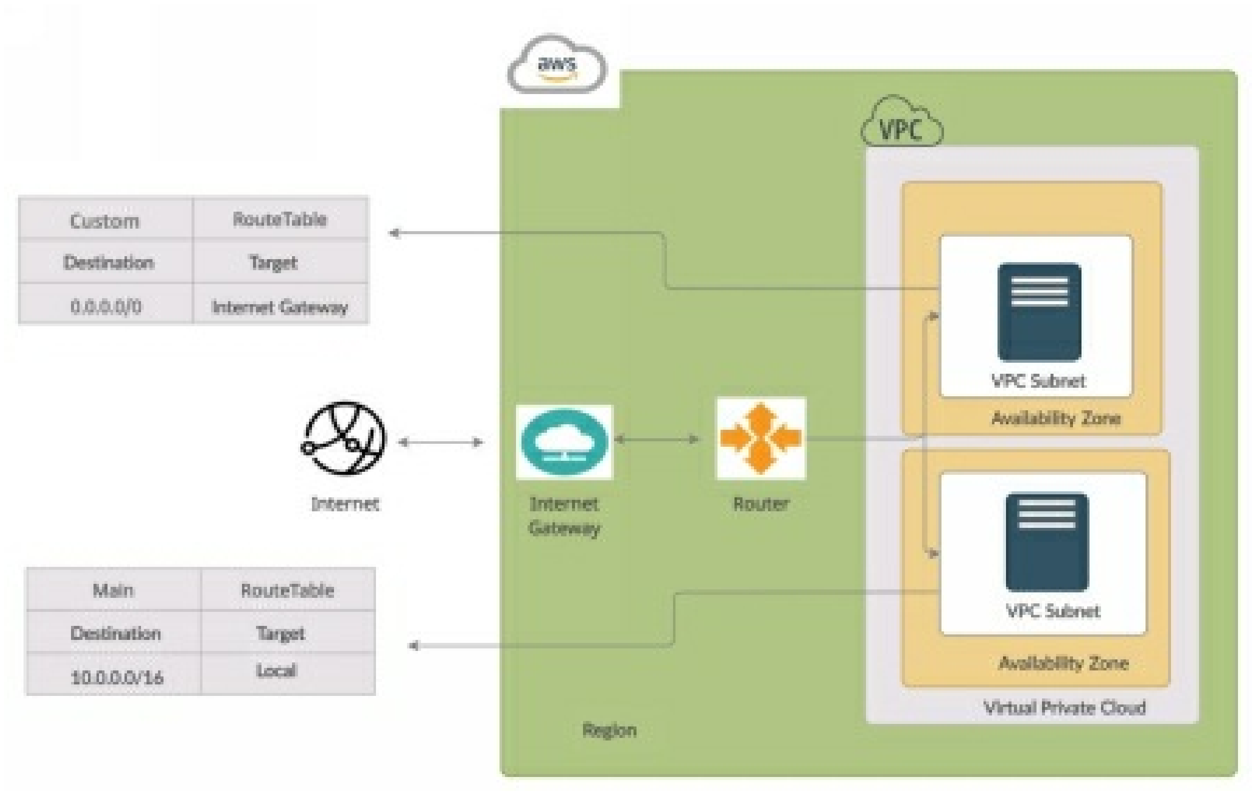


Custom route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*igw-id*

Route Tables

Route Tables are used to determine the path of network traffic. A route table should be connected to each subnet in your VPC. Only one route table can be used with a subnet at a time, but several subnets can be connected with the same route table. A common example of using the route table is that it permits EC2 instances to access the internet. In this way, you can have a public subnet where that EC2 instance belongs and links a route table to it. The route table will then provide the path that will give internet access to you through the Internet Gateway.



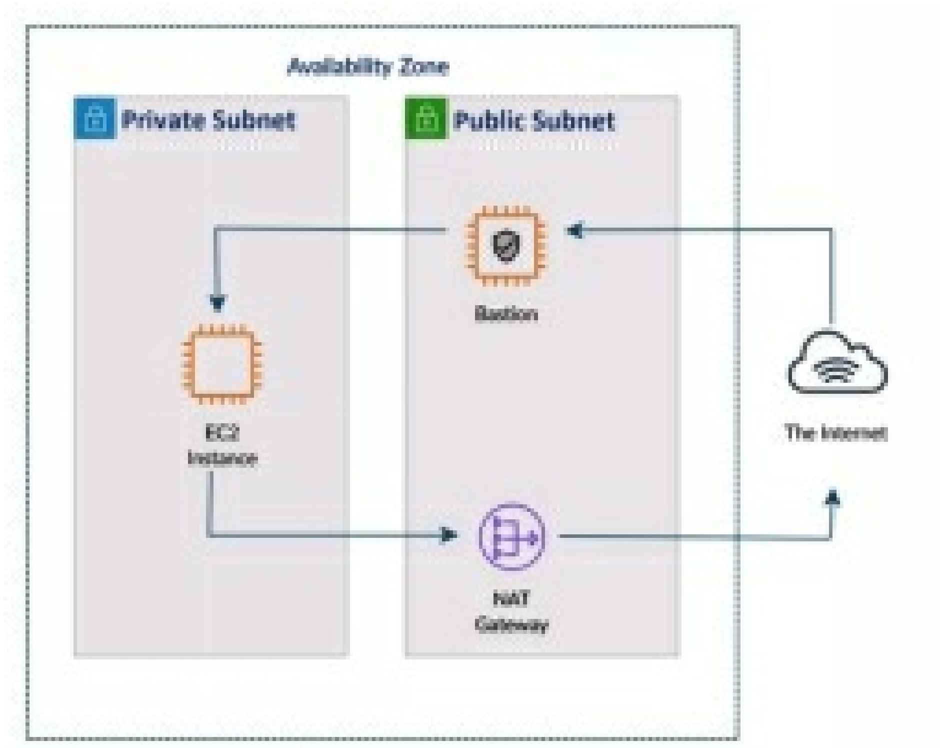
We added two route tables in this use case:

- Default route table
- Custom route table

A private subnet is attached to the default route table, which will not allow web traffic to come into the VPC. Therefore, all the traffic inside the private subnet remains local. On the other hand, Custom Route-table will notify the internet gateway to send the web traffic to the public subnet.

Bastion Host

It is a safe and robust instance that belongs to the public subnet of the VPC. It has classified security controls that allow limited inbound links from known and verified IP addresses. Generally, it uses RDP or SSH protocols for this purpose. These secure controls let the engineers establish a local and secure connection to the *Bastion Host*. Thus, the instance here serves as a '*jump server*,' which enables the engineers to RDP or SSH to the instances located in private subnets.



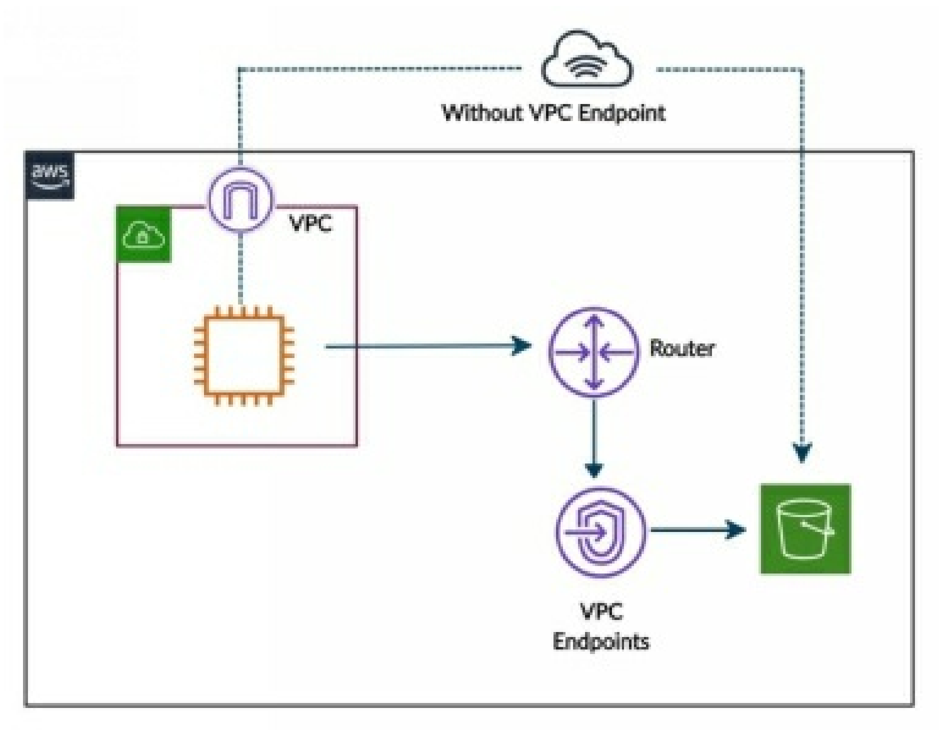
Although the NAT instance is an EC2 instance, one cannot turn it into a bastion like NAT Gateways. However, from a security point of view, you will not prefer to do so because of NATs' configuration. You would always prefer to have different EC2 instances as a bastion. Besides, a service called *Systems Manager's Sessions Manager* replaces the need for bastions where you need not launch your EC2 instance. Thus, AWS recommends it, but many companies are still using bastions because they meet their requirements.

Direct Connect

Direct Connect is an extremely fast AWS solution for establishing dedicated network connections from the user’s on-premise locations to AWS. The configurations are based on the selection of the range of bandwidth. The lower bandwidth is between 50 MB to 500MB, whereas the higher bandwidth is 1GB to 10GB. Thus, the transfer rate of the network from the on-premises environment to AWS is considerably fast. Direct Connect helps to reduce network cost and increased bandwidth throughput. It offers a better network experience for individuals and enterprises as compared to typical Internet-based connections.

VPC Endpoints

Users can use *VPC Endpoints* to privately join their VPC to other VPC endpoints and AWS services by keeping the traffic within the AWS network. The public IP-Addresses are not required to communicate with the AWS services because it eliminates the requirement of an Internet Gateway. Let us consider a use case in which you need some data from the S3 bucket. So, you request it using SSDK. Furthermore, to get the data file from S3, the request would go out from the internet gateway and come back into the AWS network.



We have two types of VPC Endpoints in the AWS Network. They are *Interface Endpoints* and *Gateway Endpoints.* Interface Endpoint provides *Elastic Network Interface* (*ENI*), a NIC which connects multiple AWS services and has a private IP address from the VPC subnet IP address set. It acts as the entrance for traffic that is going to a supported service. Furthermore, the AWS Private Link supports the Interface-Endpoints. Therefore, one can have smooth and secure access to AWS's services by keeping the network traffic in the AWS network. It supports many AWS services such as *API Gateway, CloudFormation, CloudWatch, Kinesis, AWS KMS, Secrets Manager, SNS, SQS, Service Catalog, SageMaker, Codebuild,* and *AWS Config.*

On the other hand, Gateway-Endpoint is the second VPC Endpoint type. In the Gateway Endpoints, the Route-table is used to access AWS services. The specific route in your route table used for the traffic flow of a supported AWS service is specifically targeted by the Gateway endpoint. In order to set up a Gateway Endpoint, you must identify a VPC, where you want to build an endpoint, and the service to which you want to build a link. Gateway endpoint is a 100% free service, and it is usually used for DynamoDB and Amazon S3. You will be charged for using the Interface-Endpoints. On the other hand, Gateway Endpoints are free.

VPC Flow Logs

VPC Flow Logs enable users to get incoming and outgoing IP traffic information from their network interfaces within VPC. Using VPC flow logs, you can record the information regarding the IP data flowing from and to the specified network interfaces and then store the raw data in CloudWatch, where you can view and retrieve it. VPC flow logging is important for compliance and security in the cloud environment of AWS.

VPC flow logs cannot be tagged like other resources. You can turn on Flow Logs at three different levels. They can be turned on at the network interface level, subnet level, and VPC level. When the VPC is disabled, all traffic types excluding multiple IP addresses, DNS, and DHCP traffic are monitored by the VPC flow logs. When a flow log is created, it cannot be edited but can only be deleted. Moreover, it stores the source as well as destination IP addresses. If a question arises in the exam that whether the flow logs contain the hostname or the IP address, the answer would be that it contains only IP addresses.

NACL

NACL is an additional security layer and is used as a subnet-level virtual firewall controlling the incoming and outgoing subnet traffic**.** It is enabled by default when creating a VPC. It holds internal and external rules, like in security groups. The difference here is that you can only allow the traffic in security groups, whereas it gets denied in NACL.

When you make the rules here, it is almost the same as security groups except for the rule number. This rule number determines the order of evaluating these rules. It evaluates from the smallest to the largest rule number, which can be as high as 32,766. AWS recommends that when a user comes up with rule numbers, he should use increments of 10 or 100 so that there is some flexibility to create the rules. You need to associate subnets with NACLs if you want to apply them because subnets can only belong to a single NACL. Security groups can include instances that belong to multiple cases, while for NACL's, it is a singular case.

Let us look at the two use cases for observing NACL's denying capability. Assume that a suspicious person is attempting to access the instance, and you are aware of the IP address. It can be added as a rule to your NACL to deny that IP address. Consider another use case where you know that you need not SSH into these instances. You only need an additional guarantee for a situation where someone misconfigured a security group to deny SSH access. Therefore, you will deny it on port 22. In this way, users can use NACL’s to deny unauthorized access to their resources.

AWS Transit Gateway

In addition to the remote networks, *AWS Transit Gateway* (abbreviated as *TGW*) enables the users to connect thousands of on-premise networks and VPCs together. The TGW serves as a central hub within the network, letting all VPCs, remote offices, and data centers connect to it. This configuration simplifies multiple peering connections and decreases general connectivity from local locations to the VPCs. It comes with high flexibility, allowing new cloud network architectures to emerge and replacing several point-to-point peering connections. It provides a solution for simplified network architectures, centrally managed external connectivity, and minimizes operational overhead.

PRIVATE LINK

Introduction

Before discussing *AWS PrivateLink*, its functionality, and use cases, let us first understand the relation between VPC Endpoints and Private Link. With the help of *VPC Endpoints*, one can connect his VPC to VPC endpoint services and AWS services privately. It utilizes AWS Private Link to wipe out the necessity for AWS Direct Connect, a VPN connection, a NAT unit, or an Internet gateway. The Amazon network traffic between VPCs and other services remains secure by not leaving the network.

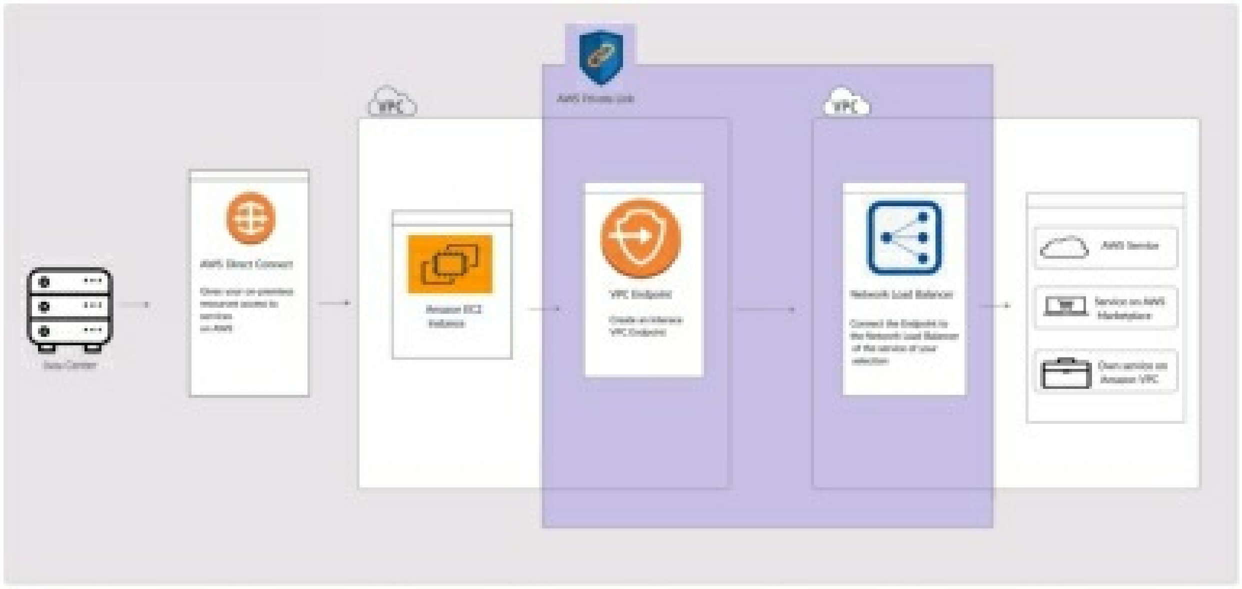
Now let us understand the concept of *AWS PrivateLink*. It offers a private connection among AWS VPCs, an on-premises application, and AWS services. It is highly scalable and available technology. It resembles Direct Connect, where it creates a private connection to the AWS cloud while Direct Connect connects the user's on-premises environments to AWS. However, Private Link secures the network traffic of the VPC environments of the users residing in AWS. Using the VPC endpoint services, it connects securely across the Amazon Network. The Interface Endpoints can, by default, allow 10 Gbps bandwidth in every AZ.

The traditional endpoints were like associating the virtual cables between the Virtual Private Cloud and Amazon Web Services. The connectivity for AWS service does not need any NAT gateway or an Internet; however, the endpoints stay outside of the VPC. While in the case of Private Link, the endpoints are made inside the user’s virtual private cloud utilizing IP addresses and ENIs. The service is now available in the user's VPC and allows him to communicate to AWS services through private IP addresses. It implies that the AWS Direct Connect can be used for accessing Private Link endpoints from your in-house data center. Moreover, you can use VPC SG’s to handle access to the Endpoints.

How does it work?

PrivateLink allows its users to connect VPCs to the AWS-supported services securely. It can link the VPCs to their own services on AWS, third-party services, and the services hosted by the other AWS accounts. There is a grid diagram below in which the in-house data center uses Direct Connect or a VPN to connect with AWS. In the checkbox below, AWS Private Link is created where a VPC generates an Interface endpoint which creates an ENI according to each AZ. The provider VPC is running a Network Load Balancer to keep the AWS services behind.

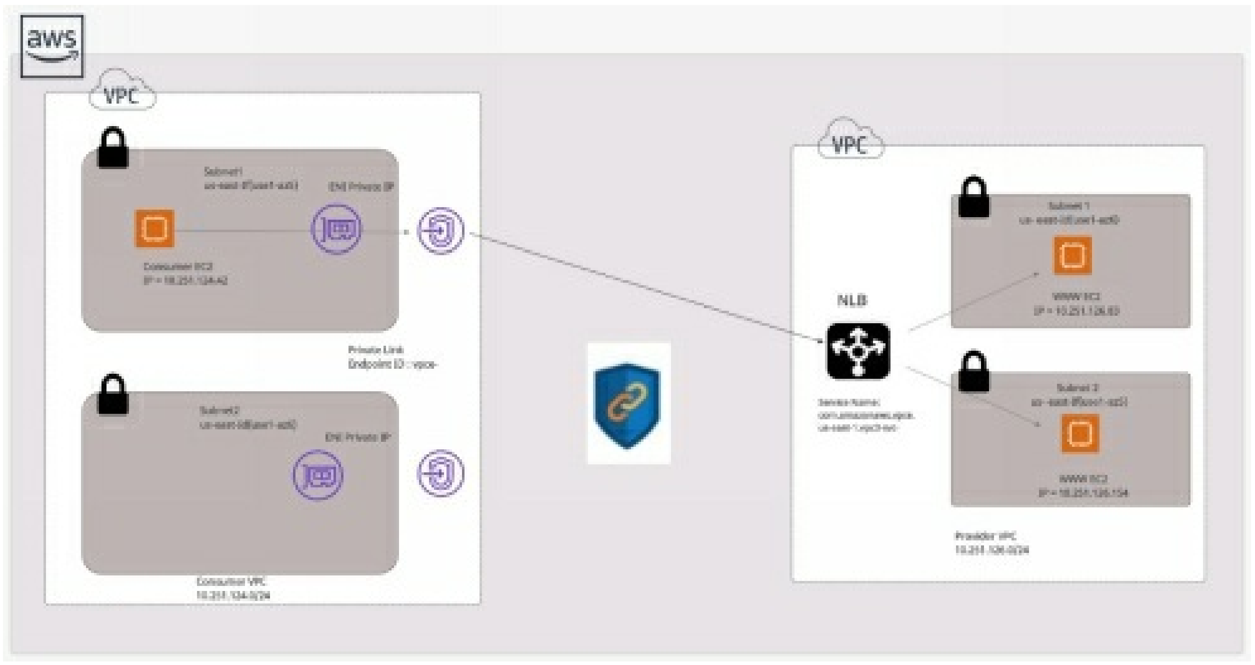
Usually, multiple VPCs can share the same services. They connect by using either a VPC public IP address across the internet or privately using the VPC peering or routing via on-premises location. AWS Private Link allows you to create service connectivity from service provider VPC to consumer VPC in a safe and scalable way. Besides, you can connect to the services that are available in a shared service environment or the AWS marketplace.



Use case

Let us have a look at the use case for AWS Private Link. In the same AWS region, there is a connection between a consumer VPC and the provider VPC. Thus, the provider VPC lies on the right side of the architectural diagram, which consists of two subnets. It hosts web services for external clients and the customer's internal products. The VPC Endpoints and the host services lie under the NLB. The consumer VPC, on the other hand, lies on the left side that consumes the services via interface endpoint. It also consists of two subnets that are identical to the provider VPC. Also, the ENIs must be available in the same AZ and same region.

To utilize AWS Private Link, the consumer VPC establishes an interface endpoint for the service in its own VPC. It will create an ENI in a subnet containing a private IP, which acts as an entrance for the traffic expected for the services on the other end. Thus, the service endpoints available via AWS Private Link will appear as ENIs with private IPs in the VPC. Then, the consumer VPC will connect to its Endpoints, and they will connect to the NLB of the provider VPC on the back end. In this way, Private Link can be used for establishing secure connections between AWS services and VPCs.



AWS GLOBAL ACCELERATOR

Introduction

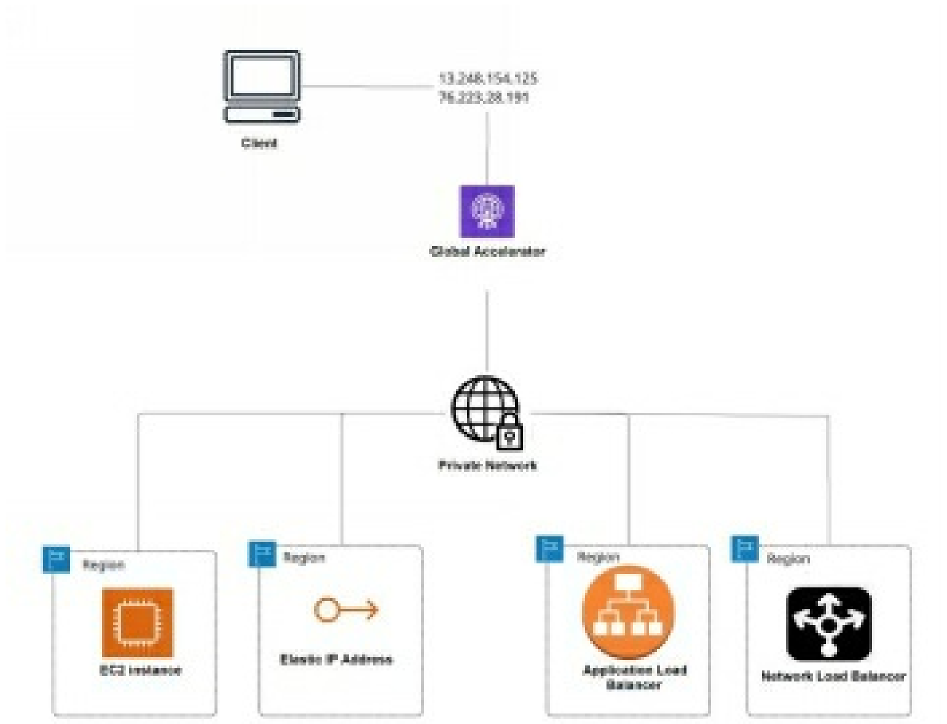
AWS Global Accelerator enhances the availability and performance of an application for both local and global users. It gives a static IP address which serves as a fixed entrance for the application endpoints in a single or multiple AWS regions, such as *ALB, NLB,* or EC2 instances. It uses AWS global network to improve the path between the users and application, boosting UDP and TCP traffic efficiency. Thus, AWS Global Accelerator continuously tracks the health of application endpoints. If it detects an unhealthy endpoint, it will redirect the traffic towards a healthy endpoint in less than 1 minute.

How does it work?

The Global Accelerator provides two *static IPv4 addresses*. You only need to define the endpoints in one or more regions. It supports a few Endpoints that are as follows.

- EC2 Instance (with or without Public IP)
- Internet-facing ALB
- Elastic IP
- Internet-facing NLB
- Internal Application Load Balancer (ALB)

It improves the route for all users, meaning that it reduces the hops until the UDP or TPC package arrives in Amazon's network and decreases the delays. Besides, only the healthy endpoints get entertained by the Global Accelerator route. The Global Accelerator establishes a peering link between VPC and the user's accelerator created with Amazon VPC. As seen in the figure below, the traffic between the VPC and Global Accelerator uses private IP addresses.



The Global Accelerator applies a couple of public IP addresses for improved latency. Besides, DNS names are used to point to both public IP addresses.

To make use of Global Accelerator, a user needs:

1. To establish an accelerator that provides a couple of Anycast static IP addresses.
2. To establish a listener for the port/port range or protocol.
3. To establish an endpoint group for each region where he wants to direct the traffic.
4. To introduce an endpoint such as an ALB for every endpoint group.

Most importantly, Global Accelerator is a useful tool for gradually moving the workload or shifting the traffic between regions and endpoints when deployments or interruptions occur. In case of a failure (when the health check finds a failed endpoint), redirecting the traffic towards other AZs, regions, or endpoints occurs in thirty seconds.

Use Cases

General Use cases for AWS Global Accelerator are:

- *Disaster Recovery:* A user can shift the traffic from those regions or endpoints influenced by an interruption to the next endpoint. The Anycast IP addresses ensure that the clients deliver requests only to healthy endpoints. Due to caching issues, it is not easy while using the Routing Mechanisms that depend on Route 53 (DNS).
- *Multi-Region:* A user can launch the infrastructure to several regions and direct the traffic to the closest region.
- *Low Latency:* It can be used in such circumstances where the latency is crucial, like trading and gaming.
- *Origin Cloaking:* A user can use the Global Accelerator as a protected & public endpoint (*AWS Shield*) for EC2 or ALB in a private subnet.
- *Static IP Address:* Few users (such as IoT tools) might not resolve the names by using Domain Name Systems. They might need static IP addresses for a Firewall rule (where NLB is the other option in that case).
- *Canary* or *Blue/Green Deployment*: Users can move the traffic between infrastructures for testing or during deployments.

SECURITY GROUPS

Introduction

In order to protect EC2 instances, SGs operate as a virtual firewall by controlling incoming and outgoing traffic. The SG contains *inbound* and *outbound rules* to filter out the incoming and outgoing traffic from that EC2 instance. These rules can be set up with a specific protocol, a port range, and the individuals who are permitted to have access. Security groups ensure security at the instance level. For instance, the outbound rule can be used to enable all network traffic to leave the instance, whereas an inbound rule can be used for allowing traffic only from a single IP address to enter the instance. Each security group can be attached to multiple instances because it operates at the instance level of a VPC. The SG and network interface are linked to each other, but the network interface is directly connected to an instance.

When a VPC is created, a default Security Group is automatically created for you by AWS. A security group cannot be deleted but, you are allowed to remove or add a rule to a default SG. There are no denying rules in security groups. Therefore, all the traffic is blocked by default unless a rule explicitly permits it. Moreover, multiple instances across multiple subnets can belong to a single security group. Let us assume that there are three different EC2 instances in different subnets. Remember that the security groups do not care about subnets. The user will assign an EC2 instance to a security group in this case so that all are connected and can communicate with each other.

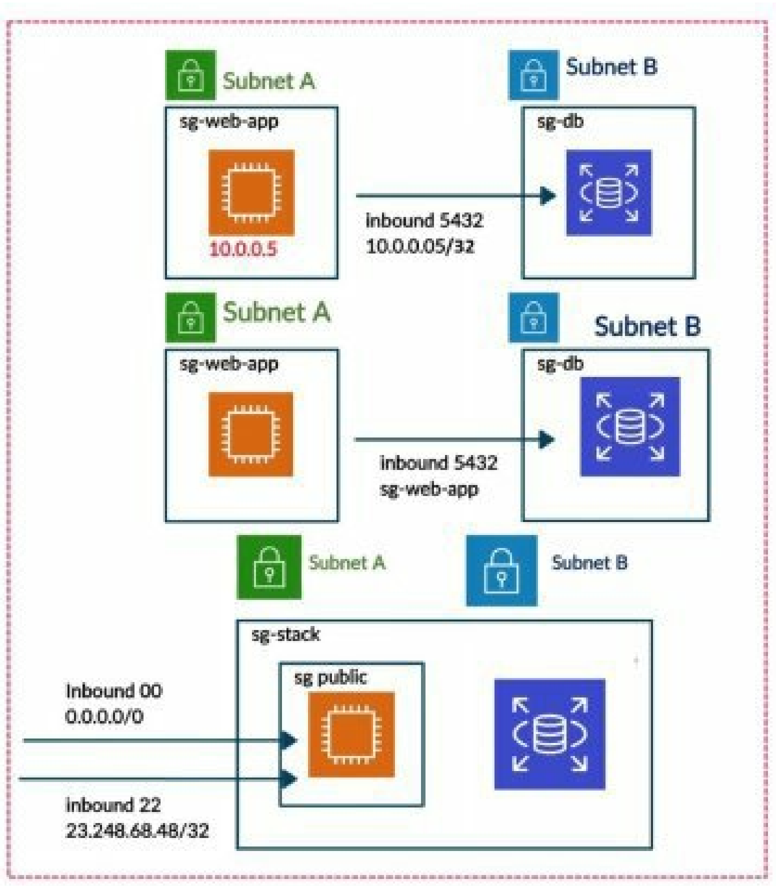
How does it work?

You can secure the cloud environment with AWS Security Group's help by managing the traffic flow into the EC2 instances. Using security groups, you can make sure that all your network traffic is flowing only through your

specified routes and protocols at the instance level. You must assign an instance to a certain security group at the time of deployment. You must assign a unique name to every SG that enables you to choose it from a menu. Rules may be added to all SGs as they will enable the traffic from and to the designated services, which includes associated instances. A security group must be created in the same VPC as the resources that need protection.
SG rules are always permitted, just like white lists. The rules that deny access cannot be created. For example, a subnet containing web servers receives the traffic from an ELB. So this ELB will be listed as the only permitted source by your SG. Security groups are stateful, meaning that the outbound request will surely pass if an inbound request passes.

Use Cases

There are three use cases for a security group that perform the same functions with different configurations. To understand the variations in which you will perform events, let us assume a web application working on the EC2 instance. It is connected with the RDS database to retrieve the information from a private subnet. In the first case scenario, there is an inbound rule on a database that allows traffic from Postgres port number *‘5432’* for this specific IP address. It will allow the EC2 instance to connect to that RDS database. Therefore, you can specify the source to be an IP range or a specific IP. A better way of declaring one IP address is /32.



The second scenario is the same as the first use case. The only difference is that instead of providing an IP address as the source, you can provide another security group. Therefore, anything within the security group can gain access to inbound traffic on *5432*. In the last use case, you have inbound traffic on port *80* and port *22* for the SG public group. You have an EC2 instance and the RDS database as well residing in its security group. Therefore, the EC2 instance can communicate with the RDS database without exposing it.

Moreover, there will be no Public IP address in it as it resides in a private subnet. However, the EC2 instance can obtain traffic from the internet. In short, an instance can belong to multiple security groups where rules are nonrestrictive. Therefore, assume that there are two security groups, and one of them allows the traffic. So, this security group will take priority over the SG stack, as it does not have anything and denies everything by default. However, it can override the previous rules when it permits new traffic. Thus, you can also nest multiple security groups into one.

Limitations

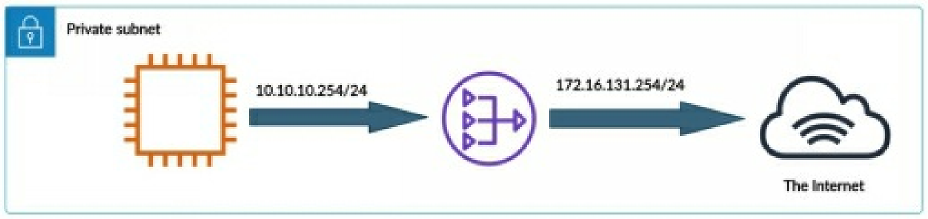
Security groups have some limitations. You can have a maximum of 10,000 security groups in a region, out of which it allocates 2,500 by default. If you want to go beyond 2,500, you need to request AWS to increase service limits. For each security group, 60 inbound and outbound rules are allowed to have and 16 security groups per ENI. Five of them are already present by default. Therefore, the number of security groups connected to an instance depends on the number of ENIs attached to that security group. It means that if there are two ENIs attached to a security group, there will be ten security groups by default. On the other hand, if the upper limit is 16, there will be 32 security groups on a single instance.

NETWORK ADDRESS TRANSLATION

Introduction

Let us talk about the role of *NAT* in AWS network architecture. It is a method of remapping an IP address space into another. NAT devices are used to connect the private subnet instances with the AWS services or the internet. However, the internet is restricted from creating links with the private subnet’s instances. With NAT devices, private subnet traffic is directed towards other AWS services or internet, and in return, the instance gets its response. As soon as the traffic is routed to the internet, the NAT device address takes the place of your instance's source IP address, and when the traffic returns, the NAT device converts it to your instance's private IP address.

Let us assume that there is a local network with its personal IP address space. The IP address changes as it passes through NAT. If you have private networking, you need help in getting outbound access to the internet. Therefore, you must use the NAT gateway to remap those private IPs. Secondly, when there are two networks with overlapping network addresses, you can utilize the NAT to make the addresses more appropriate for communication.



Types of NAT Devices

There are two options to introduce NAT in AWS, which are *NAT Instances* and *NAT Gateways*. Let us see the comparison between these two devices. The *NAT instance* needs to be configured, which is a regular EC2 instance used for remapping. The NAT instances must be launched in the public subnets to work because it requires internet access, and the internet cannot be accessed with a private subnet.

On the other hand, *NAT Gateway* can set up the EC2 instances for the users. You do not need to access it because it is a managed service. Each NAT gateway is built with redundancy in a particular AZ. NAT gateway will not just launch one but many redundant instances for you. It is because when you launch your own NAT instance, and if it gets taken down, then you would need to run it multiple times. It is done to achieve the desired durability and scalability of these instances based on your traffic. NAT gateways will take care of this problem by automatically creating redundant instances in the same AZ for you. The only drawback of NAT gateway is that it does not launch them across other AZ's. Thus, you need to launch a NAT gateway in each AZ, which results in redundant instances.

Moreover**,** you can use NAT AMI for launching a NAT instance, which will then run on your VPC as an instance. One must deactivate the source and destination checks on instances while creating a NAT instance. There is excessive manual work involved for users to achieve high availability, scalability, and durability for NAT instances. To make a way out from the private subnets to the NAT instances, they must reside in the public subnets. The amount of traffic that NAT can handle is determined by its size. One can achieve high availability by using Auto-Scaling Groups. You can automate failovers between multiple subnets in different AZ's by using a script.

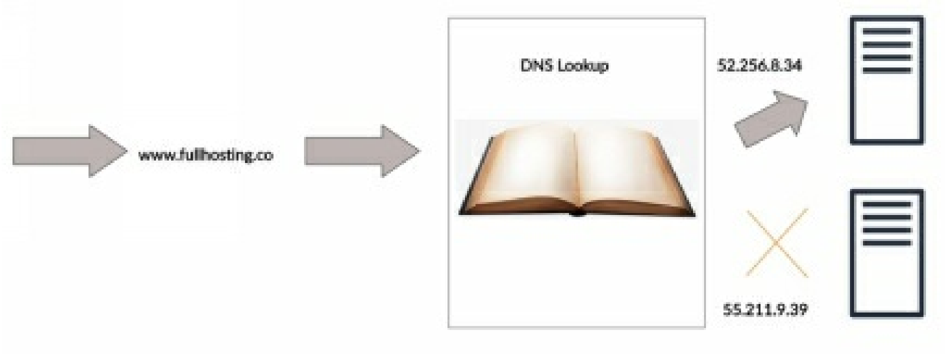
On the other hand, NAT gateways are redundant inside the availability zone to sustain an EC2 instance's failure. You cannot span multiple AZ's because only one NAT gateway can be housed in one AZ. They are considered the perfect fit for an organizational system. There is no need to patch the NAT gateway and disable the source or destination checks on it. The public IP address gets automatically assigned to a NAT gateway. Route tables for the NAT Gateway need to be up-to-date. If the NAT gateway goes down, the internet access of resources sharing the same gateway in multiple AZ environments will be lost. However, if you create a gateway in every AZ and configure the route tables accordingly, you will not face this issue. Usually, users prefer to use NAT gateways because it is the updated approach, whereas the traditional approach is still available in AWS.

DOMAIN TERMINOLOGIES

Before we start with Route53, let us understand some important terminologies.

DNS

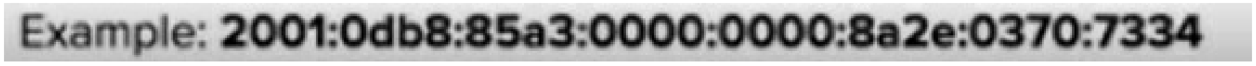
Mostly, the descriptions of *Domain Name System* (DNS) use the analogy of a phone book. It is a protocol inside the set of standards that monitors computer data exchange over the internet and several known private networks. An essential function of DNS is that it converts domain names such as '*fullhosting.com'* into IP addresses like *'70.42.251.42'*. Computers use these IP addresses to identify each other over the network. It enables the user's computer to find specific servers on the internet automatically, which depends on the domain name that he is browsing.



Internet Protocol

An *IP address* is exclusively used for identifying computers on a network. It allows communication between them by using the *Internet Protocol*. There are two types of variations in IP addresses, which are *IPv4* and *IPv6*. IPv4 is the most familiar IP address among users since it is in use for a long time. It is a *32* bits address space with *4,294,967,296* available addresses. You might run out of IP addresses because there is a limit for writing numbers.

Therefore, IPv6 was introduced to overcome this issue, which uses *128* bits address space having *340 undecillion (1+36 zeros)* potential addresses. It is lengthy and complicated to look at, but you will not run out of address. However, you can use both types of IP addresses on AWS. The following example shows the address of IPv6.


Example: **2001:0db8:85a3:0000:0000:8a2e:0370:7334**

Domain Registrars

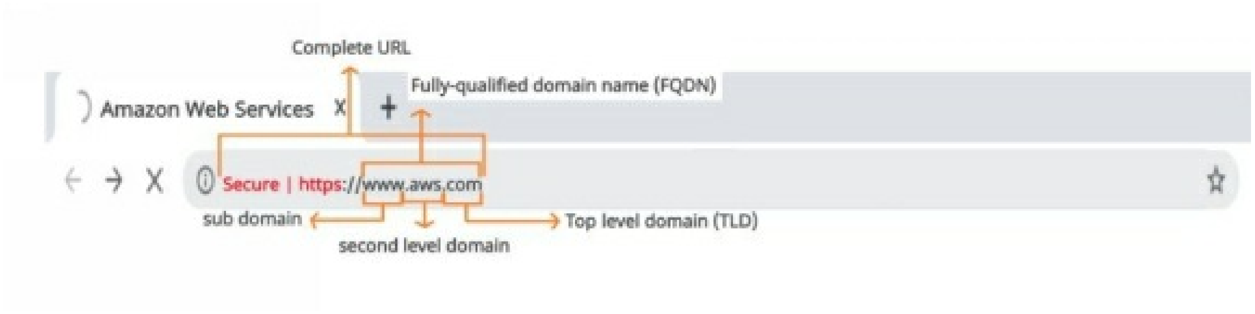
Domain Registrars are the authorities that allocate domain names against one or more *Top-level domains.* AWS has its domain registrar with Route 53. The *InterNIC* service (provided by *ICANN)* is used to register domains and makes sure that domain names are unique throughout the internet. After registering a domain name, it will be available publicly in the central *WhoIS* database.

WHOIS is a public directory where you can find who is accountable for a domain or IP address. *WHOIS* directory can be used to find out the owner of the domain. It includes his phone, email, name, and address. Therefore, you can find the domain owner by typing it in the *WhoIS* database, where the Registrar contacts will be visible. In the case of Route 53, you can pay additional money for keeping the information private.

Top-level Domains

There are different levels of domains, which include *Top-Level Domains* and *Second-Level Domains.* When you select a domain, you need to establish a unique combination of bits before and after the *'dot.*' A *Top-Level Domain* is a domain section that appears after the dot-like org, net, or com and Internet-Assigned-Numbers-Authority (IANA) manages these domains. A domain can relate to or unrelated to a geographical region.

- The domains that are not related to a country are known as Generic top-level domains. The most common gTLD domains include com, org, and net. They belong to ICANN's policies.
- The domains related to a geographical location or a country are known as Country-code top-level domains. The ccTLD consists of two letters as de, NL, or DK.



A *Second-Level Domain* (*SLD*) is the section of a domain name that appears to the left side of the *'dot*.' On the other hand, a TLD belongs to the right side of the dot. It is also known as a domain extension. Let us take an example of **'**fullhosting.com**.**’ In that case, ‘fullhosting’ is SLD, whereas ‘.com’ is TLD. Different organizations manage these domains. There are a vast number of top-level domains, such as Disney has its domain **‘.***abc***,'** and AWS has its ‘.*aws’* domain.

Now, it is time to head over to our main topic, i.e., Route53.

ROUTE 53

Introduction

AWS Route 53 provides a cost-effective and secure Domain Name System for companies and developers to route users to their web applications. It is used to translate the domain names into IP addresses for routing traffic to a website. Users use AWS Route 53 for their domain management. It is a reliable and affordable DNS service for businesses and developers. Specifically, it is useful for managing AWS infrastructure. In simple words, there are three purposes of Route 53:

- Route Internet traffic,
- DNS Registration, and
- Health Check status

In Route 53, domain requests are resolved by a global network of DNS servers, which makes it fast and reliable. The Domain Name System is an internet routing layer. It maps domain names in human-readable format into machine addressable IP addresses such as *www.fulllhosting.com* into 23.21.47.33. It is used especially for registering and managing domains. Various record sets can also be created on a domain using Route 53 and implementing complex traffic flows like blue/green failovers. Besides, you can also resolve VPCs outside of AWS and continuously track the records via health checks. Using Route 53, an organization can easily track and route global data traffic. AWS provides a user-friendly and simple Restful API for its management in the Command Line interface. Thus, it serves as a programmable DNS for the infrastructure as well.

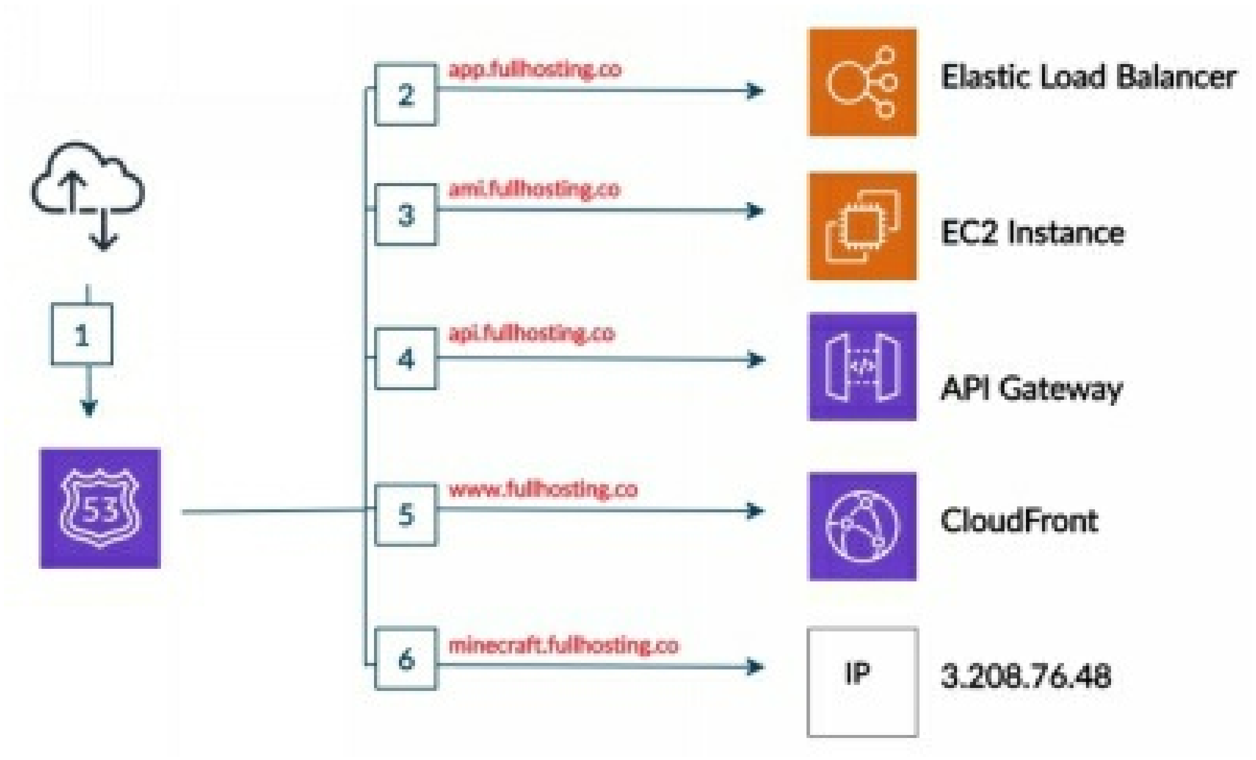
How does it work?

Lets us look at the overview for the working of Route 53 to route the traffic between the hosted web apps and end-users:

1. First, the domain name gets registered with Route 53 and then configured for routing the internet traffic to the servers. Both AWS public and private cloud infrastructures can act as the servers hosting the domain name.
2. A complete URL or the domain name is entered by the users in the search bar of any web browser.
3. The request is routed to the DNS resolver by the Internet Service Provider.
4. The DNS resolver forwards the user request to its root name server, and then it is routed towards the Top-Level Domain server. Eventually, it is directed to Route 53.
5. The domain name IP address is returned by the Route 53 name server to the DNS resolver.
6. When it receives the required IP address, it will direct the user request to the content hosted server according to the Route 53 service configuration.
7. The health of backend servers is also monitored by Rout 53. There is a feature known as *DNS Failover*. It is used to check the endpoints for availability. In case an endpoint turns out to be unhealthy, Route 53 will direct the traffic towards some other healthy endpoint. With the help of CloudWatch, an alarm can be triggered to notify the recipient to take necessary action.

Use Case

Consider a use case in which you have a domain name *'fullhosting.co'*. You have Route53 to manage the name servers allowing you to set the Record Sets within Route53. There is a set of various Record Sets for sub-domains, and you want these subdomains to point to different resources in AWS, as shown in the diagram below.



Thus, your app uses 'app.fullhosting.co' *to* point out to ELB because this app runs behind it. When you need to work on the AMI image, an EC2 instance can be launched and pointed to the subdomain 'ami.fullhosting.co' there**.** If the API gets powered by an API gateway, you can use 'api.fullhosting.co'. For static website hosting, you can point out to CloudFront and use **'***www.fullhosting.co'* for Cloud distribution. You might run a Minecraft server on a specific IP from a learning perspective, which is probably an elastic IP, as you do not want it to change. Therefore, you can use '*minecraft.fullhosting.co***'** for this purpose**.** This is a very simple example; however, now we will learn the features of Route53.

Features of Route53

Record Sets

We observed several subdomains in the previous use case, which pointed to AWS resources. By creating *Record Sets*, you can create the link for Route53 to point to these resources. You can create Record Sets from different types of records. You only need to fill your subdomain or even leave an empty domain and choose the type. In the case of the A-IPv4 address, it allows you to direct the subdomain to a specific IP address. Then, there is a unique option created by AWS for Recordsets known as *the Alias* option, which allows you to select the AWS resources directly. Thus, you can select Elastic Beanstalk, CloudFront, S3, ELB, API gateway, or VPC. You would need to do this by making a traditional record type because Alias can detect IP addresses' changes and keeps on pointing that endpoint to the accurate resource. It is recommended that one must use Alias because it makes it easier to manage the connection between resources through Route53 Record Sets.

Health Checks & Failover

The ability to perform *Health Checks* is another very important function of Route 53. They are good for monitoring the performance and health of web servers, web applications, and other resources. They can monitor one of the following:

- Amazon CloudWatch Alarm Status
- The performance of allocated resources, such as web servers
- The other health-check status

When a health check is created, you get its status and receive the notifications in case of any status changes. Furthermore, *DNS Failover* can also be configured. Therefore, health checks regularly monitor the health of an instance. You can check health by default after every 30 seconds and decrease it to 10 seconds. A health check can start a failover if the status comes back as unhealthy. CloudWatch alarms can be used to inform you about the unhealthy status. A health check can also monitor other health checks to create a chain of reactions. Users can have 50 health checks in a single AWS account at the affordable cost having $0.50 per endpoint.

Resolver

You can route DNS queries between your VPC and network with the help of Route 53 Resolver. It is a regional service and officially known as '.2 resolver'. Therefore, it is a tool for cloud and on-premise hybrid environments. It offers automated DNS resolution in the user VPCs and entertains DNS queries for domain names by default, such as ELBs or domain names for EC2 instances.

A resolver performs repeated lookups against public name servers for all other domain names. In the case of on-premises instances, they cannot resolve Route 53 DNS entries and vice versa. Thus, you can configure DNS resolution between the in-house local network and AWS VPC through a VPN connection or Direct Connect. You need to create a Resolver endpoint in the VPC to use inbound or outbound forwarding.

Traffic Flow

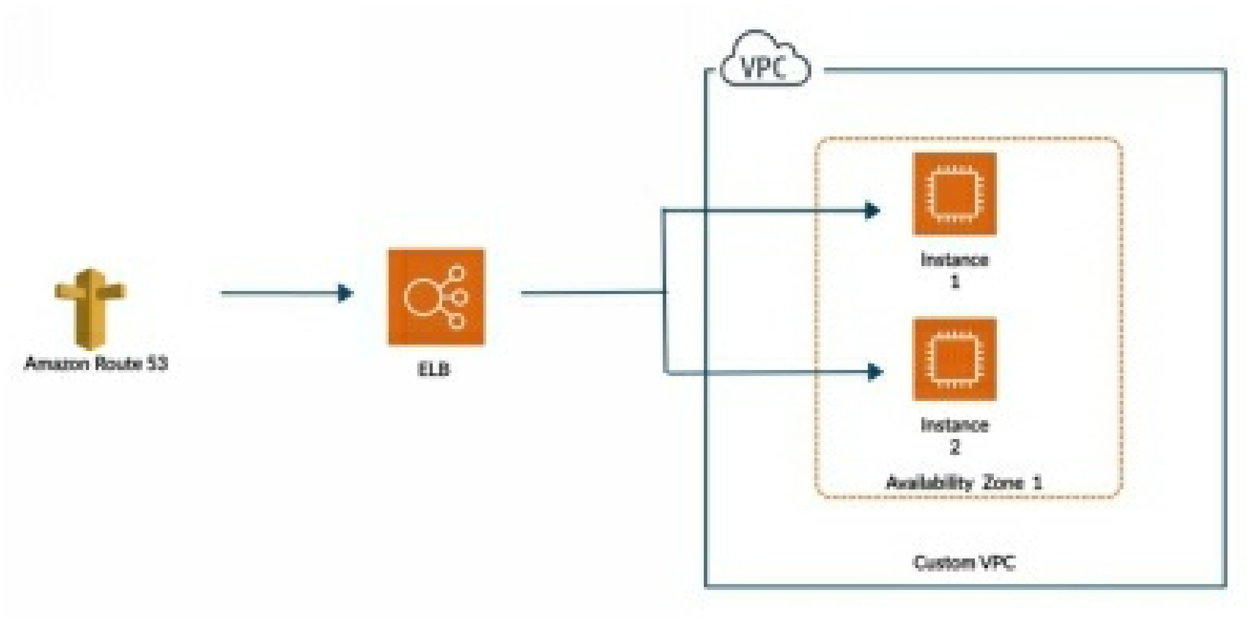
Traffic Flow is a visual editor that allows users to create complex routing configurations within Route 53 easily. The other advantage of traffic flow is the versioning of these policy routes. Let us assume that you create a complex routing policy, and the next day you want to change it. Therefore, you could save it as version one and enroll in version two, or we can say you can roll one out or roll it back to updates. You need to pay per policy record for dealing with traffic flow. However, you do not pay until you create it. You can work with it to get the idea of various routing rules and generate creative solutions for your infrastructures. As we have discussed traffic flow now, let us discuss the Route53 routing rules in detail.

Routing Policies

There are seven types of *Routing Policies*, which is a major advantage of Route 53. You can select a routing policy while creating a record. It determines the response of Amazon Route 53 towards queries. To understand these policies, we will discuss their use cases individually.

Simple Routing Policy

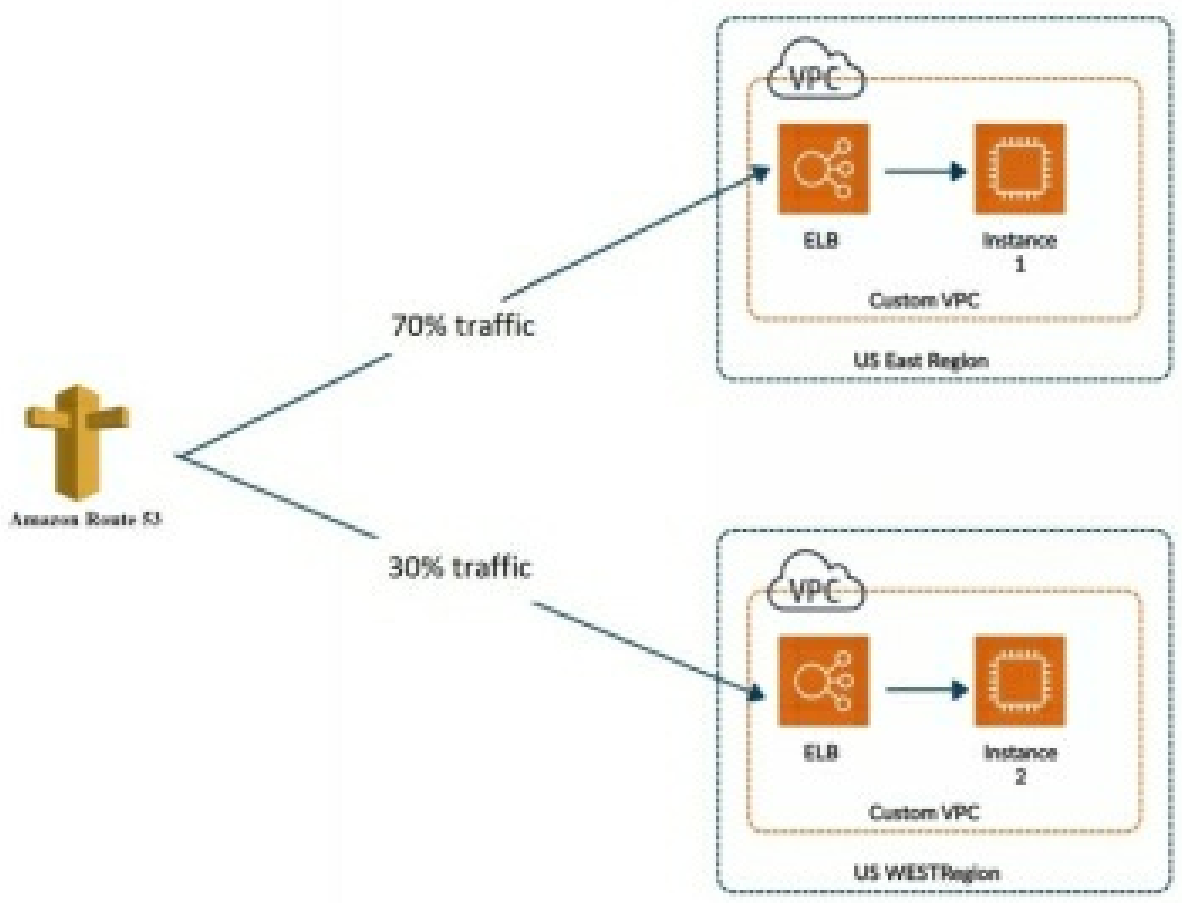
Simple Routing Policy is the most fundamental routing policy. It always uses an *A record* to resolve a single resource without any specific rules. You can direct the traffic to the single resource with this policy, e.g., to the webserver of a service provider. In Route 53 console, multiple records cannot be created with the same name and type using a simple routing policy. However, multiple values can be specified in the same record, such as multiple IP addresses. Let us say that a DNS record gets created to settle the domain to the ALIAS record. It will route the traffic to an ELB, which will load balance the set of instances.



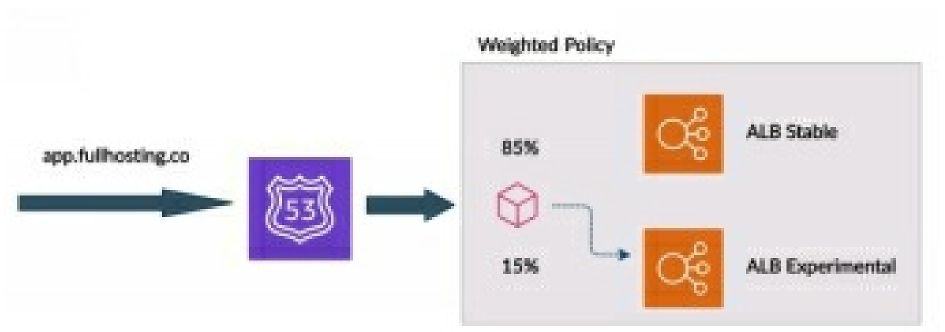
When creating a record, you need to set its name and type. In this case, let us assume that the name is Random, and the type is A-IPv4 address**.** The Routing Policy is always set to simple by default. It means that you can provide more than one IP address using a simple routing policy. A single IP means the Random will move towards the first IP address every time. On the other hand, if there are multiple IPs, it will randomly select one.

Weighted Routing Policy

When multiple resources are required for the same functions, users apply *Weighted Routing Policies* to split traffic between the resources relying on a predefined weight. It is useful for several reasons, such as testing new versions of software and load balancing. It allows you to split the traffic depending on various assigned weights.

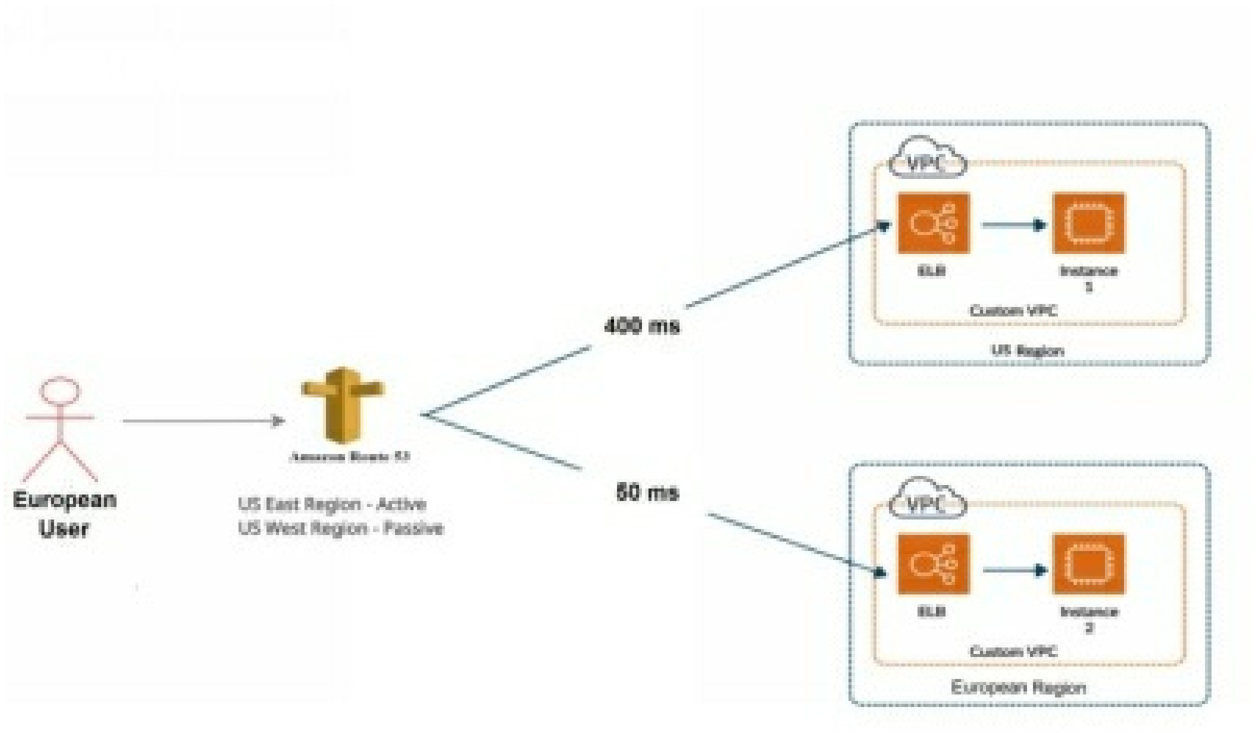


Let us discuss the use case for this policy. Let us say you have a domain ‘app.fullhosting.co,' and create two identical Record Sets in Route 53 using this domain. Therefore, for a weighted routing policy, you need to set two different weights for each of them and set them both to weighted policy. Let us suppose that you create the first recordset, '*Stable,’* and it is set to *70*%. Then, the other record set, 'Experiment,' is created with the same sub-domain and is set to *30*%. Now, when the traffic hits ‘app.fullhosting.co,' it will observe the two weighted values. The *70*% network traffic will go to *Stable*, whereas *30*% of traffic will go to the *Experiment* recordset. It was a simple use case to help you grasp the concept. However, it is recommended to test a limited amount of traffic to mitigate the effect while testing new experimental features.

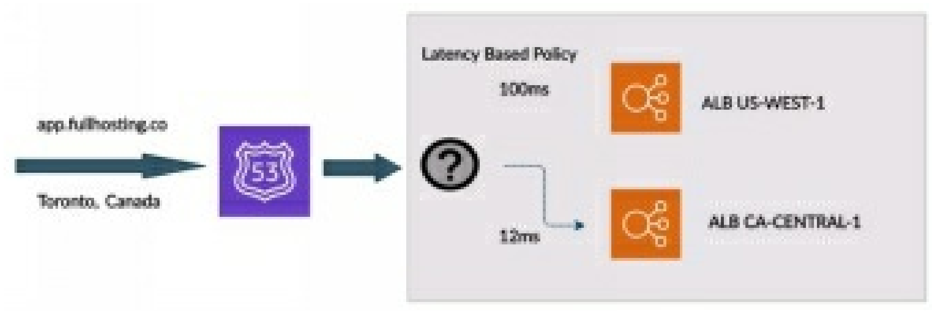


Latency Based Routing Policy

This policy enables you to direct traffic depending on the lowest possible network latency for the region-based end-users. Users apply this policy when there are multiple resources for the same functionality. Route 53 is intended to address DNS queries with the high latency possible, e.g., the region that offers the fastest response time.

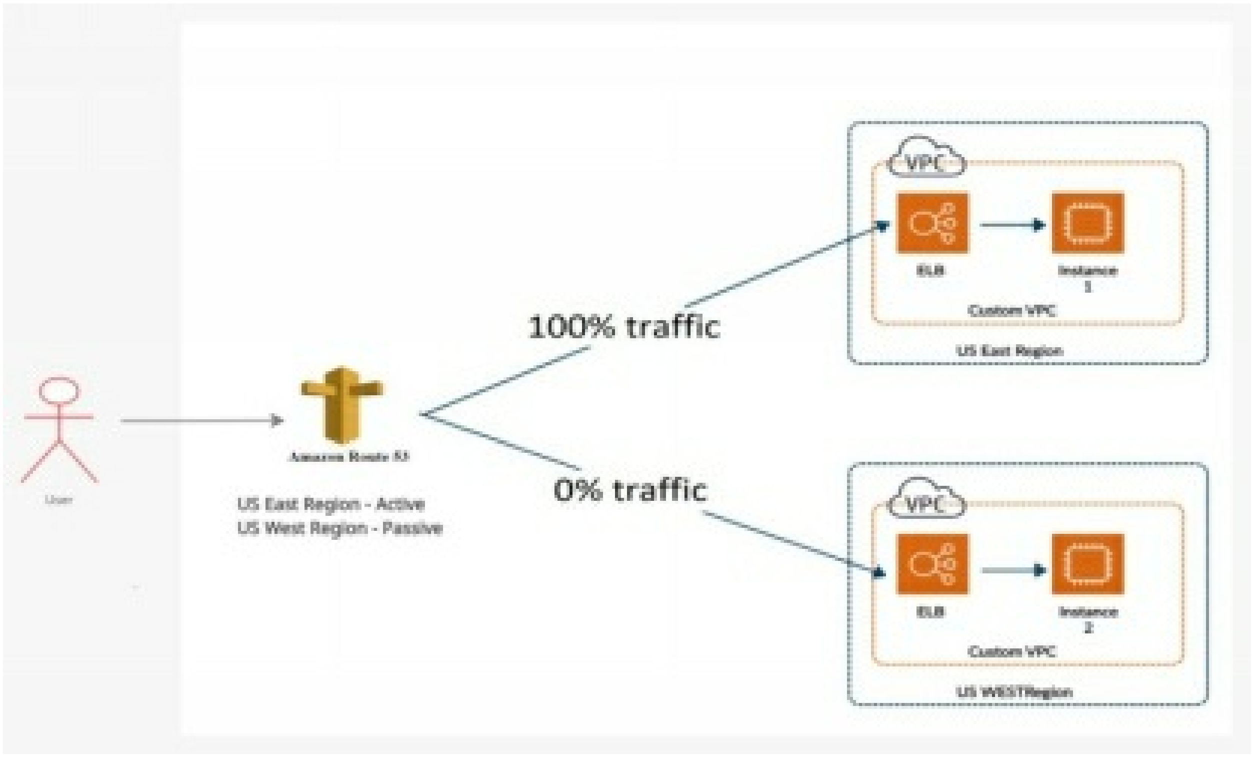


Consider a use case in which the traffic from Toronto hits your domain app.fullhosting.co**.** Therefore, you create two records having latency with the sub-domain. Let us say that one is set to *US-WEST*, which is located on the west coast, while the other to *CA-CENTRAL*, located in Montreal. Thus, you have set the policy for latency-based routing, which will observe the records with the least amount of latency. The one with the lowest return in milliseconds is the one to which the traffic will be redirected. Besides, it does not have to be the nearest one geographically. Logically, the closer ones should be faster. In this case, *CA-CENTRAL* has 12 milliseconds of latency. Thus, it will route the traffic to ALB *CA-CENTRAL*. That is how Latency Based Routing Policy works.



Failover Routing Policy

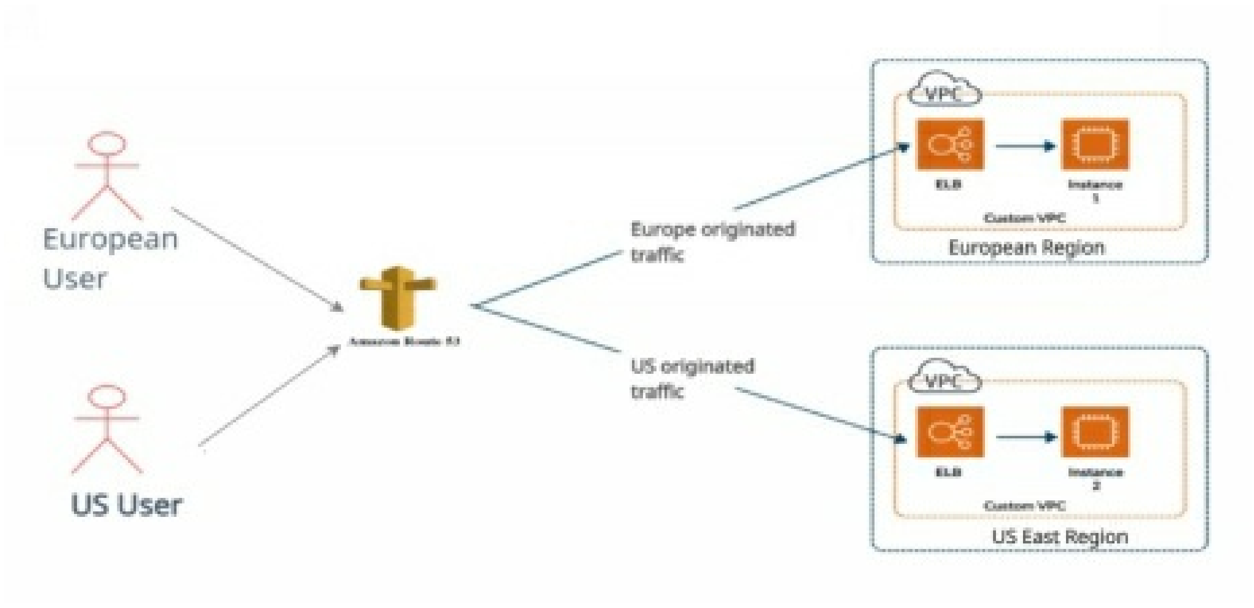
As the name reflects, this Routing policy is concerned with failovers. It allows you to create an active and passive setup in a situation where you want a primary site in one location and a secondary data recovery site in another location. Moreover, Route 53 automatically tracks the primary site through health checks to determine the endpoints' health. If the endpoint is not healthy, then it will automatically redirect the traffic towards the secondary location.



Let us assume the same domain ‘app.fullhosting.co’ with instances in primary and secondary locations. Thus, Route 53 will run the checks on these endpoints with the help of a health check, and if it finds the endpoint unhealthy, it will redirect the traffic to the secondary location. Therefore, you have to create two routing policies with the same domain and choose the primary and secondary locations.

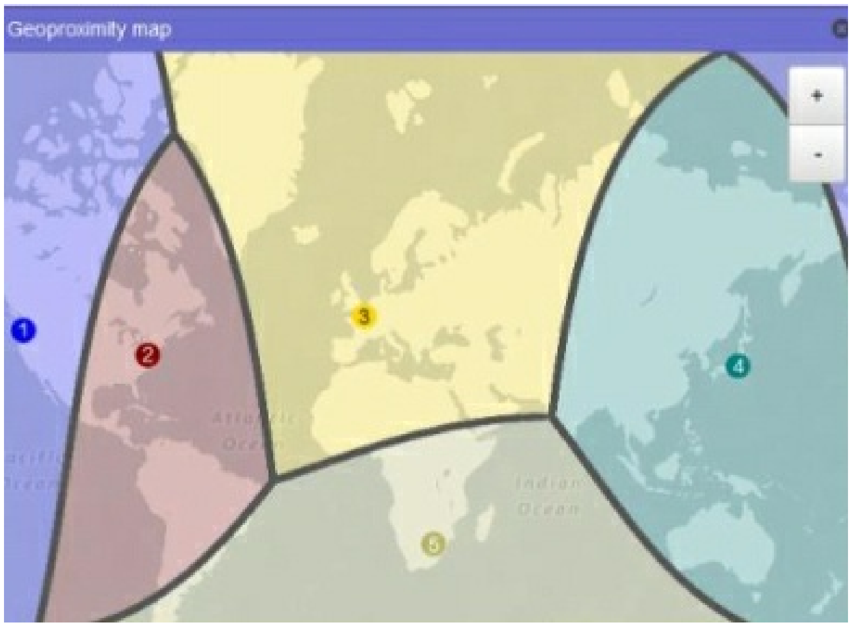
Geolocation Routing Policy

Geolocation Routing Policy enables users to direct the traffic based on geographical location, which is the request's origin. This policy enables traffic to be routed to resources in the same location from which the request was sent, i.e., it allows user locations to have site affinity. Let us assume that a request approaches ‘app.fullhosting.co’ from the US. The Record Set for Geolocation is set for *North America*. So as the US is in North America Region, it will go to this set of records.



Geo-proximity Routing Policy

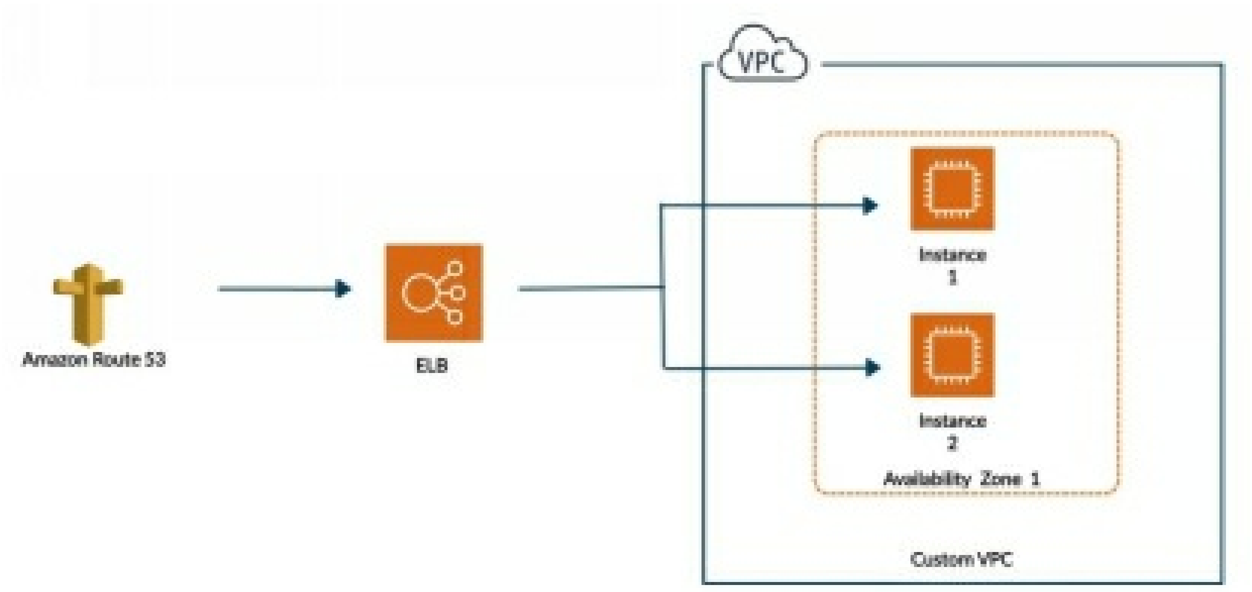
Route 53 directs the traffic to certain resources using *Geo-Proximity Routing*, which depends on your geographic location and resources. Alternatively, you can also select to move specific traffic to a specific resource by entering an attribute called a *Bias*. Bias increases or decreases the geographic region's size, which routes the traffic to a resource.



The geo-proximity Routing Policy is perhaps the most complicated one because it sounds like Geolocation. One cannot create it using Record Sets. You must use traffic flow because you have to observe the traffic activity continuously. So, you can set as many regions or points as you want. While selecting a region, you can either choose from one of the existing AWS regions or set your own coordinates. Let us say that you create a Geo-proximity routing for a region. In that case, you give a *bias* around a specific location with certain boundaries. If you give a region *25*% more bias, then this area will expand. On the other hand, if you reduce *25*% from a region, it will shrink back to normal. So, this is how the policy on Geoproximity routing works.

Multi-Value Answer Policy

Multi-value Answer Routing Policy is the same as the Simple Routing Policy. However, it returns multiple values in response to DNS queries, which can be a web server's IP address. Multiple values for each record can also be specified. Besides, a multi-value answer policy allows you to perform health checks for every resource. It is not a replacement for ELB, but it is a way of returning multitudes of health detectable IP addresses that uses DNS to boost availability and load balancing.

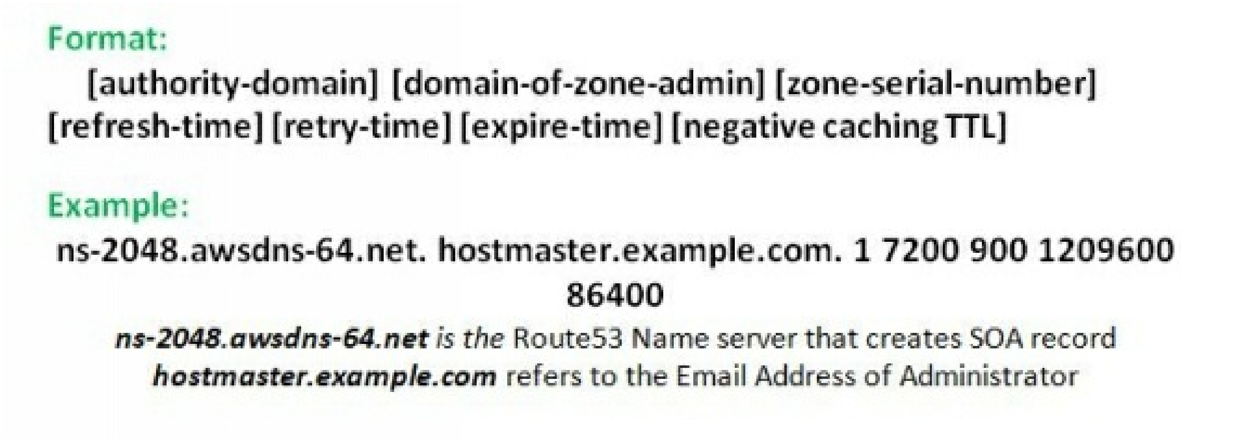


Records

Let us talk about various types of records in AWS Route53.

SOA (Start of Authority Records)

Whenever a domain name generates, the users require a bunch of records that inform them what to do. The *Start-Of-Authority* (*SOA*) is one of the records in which domain admins provide additional information about a domain. It includes the admin's email address and how often a domain is updated. For example, it includes the time information taken to perform a failover to the secondary namespace if the master fails. Below is an example of the format of SOA:



Format:

[authority-domain] [domain-of-zone-admin] [zone-serial-number] [refresh-time] [retry-time] [expire-time] [negative caching TTL]

Example:

ns-2048.awsdns-64.net. hostmaster.example.com. 1 7200 900 1209600 86400

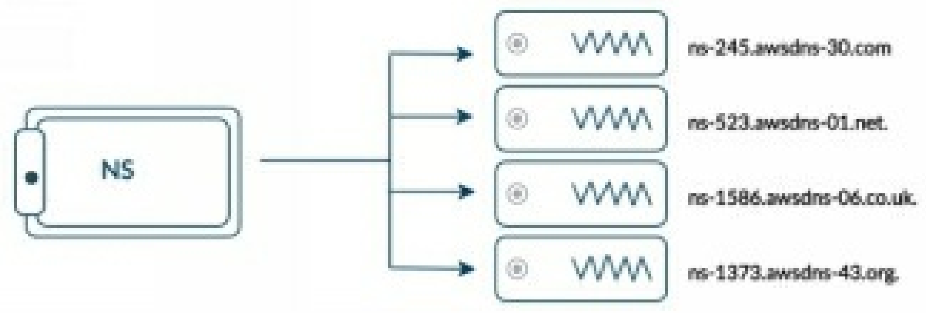
ns-2048.awsdns-64.net *is the* Route53 Name server that creates SOA record

hostmaster.example.com refers to the Email Address of Administrator

However, you do not need to provide all the information because it is optional. Furthermore, you cannot have more than one SOA in a single zone.

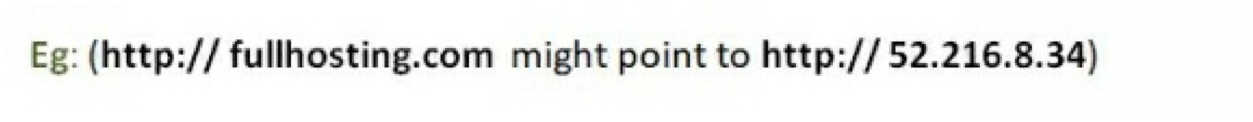
Name Server Records

The second most essential records after the SOA are *Name Server Records* (*NS*). Top-level domain servers use these records to route traffic to the DNS server holding the DNS records authority. The user domain name would not operate well in the absence of these records. The user will come across several name servers that cause redundancy. For instance, *GoDaddy* provides two name servers, and *AWS* provides four name servers. Let us say you manage your DNS records with Route 53. Thus the NS records for the domain will link to AWS servers, and you will have four records with too much redundancy. These type of records include.*co.uk*, *.com*, *.org*, and *.net.*



Address Records (URL to IPv4)

Let us have a look at *Address Records* or *A Records*. They are the most fundamental type of DNS records. They enable a user to convert the domain name directly into an IP address. Let us assume that you have a domain, '*fullhosting.com,'* and you need to point it to the IPv6 address *'52.216.8.34'*. Thus, in that case, you can use A Record to translate this domain into IPv6 address**.**


Eg: (**http:// fullhosting.com** might point to **http:// 52.216.8.34**)

Besides, you can also use naked domain names when there is no subdomain like 'www' as an *A* record.

CNAME Records (Canonical Records- URL to URL)

CNAME stands for *Canonical Names.* It is another fundamental DNS record used to resolve one domain name to another instead of IP addresses. You can send all naked domain traffic to the *www* route. Thus, you will specify the naked domains and send them to look like the *www* domain. Therefore, one can consider *A Records* as IP addresses and *CNAMES* as domain names.

Alias Records

These records indicate a URL to the AWS resource. You can use Alias records for mapping the resource record sets in your hosted area to the websites using S3 Buckets, ELB, or CloudFront.

Time-to-Live (TTL)

TTL is the duration of time the DNS record uses to resolve servers or the user's local machines. The shorter the TTL, the faster the changes in DNS records will transmit over the internet. You must specify Time to Live while creating a recordset.

AUTO SCALING GROUPS

Introduction

Auto-Scaling Groups or *ASGs* allow their users to set scaling rules to automatically launch additional EC2 instances or shut down instances to meet current needs. ASGs consist of a set of EC2 instances treated as an automatic scaling and management group. The automatic scaling occurs via health check replacements, scaling policies, or capacity settings. Let us discuss autoscaling via capacity settings first.

Capacity Settings

Capacity Settings are the best way to use Auto-Scaling Group without any configurations. Thus, there are three options: '*Desired Capacity,' 'Min,'* and '*Max*.' The *'Min'* is the least number of EC2 instances allowed to run, and **'***Max*' refers to the most instances that can be run. In the case of 'Desired capacity,' it is the number of EC2 instances a user wants to run.

Consider a situation in which you have a new auto-scaling group where *Min* is set to one. When you launch it, and nothing is running, then the ASG will spin up one instance. Furthermore, a server will always spin up at least one instance if it crashes for being unhealthy. Then there is an upper level where you cannot exceed two instances because the ASG could trigger more instances. It is like a safety net that ensures that you do not have multiple running servers. The 'Desired capacity' is the user's required capacity that he wants to execute. Therefore, AGS will try to obtain that value; however, there is no guarantee that it will always be the same value. The auto-scaling disables the EC2 installation and spins up a new one if Min functionality is one.

Health Check Replacements

Health Check is another way of auto-scaling to occur with Auto-Scaling Groups. ELB and EC2 health checks are the two forms of health checks. Let us discuss the EC2 type first. When you apply a health check, it will check whether the EC2 instance is healthy or not depending on two checks performed on the instance. EC2 is considered unhealthy if any of these two checks fail. If *Min* capacity is one, then the auto-scaling will spin up a new EC2 instance by disabling the current one.

In the case of ELB type, the health checks depend on the elastic load balancer. It performs health checks by pinning an endpoint on the server, which could be HTTP or HTTPS, and expects a response. Let us assume that you want 200 back at that specific endpoint. Therefore, if you have a web application, you need to create a small page called health check, which will return to 100. If it does so, it is considered healthy, or else, the auto-scaling group will disable that instance and spin up a new one for ELB by setting *the Min* value to one.

Scaling Policies

Finally, the most important way of triggering auto-scaling with ASG is *Scaling Policies*. There are three types of scaling policies. Let us start with *Target Tracking Scaling Policy*. It maintains a specific metric at a target value, meaning that you can select the metric type with an average CPU utilization. Let us say that the target value is set to 75%. Thus, if it exceeds this value, then you can introduce another server. Besides, if you add instances, it means you are scaling out. While, if you remove instances, you are scaling in.

The *Simple Scaling Policy* is the second type of scaling policy. It scales when the alarm breaks out. You create the desired alarm and ask it to scale in or scale out. This scaling policy is not suggested for current times because it is old. There is another similar yet more robust policy known as *Scaling Policy with Steps.* This policy allows its users to scale, depending on when an alarm breaches, but based on the changing alarm value, it can also increase the number of instances. In the case of Simple Policy, there was a single value. However, in this case, the alarm value changes with time. Let us consider a use case where you set the alarm between one and two and then add one instance. When it moves between two and three, it will add another instance, and by exceeding three further, it adds another instance. Therefore, it helps you to grow depending on the alarms that change over time.

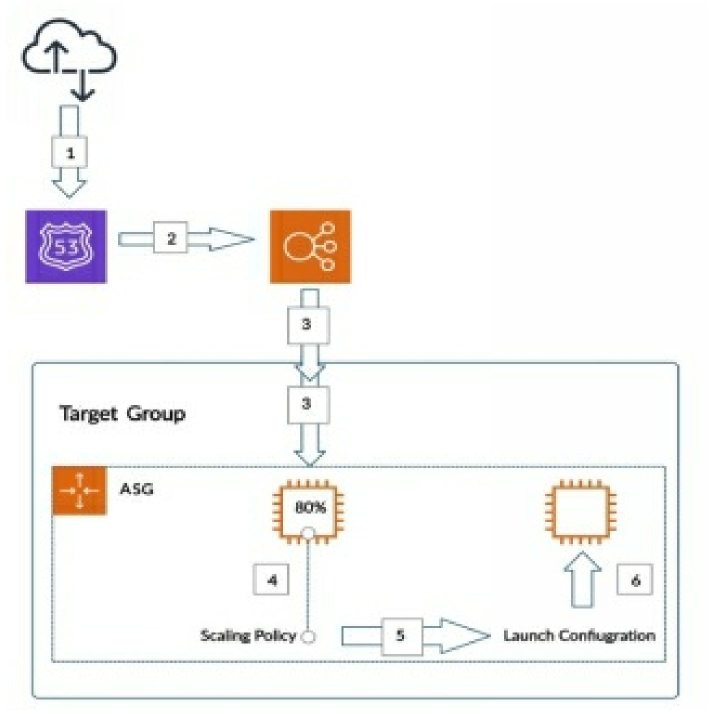
ELB Integration

We have learned that you can perform health checks based on ELBs. However, in *ELB Integration*, you can observe how ELBs are linked to an ASG. There is a bit of diversity in the ELB integration based on the kind of load balancers we will address in the next segment. The procedure to associate ELB with ASG is very simple. There are two fields in auto-scaling group settings, including *Classic Load Balancer* and *Target Groups*. For a classic load balancer, all you need is to select the load balancer and associate it. On The other hand, you can associate the target group as it lies between the auto scaling group and the load balancers.

Use Case

Let us discuss the ASG use case in which you receive a burst of traffic, and auto-scaling occurs. The ASG uses a launch configuration attached to it to launch a new instance. Let us assume that a web server is running on an EC2 instance. A sudden wave of Internet traffic hits your domain and columns into Route 53, which in turn directs the traffic to the ALB that has a listener. This listener directs the traffic to the target group. The EC2 instance is associated with the target group here. In the target scaling policy, you have already mentioned that if CPU utilization goes over 75%, it will spin up a new EC2 instance. So, the excessive traffic flow causes the CPU utilization to exceed 75%. Thus, ASG uses a launch configuration and spins up a new instance. This is the functionality of the auto-scaling group.

The architectural diagram reflects the overall visibility of the working of the entire pipeline.



Launch Configuration

For *Launch Configuration*, when an ASG launches an EC2 instance, it contains the information about the configurations required for an instance launch. Launch configurations can be set with the help of an ASG, which is like launching an EC2 instance. Thus, you follow the same procedure and set the options. The difference is that in the end, you save the configurations instead of launching the instance. It is known as Launch Configuration. You cannot edit these configurations after creation. Therefore, if you want to update or replace the existing launch configurations, you need to make a new one or clone the current configuration. AWS offers a useful service to clone an existing configuration and make some changes using a button. Also, the Launch Configurations with versioning are known as *Launch Templates*. They are a new version of launch Configuration a user can use.

ELASTIC LOAD BALANCER

Introduction

With the help of *Elastic Load Balancer* (*ELB*), you can distribute incoming application traffic to different targets. *Targets* may include IP addresses, containers, Lambda functions, or EC2 instances. So ELB includes physical hardware or virtual software, accepting incoming traffic and distributing it across multiple targets. It can manage the load using specific rules, and these rules vary depending on the types of load balancers. Thus, we have three load balancers to choose from for the elastic load balancer: ALB, NLB, and CLB.

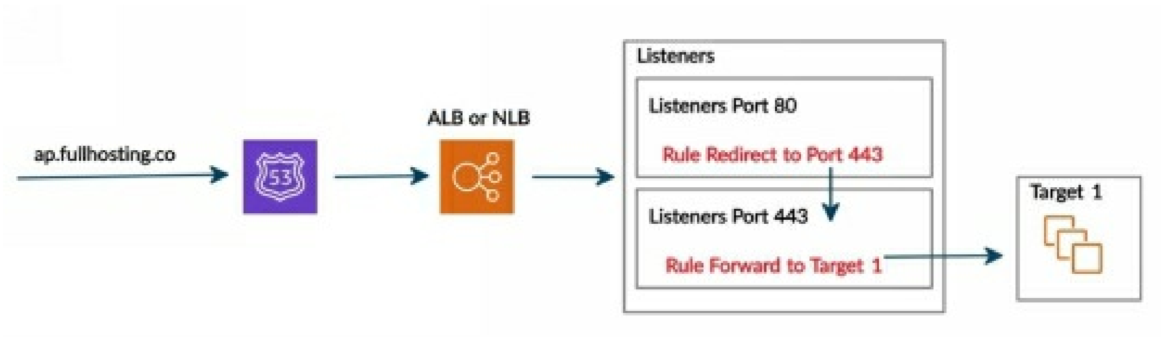
Before discussing types of ELB, you first need to understand three components for ELB traffic flow. These components include *Listeners, Rules*, and *Target Groups*. Depending on the load balancers, these components can differ. The listeners listen for incoming traffic and assess it against a particular port (port 80 or 443). Then rules decide the actions against the traffic. Lastly, there are target groups which help you gather EC2 instance (to which you want to direct the traffic to) in logical groups. So, starting from ALB, let us discuss these load balancers in detail.

Types of ELB

ALB

ALB refers to the *Application Load Balancer* used to balance *HTTP* and *HTTPS* traffic. It works at the OSI model's seventh layer, which is the application layer. You can add routing rules to the listeners by using a feature of ALB called *Request Routing*. It depends on the HTTP protocol. Therefore, Request-Routing rules are only used for ALB. You can add a *Web application firewall* to ALB as they both are application-specific. If we consider a use case for an application load balancer, then it is suitable for web applications.

Let us assume that the incoming traffic from Route 53 points to the user application load balancer. As soon as it goes there, the ALB goes to the listener that checks the port it runs on. If it is on port 80, the traffic will be redirected to port *443*. After that, it will move towards the listener that has a rule attached to it. This rule will forward the traffic to *Target* 1, which contains all the EC2 instances. The target group will distribute the traffic equally to the instances registered with it. You can also locate the position of the listeners. In this case, you have ports 80 and *443*, and this is the position where the listeners and ALB's reside. SSL certificate is also attached here because these rules appear for ALB, not for the NLB. Thus, there are a few more complex rules here. Thus, the ALB only forwards the traffic to the target.

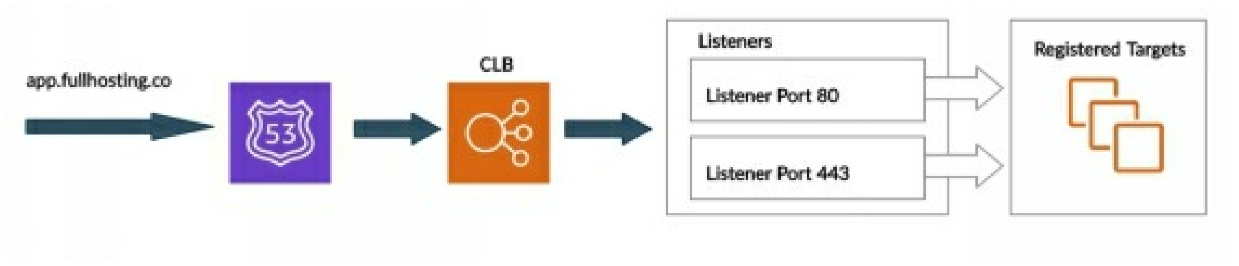


NLB

NLB is a *Network Load Balancer* used to balance *TCP/UDP* traffic. It works at the OSI model's 4th layer, which is the transport layer. It can process tens of millions of requests every second while retaining a low latency at the same time. It also performs *cross-zone load balancing*. It is suitable for the primary network performance of an application and supports multiplayer video games as well.

CLB

CLB is the *Classic Load Balancer* which is the AWS first load balancer. It can balance HTTP or TCP, but not simultaneously. It can use layer 4 to balance TCP applications and specific features of layer seven, like sticky sessions. So, when the internet traffic flows to CLB, there are listeners who listen to ports, and instead of target groups, you have registered targets. There are EC2 instances only linked with the classic load balancer. CLB can also perform cross-zone load balancing. If an application is not responding, it responds with a 504 error in the case of a timeout. Since CLB is the earliest load balancer so, it is not preferred anymore. However, network load balancers and application load balancers are frequently used.



Features of ELB

Sticky Sessions

The *Sticky Sessions* are a modified form of load balancers, which can link a user session to a particular EC2 instance. It is useful when the specific information is stored locally on a single instance, and you have to continuously send the user to the same instance. So the workflow of sticky sessions involves the following steps. In the first step, you direct the traffic to the first EC2 instance, which sets a *cookie*. So, when the person visits the next time, you check for the cookie. If the cookie exists, then you will send it to the same EC2 instance.

Currently, this feature is only compatible with CLB and ALB. Thus, if you want to set it for an ALB, you need to set it on the target groups instead of an individual EC2 instance.

X-Forwarded-For-Header

The *X-Forwarded-For-Header* is a method to identify the user's originating IP address while connecting to a web server using a load balancer or HTTP proxy. Let us consider a use case in which a person sends a request to your web application, and you need his IP address. When the request comes through, and you look for it on the EC2 instance, it turns out that it is the load balancer's IP address instead of the user's IP address. Therefore, you need *an XFF Header* to find the IP address of the user. It is a standard header to deal with load balancers. You must ensure that you use the XFF header in your web applications. You only need to read it in the web applications to get the user's IP address.

Health Checks for ELB

Let us look at the *Health Checks* for *Elastic Load Balancers*. They take the traffic away from an unhealthy instance and direct it to healthy instances. You can determine the unhealthy instance by using all available options. You need to set on a target group for ALB, whereas CLB directly sets on the load balancers itself. Then you need to ping the server at a certain URL with a specific protocol and receive a particular response. If it repeatedly happens over a specified interval, it will mark it as unhealthy. The load balancer will stop sending the traffic to it and declare it as out of service. One must understand that ELB does not terminate unhealthy instances. It only reroutes the traffic to healthy instances.

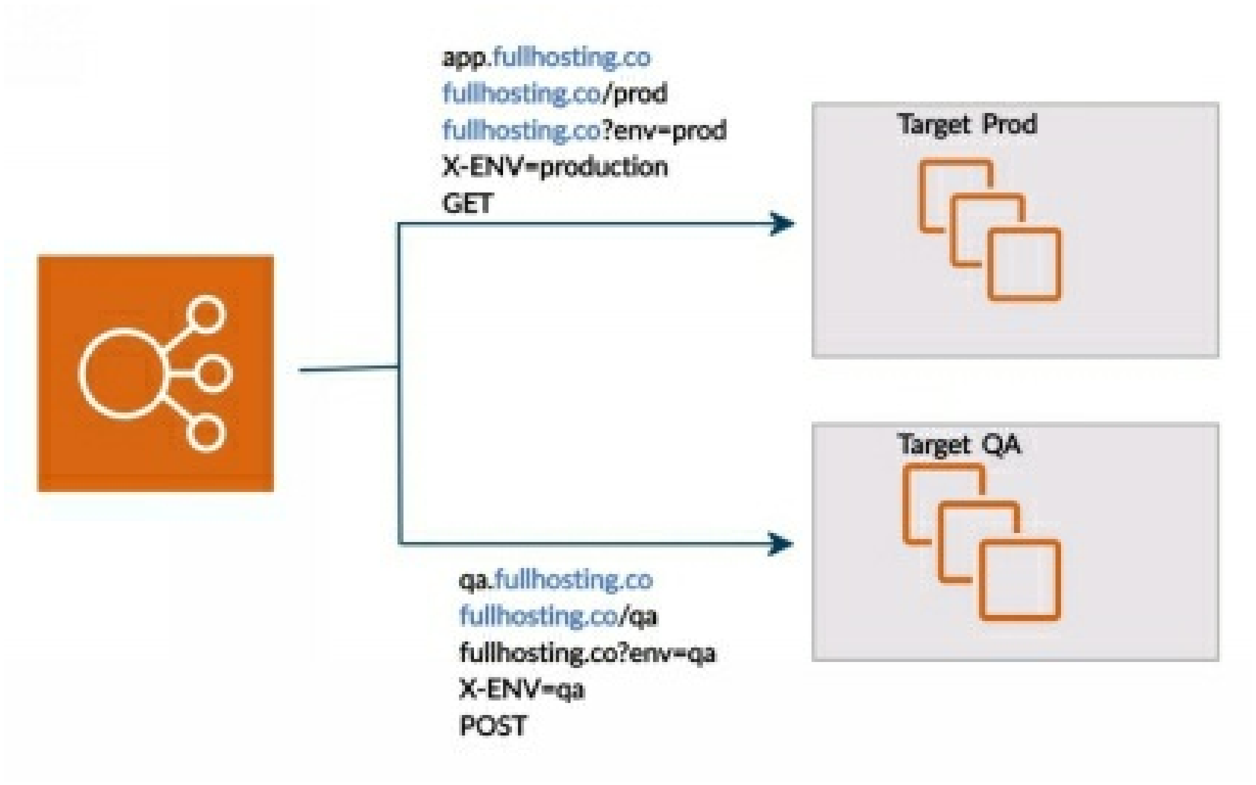
Cross-Zone Load Balancing

It is a feature that is available for Classic and Network Load Balancers. Let us assume the use cases where you can enable and disable it, and then observe the difference between them. When you enable this feature, it will distribute the requests equally across the instances and all the enabled available zones. Let us assume that there is a collection of EC2 instances in two different AZs with equal traffic distributed across them. When you disable the feature, it will distribute the requests evenly across the instances only in its availability zone. It is distributed equally in both availability zones. If you want to enable it, you can find it under the description tab. You can change the attributes and mark the checkbox on *Cross-Zone Load Balancing.*

Request Routing

Request Routing is a specified feature for ALB. It allows you to apply the rules to an incoming request and then reroute the traffic. You can check on different conditions here. There are six conditions to add: source *IP, path, host header, query-string, HTTP header*, and *HTTP header method*. After that, you need to choose the *'Then'* option, which allows you to redirect, forward, authenticate, or return to the fixed response.

Let us assume a use case in which five different examples use Request Routing. The first approach to use this feature is to direct traffic depending on the subdomain. A subdomain app can go to a *Target Prod*, and *QA* can go to *Target QA*. The other way to use it is on the path where you can have *fullhosting.co/prod* and *fullhosting.co/QA* that will route it towards respected target groups. You can do it using a query string like *fullhosting.co?env=prod* and *fullhosting.co?env=qa*. You can also do it with the HTTP header. Lastly, you can use *GET* and *POST* methods here so each GET method can go to Prod, and each POST method can go to QA. Thus, the above five use cases explain Request Routing.



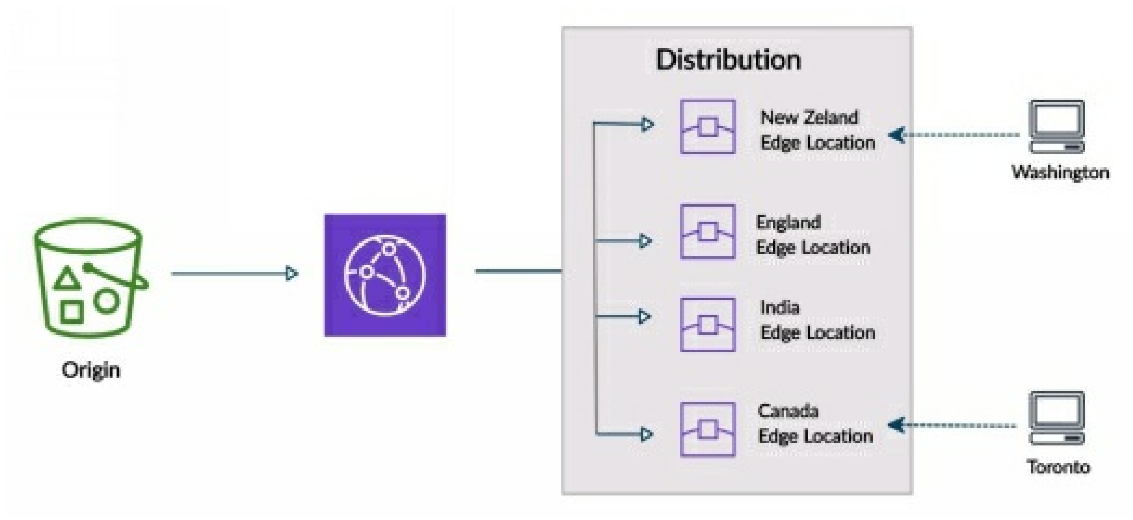
CLOUDFRONT

Introduction

CloudFront produces cached copies of a user website at different edge locations around the world. It is *a content distribution network (CDN)*. One must be aware of a content delivery network to understand the concept of CloudFront. CDN is a massive distribution of caching servers around the world. It contains the content stored in the origin servers and directs the viewers to the most suitable location to see the content stored in the cache. The nature of the content can be *Static* (does not change) or *Dynamic* (does change). CDN improves the scalability and performance of applications.

CloudFront is Amazon's worldwide content delivery network with massive capacity and scale. You can also use built-in security features for the best services. A user is in control of the service and can make changes during the work. It includes real-time reporting to monitor the performance and modify the application or the way CDN interfaces with the application. Moreover, it has been designed for static and dynamic objects and video delivery.

Let us explain the concept of CloudFront by using a graphical representation of a CDN. You host your content on the internet, with S3 being the origin. The CloudFront server distributes a copy of the website on several edge locations. These edge locations are the servers that are closer to users worldwide. When the user accesses content from Toronto, it will not go to the S3 bucket; rather, it will send the traffic to the CloudFront. Then, the Cloud front will send it to the nearby edge locations to ensure the lowest latency for the user.



Core Components

There are three main components for CloudFront, i.e., *Origin*, *Distribution*, and *Edge Locations*. An origin is a place where the original files are located. The distribution is a set of edge locations that determines the actions of cache content. The definition says that it fetches the data from the origin and upgrades the cache at any frequency or uses HTTPS. Then you have the Edge Location, a server that is close to the actual user that stores the content cache. Let us discuss all the components of the CloudFront now.

Origin

Origins either describe the Amazon S3 bucket, HTTP server, or an EC2 instance from where CloudFront fetches the content. A user must create at least one origin. There are a few factors that you can use to maintain security and ensure that origin is delivering content to the CloudFront. One feature is *CloudFront OAI*, an Origin Access Identity that limits access to the S3 bucket content using CloudFront. The requests get denied if they are not coming from CloudFront. CloudFront can use a *Signed URL* to access the origin, and the origin only responds if the content of a signed URL is valid. You can also use *Origin Custom Header* that returns from CloudFront to the origin. The origin will search for the header depending on its value, and the request would be denied if it does not exist or the value is incorrect.

Distributions

Let us explain the *Distribution* component of CloudFront in more detail. It is a set of edge locations that act as an indicator of the original content you are hosting in AWS or custom origin. You must specify the origins in the distribution so that CloudFront can determine the location of the content on the user's request when the content is not stored in the cache.

Specifying the origin is the first thing in setting up distributions. While setting up distribution, a Price class determines the cost and the amount of replication of copies. Selecting all edge locations results in better performance as the website would be accessible worldwide. There are two types of distributions, i.e., *Web* and *RTMP*. Web distribution is used for the websites, whereas RTMP is used for streaming media. You can serve up streaming video under the web as well; however, RTMP is a precise protocol.

Edge locations

An *Edge location* is a small setup in different areas providing low latency connections by delivering static data contents from the request's closest location. It just travels to the nearest edge location and transfers the data rather than receiving its source data. Edge locations handle CloudFront and Route 53 requests. As a result, requests for each of these services are automatically routed to the nearest edge locations. It reduces access time and provides a faster response.

Features of CloudFront

Let us look at the important feature of CloudFront, starting from behaviors.

Behaviors

The ability to perform all the configurations is known as *Behavior*. It allows you to implement policies, change, or alter the type of content you are delivering. It depends on who is requesting it or the duration for which an object stays in the cache. CloudFront distributions have one-to-many behaviors with one default Behavior. You can have different behaviors in order of precedence, and the default behavior will take place if all the conditions fail to meet the order of precedence.

There are many options while setting up behaviors. You can change the direction of traffic to HTTPS and limit specific *HTTP methods* or *viewer access***.** Besides, you can set *TTLs (time to live)* as well. In TTL, the content expires every two minutes and needs a refresh, depending on your desired content. There is another concept in CloudFront, known as *Invalidations*. It configures the TTL (Time-to-live) manually so that you do not need to wait until the TTL expires. So, it is useful while making any changes to the S3 bucket. As you are going to create invalidation manually, therefore those changes will appear at once.

Restrictions, Errors, and Tags

CloudFront enables you to restrict access depending on the requester's geographical location at no additional cost. You can either blacklist or whitelist a location based on security reasons or the fact that you can only distribute the content to a few regions. Furthermore, if an error occurs, it allows you to return to an error page. You can cache the error page for a specific time and set a response code that needs to be sent.

AWS WAF

Web Application Firewall is abbreviated as *WAF*. This feature helps in defending the web applications against attacks that compromise their availability, security, or resource consumption. WAF is considered as a front-line defense for AWS ALB, API Gateway, and CloudFront. It can define customizable security rules that determine to allow or reject any traffic. It is the seventh layer of application protection created in each edge location of the CloudFront worldwide.

First of all, CloudFront will receive a request from the end-user and then forward it to WAF for examination, and then it will decide whether to allow or block that request. The websites hosted on platforms other than AWS can be protected by integrating WAF with the CloudFront. The rules created in AWS WAF include applying restrictions on particular IP addresses, URI strings, and HTTP headers. It can be used with:

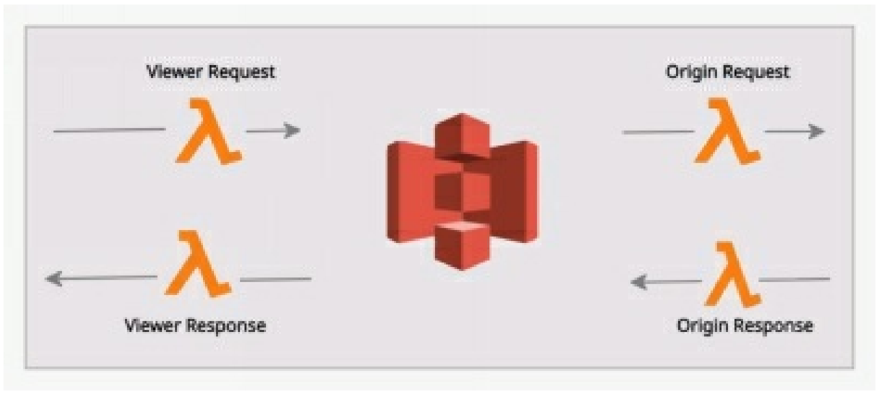
- CloudFront, as part of a content delivery network,
- API Gateway for APIs,
- ALB as a front end for web servers/origin servers operating on EC2

The users will be charged on the basis of the ACLS used, the number of rules in a single ACL, and the number of web requests received.

Lambda@Edge

Another exciting feature in CloudFront is *Lambda@Edge*. There are lambda functions that override the Behavior of requests and responses that flow to and from a CloudFront. There are four Lambda@Edge functions: the *origin request, viewer request, the viewer response,* and the *origin response***.** Let us describe these functions individually:

- *Viewer request* means that when a request from a viewer is received by the CloudFront.
- The *Origin request* means that before a request is forwarded to the origin by the CloudFront.
- *Origin response* means that when the response from the origin is received by the CloudFront.
- *Viewer response* means before the response to the viewer is returned by the CloudFront.



Let us assume a common use case of Lambda@Edge in which there is protected content, and you need to authenticate it against a Cognito. Only those users that reside in our Cognito authentication system can access the content. It is one way to protect content. However, there are many other creative solutions where you can use Lambda@Edge, such as you can use it to serve up *A to B testing* websites. When the viewer's request arrives, you have a roll of the die, and it will change what it serves back. Therefore, it could set up A or B. Hence, Lambda@Edge provides many opportunities.

CloudFront Protection

Let us discuss *CloudFront Protection*. Suppose CloudFront serves up your static website and you have protected content like video content that you do not want others to access easily. In that case, there is an option to restrict access to the viewers while setting up the CloudFront distribution. *Signed URLs/Cookies* can be used to view any content. When you apply this check, it creates *Origin Identity Access (OAI)*. OAI is a virtual user identity that allows the CloudFront distribution to fetch private objects. It automatically configures so that you can use Signed URLs or Signed Cookies.

A *signed URL* is a URL that CloudFront provides to access the private cached objects temporarily. Then, there are S3 features known as *Pre-signed URLs*. One must understand that the Pre- signed URLs belong to S3, and Signed URLs belong to CloudFront. Then there are signed cookies, which are like the signed URLs. However, the difference is that you deliver a cookie with the request to enable the users to access several files. It eliminates the need to generate a signed cookie repeatedly. Thus, you can access as many files as you want till the cookie is valid and it passes along. It is helpful for video streaming. You cannot do protected video streaming with signed URLs because when all the video streams get delivered in chunks, you need to set a cookie. So, these are the options to protect content with the help of CloudFront.

Use Cases

As we have discussed AWS CloudFront's features, let us now extend the dimensions of its applications.

Static Quality Caching

The delivery of static content (such as JavaScript, images, vogue sheets) is accelerated for the global audience by Amazon CloudFront. By default, it will provide a multi-level cache and regional edge caching to enhance latency and minimize the burden on the origin servers when the item is not cached at the sting.

Live & On-Demand Video Streaming

You have got a number of options to stream the media with CloudFront. It can deliver 4k media for the viewers. The file will view live and recorded events.

Security and DDoS Protection

CloudFront can be integrated smoothly with AWS WAF to secure layer seven and AWS defend to mitigate DDoS attacks at Layer 3/4. AWS CloudFront will also provide authentication to the viewers with signed URLs and negotiate TLS connections with the most suitable security ciphers.

Dynamic and Customized Content

CloudFront provides intrinsic network optimizations to assist in improving the responsibility, performance, and global access of the dynamic components of an application and the delivery of customized content to all viewers.

API Acceleration

CloudFront is mostly used to accelerate and secure API calls. By default, it is integrated with the Amazon API entranceway and supports proxy ways such as *PUT*, *DELETE*, *POST*, *PATCH*, and *OPTIONS*.

Software Distribution

When worldwide spread users transfer software updates, CloudFront will scale them automatically. With the help of CDN, any software can be created right on the spot wherever a user is located.

CHAPTER 7: AWS SECURITY

Overview

We will discuss *AWS security* in this chapter as Cloud Security is the highest priority in Amazon Web Services. The AWS data is as secure as in traditional data centers. Security is the key concern for most cloud technology adoptions. However, many people who implemented cloud technology in the early days claim that the cloud solution has better security than on-premises security. AWS security works on *a Shared Security Responsibility model,* meaning that Amazon will secure its infrastructure. At the same time, you can have your security controls for the data and the applications you launch and store in the cloud. Not only the security level of AWS is higher than on-premises, but the cost of using this type of security for your applications is also very low as compared to on-premises.

Amazon has different security tools to help in implementing the best AWS security practices. AWS commonly uses some of these services, such as IAM, KMS, and Amazon Cognito. Let us start this chapter with IAM as it is the most important security feature in AWS.

AWS IAM

Introduction

IAM stands for *Identity and Access Management.* It is a web server that manages access to the AWS resources in a safe and secure way. With IAM, you can safely manage contact to AWS resources. You can also create different groups and let them access specific servers or limit them from using certain services. All of this is possible with identity and access management, and it comes without any extra charges. It allows you to authenticate or limit a certain set of users to access the AWS account or specific resources in the AWS account. Let us assume a use case in which there are three groups, including group1, group2, and group3, and the IAM administrator wants to allow permissions to a different group of users. Let us assume the administrator wants to restrict group2 to access AWS resources. The IAM authorizes the administrator to allow access for a certain group to certain resources. In this case, the IAM admin can grant access to group1 and group 3 and restrict access to group2. So, that is the function of IAM.

How does it work?

Let us discuss a few elements that complete *IAM workflow*. There are six categories: Actions, Principle, Authentication, Resources, Request, and Authorization. Let us explain these core elements of IAM workflow.

Principle

An application or a person that can request operations (actions) on AWS resources is known as a *Principle*. An action on an AWS resource can be performed only by a principle as it is the main element of IAM. It is an entity that takes actions on AWS resources, which may be a role, a user, or an application that tries to access the AWS environment.

Authentication

It is the process of identity verification of the principle that wants to enter into the AWS network. It is the second element of IAM. You will be considered authentic if you enter the credentials or if you provide the required keys to verify that you are the same person as in records. To get Root User authorization, you have to use your email and password to log in from the console. As an IAM user, your account ID must b entered first, then your username and password. You must provide the two secret keys (access and secret access keys) for AWS CLI/API authentication.

Request

The *request* is another element that makes the IAM. If a principal wants to use the AWS console, a request is sent from the user (principal) to the AWS. The request carries information about the actions of the principal. A request can be of three types such as *Put*, *Get*, or *Delete* request. This data includes resource and request origins on which the action needs to be done, such as EC2 or S3 bucket. The origin may be an AWS environment or any other Cloud Service provider. So, this data will be available in the request. AWS collects the request data into a request context, which evaluates and authorizes the request.

Authorization

In *Authorization*, IAM uses data from the Request Context to find similar policies and determines whether to accept a request or not. By default, all resources are denied according to the evaluation logic, so a request is only authorized by IAM if the matching policy allows every part of the request. In any permission policy, an Explicit allow overrides this default behavior. A permission policy may be identity-based or resource-based. If it is not the explicit allow, then it overrides the default denied**.**

On the other hand, if you deny service, it will overwrite all the allow statements and other statements. As a result, explicit deny will take place. By default, only the root users in the AWS account can access all the resources. You must be authorized by the policy if you do not log in as a root user.

Actions

Let us briefly discuss Actions now. After authentication and authorization of the requests, AWS approves the actions that you are going to take. With Actions, you can view, create, edit the content, and delete a resource. It is all based on the action that you are allowed to do. For instance, IAM supports about forty actions for a user resource, including:

- UpdateUser
- GetUser
- DeleteUser
- CreateUser

Resource

An object that exists in service is called a *Resource*. So, you can perform a set of actions on related resources. Anything is considered incomplete if it does not get mentioned in the action or resource. For instance, requesting access to the instance for an IAM role after removing it will result in a denied request.

IAM Components

Let us discuss a few more IAM components, which include Roles, End Users, Groups, and Policies. The End Users are the basic building blocks. When different end-users combine, they form a Group. Policies are the engines that allow or deny a connection. It means that access to a resource depends on the policy. Lastly, the Roles are temporary credentials that you can expect for an instance when they are needed. Let us discuss them one by one.

End-User

In an AWS environment, an *IAM End-User* refers to an entity that a user can create. It indicates a person or a service that communicates with the AWS environment. IAM User has a few credentials with it. By default, it does not have any secret key or a password.

The root user creates a new user in the AWS account and assigns the desired credentials. These credentials get linked with the end-user. By default, a user cannot perform any operation in AWS. The benefit of having a one-to-one user is that you can assign permissions to the users individually. For instance, you might want to give administrative permission to some users to manage your AWS environment.

A user is an entity in the AWS account, so it is not bound to pay separately. It can start using the servers, and that will build under the account it is associated with. A separate user does not indicate separate payments for each user. They are just users of that account, and the entire account pays for the services it provides. The IAM user does not always represent a person. It is an entity that has credentials and related permissions. Therefore, access to AWS resources can be securely managed by IAM.

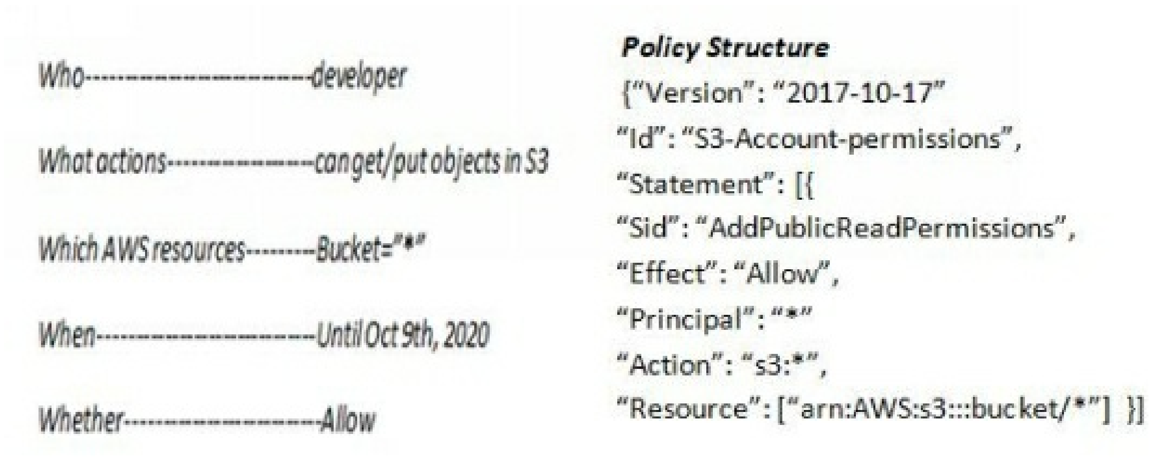
Group

The collection of IAM users is known as the *IAM Group*. It specifies permissions for several users. It ensures that any permission granted to a group is also applicable to the user associated with that group. A group may have multiple users. However, a group cannot be nested. It means that a group can not belong to another group. Instead, only users can belong to a group or several groups. A group needs to be created because there is no such thing as a default group that automatically includes all users in a group.

Let us consider a use case where you want to create groups for a small company that is expanding every day. So, you create an admin group to create and manage all the users. This admin group further creates two groups, which are a developer group and a tester group. Every group consists of different types of users, and every user has its credentials. Managing a group is easy with AWS IAM. You need to set permission to a group, and it gets applied to all the users in that group. Suppose a user joins the group after you applied the policy. In that case, the new user will automatically acquire all the policies and permissions specified in that group. So, the IAM can handle the administrative burden for handling the user permissions and privileges.

Policy

Our next component is the *policy*, which sets permissions and manages access to the AWS resources. These policies are clear documents that are stored in JSON format. Policies define user's access to a resource and the action they can perform. In AWS, the policy is the entity that defines the permission to take action if it is attached to a resource, and then AWS examines these policies. When the principal makes a request, the policy will define whether to allow or deny a user access to the service. Let us consider a use case where you have to allow developer access to your S3 buckets. So, a policy will contain a few statements and structures as:



Who-----------------------developer

What actions----------------can get/put objects in S3

Which AWS resources---------Bucket="*"

When-----------------------Until Oct 9th, 2020

Whether-------------------Allow

Policy Structure

```
{"Version": "2017-10-17"
"Id": "S3-Account-permissions",
"Statement": [{
"Sid": "AddPublicReadPermissions",
"Effect": "Allow",
"Principal": "*"
"Action": "s3:*",
"Resource": ["arn:AWS:s3:::bucket/*"]  }]
```

The policy tries to answer some questions. The developer wants to access a service to download or upload an object in S3. In that case, he needs to *Get Object* or *Put Object* in S3 using the JSON language. The * symbol indicates all buckets in S3. Then the developer wants to add these additional statements to be in a policy till *9th OCT 2020*. You have allowed the permission so the developer can get and put the objects in an S3 bucket till *9th OCT 2020*.

Types of Policies

The *Online Policies* and *Managed Policies* are the two main policy groups**.** The Managed policies are the default policies that you can link with multiple entries in the AWS account, such as users, roles, and groups. It is a single identity-based policy that is divided into two subcategories. The first one is *AWS Management Policy* created by AWS, and the other is the *Customer Managed Policy*. The IAM policy can be created and edited in the visual editor. The JSON policy document can also be created and uploaded to the console to start using them. On the other hand, online policies are incorporated into a single user, a role, or a group directly. You create and manage online policies.

Role

A Role is a permission set that specifies the permitted or refused actions by an entity in the AWS console. It is quite similar to the End-user. The only dissimilarity between role and user is that the user permissions are preprogrammed and cannot be altered. In contrast, you can access the role permissions by any entity, whether it is a user or an AWS service. In detail, a role is an entity with an attached permission policy that defines the actions of identity in an AWS account. A user is individually associated with one person, whereas anyone can assume a role. Instead of long-term credentials, including passwords or related access keys, a role has temporary credentials. So, when you assume a role and want to access any service, you will get these temporary credentials created dynamically. Furthermore, user permissions are permanent, and role permissions are temporary.

An application or a service that normally does not have access to AWS resources can assume roles. Roles are essential components of IAM for such situations. Consider a use case in which you want to allow users permission to access the AWS resources in an account or share the resources between two AWS accounts. A mobile application can also be allowed to use AWS resources. However, you do not want to save the key or credentials in the mobile application. Thus, you can use roles to allow temporary access to your resource. There are some situations where you need to grant access to the resources with defined identities outside of AWS. For instance, if someone is already a *Google* or *Facebook* authenticate, you can use roles to let them access some of your AWS account resources. Sometimes you may need to grant access to a third party such as a consultant or an auditor to your account so that they can have access to your account to audit your AWS resources for a limited time. They only want access until their audit is over, so they are not permanent users.

Use Case for Role

Let us see how roles grant permissions to other services. Consider a use case scenario where the EC2 instance needs temporary access to data in the S3 bucket. First, you need to create a role or permission by using policies because roles use policies. Policies are the work engines for every permission-related operation. For allowing bucket access, a role has to be created with a suitable policy; after that, a role-based instance can be launched. The roles can be assumed by any entity, which may be a user, an application, a service, or any other AWS resource. It is another resource of the AWS account in this case. For an EC2 instance to assume a role, the role must be created and linked with an EC2 instance first.

The same role can be attached to another EC2 instance, user, or database service. Anyone can assume roles. The EC2 instance contacts the role whenever it needs to interact with S3. Then the role provides this EC2 instance with temporary credentials that will expire after an hour. Then, the EC2 instance can access the S3 bucket with these temporary credentials and access the files. The EC2 instance will contact the role if it needs to interact with S3 in the future, which will assign a new credential for a limited period. With these new credentials, the EC2 instance can interact with the S3 bucket again and get the data. Therefore, the password is not permanent. It keeps changing. It is a secure token that EC2 instance or any other application that uses the role can access those resources on the other end.

Features of IAM

The main features of IAM are:

- *AWS account shared access:* The key feature of IAM is that it allows you to create unique usernames and passwords for individual users or resources and representative access.
- *Free of cost:* If you want to create additional users, groups, and policies, they are available for free. There are no extra charges for IAM security.
- *Multifactor authentication (MFA):* IAM supports MFA, where users can enter their username and password. Then they enter a one-time password from their devices - an additional authentication factor that uses randomly generated numbers.
- *Identity Federation:* You can make IAM trust Facebook or Google Account authentication method and allow access on its basis. It also enables its user to keep the same password for the cloud environment and on-premises.
- *Granular Allowance*: You can apply restrictions on requests. For instance, you can allow users to download data but prevent them from updating data through policies.
- *Password Policy*: You can reset or rotate a password locally with the IAM password policy. You can set rules for passwords as

well. For instance, you can set a rule such as picking a password or the number of attempts to enter a password before denied access.

- *PCI DSS Compliance: IAM complies with the PCI DSS*, which is the data safety standard for the companies to manage branded credit cards from major card schemes.

AWS KMS

Introduction

With *Key Management Service (KMS*), you can create and control cryptographic keys and monitor their usage across different platforms and applications. With KMS, you can securely manage keys across AWS services and hosted applications. It uses the method of *Envelop Encryption* and maintains keys in *Hardware Security Modules* (HSMs). Envelop encryption means the data is encrypted and stored locally in the AWS service or application along with the key. With KMS, the administrators can create, control, and delete the keys to encrypt the stored data in the products and databases of AWS. It provides centralized control over encryption keys to define the user data. You can import, create, rotate, outline, delete, and deactivate the usage policies for secret writing keys and inquire about using these keys to encode the user data.

KMS is a multi-tenant *Hardware Security Module (HSM)*. It is part of hardware at the AWS data center specifically designed to store keys within the memory. So, it is very secure and cannot be written to the disk. KMS is said to be multi-tenant because some other users might be using the same part of the hardware and are virtually separated from each other via software. Lastly, KMS uses envelop encryption which uses CMKs and data keys so that when you encrypt the data, it gets protected. It uses CMKs to encrypt the data keys and data keys to encrypt the plain text. However, you need to protect the encryption key. There is a master key for encrypting the data key, which acts as an additional layer of security, and that is why it is known as envelop encryption, as it is just like keeping your key in the envelope where no one can reach it. AWS KMS is reachable within AWS Identity and Access Management by selecting the section '*Encryption Keys*' or using the software.

CMK

CMK is a Customer Master Key which refers to the logical depiction of a *Master Key*. You can use it to encrypt and decrypt a *Secondary Key* (A secondary key is also known as a data key). Only CMK can decrypt the data key when requested by an application or a service. The data key is stored at the service end. Furthermore, there are two types of CMKs. One gets created automatically by AWS when the first encrypted resource is generated whereas, the users create the second. Users can only monitor the usage of the keys while the KMS maintains the lifecycle and permissions of the keys from the previous category.

How does it work?

Mostly the AWS KMS gets integrated with the Cloud Trail to provide the encryption or other services with the help of key usage logs for the provision of regulatory services, auditing, and compliance needs. You can securely manage and store your keys with AWS KMS. The government-approved-HSMs will generate and protect these keys, allowing you to use them in plaintext format in the modules. With master keys, you can directly submit your data to encrypt or decrypt KMS. Furthermore, you can set specific usage policies on them to define the users for data encryption and decryption.

Functions

AWS KMS includes both *Management functions* and *Cryptographic functions*.

KMS Management Functions

- With AWS KMS, you can define, create, and provide a list of master keys.
- The master key can be enabled or disabled.
- There is a possibility that you can build and access different control policies and also display grants for your master key.
- The automatic rotation of cryptographic material can easily be enabled or disabled in a Master key.

- It is possible to import cryptographic material into the AWS KMS master keys.
- You can also tag the respective master keys to identify, monitor easily, and categorize them.
- You can make, delete, update, or even list all aliases, most of which are associated with the respective master key.
- You can remove the master keys to get notified of the complete key lifecycle.

KMS Cryptographic Features

- You can easily encrypt, re-encrypt, or decrypt the data in any way possible.
- You may also export different services to develop data encryption keys. So, in the presence of a master key, these services are in plain text or encrypted form.
- You can generate random numbers for various cryptographic applications.

AMAZON COGNITO

Introduction

Generally, web applications permit the successful application sign-in using valid user credentials. The user credentials consist of a username and a strong password. The new authentication process adds even more ways to ensure user authentication. When using AWS, you can achieve it through the functionality and capabilities of *AWS Cognito*. It is a decentralized way to manage authentication. One may consider it as Log-on/Log-in integration with the user applications and other social identity providers.

How does it work?

Amazon Cognito gathers users' profile attributes into directories known as *User Pools*, which are then used by a Smartphone app or a web app to set up restricted access to AWS services. End-user data is clustered in an *Identity Pool* so that the operating systems, devices, and client access platforms receive it to manage federated identity groups. When the device is connected to the internet, the data integrates with AWS, allowing you to view it from another device. In case you are offline, you can save the data locally on the *SQLite* database before reconnecting. In the sync store, Cognito combines sets of data with identities and stores encrypted data as a value or key pair. You can store up to 20MB of data with a limit of 1MB of individual data set. A developer can set Amazon Cognito to get a stream of events when he updates and synchronizes the data.

Web Identity Federation

Understanding the terms *Identity Providers* and *Web Identity Federation* will help you fully understand Amazon Cognito. In the Web Identity Federation, you exchange the identity and security information between the application and identity provider. On the other hand, the Identity Provider is a reliable provider for your user identity, which allows valid access to other services. *Facebook*, *Digits, Google*, *Amazon*, and *Twitter* are considered public identity providers for Amazon Cognito. The identity provider verifies identity for the end-user and passes an *Open-ID Connect token or OAuth* to Amazon Cognito. The user who receives short-term AWS credentials with limited access will have a new Cognito ID.

Key Components

Let us discuss the two key components of Amazon Cognito in detail. The Cognito user pool is the directory of users to verify the identity providers. The Cognito identity pool offers temporary credentials for the users to access AWS services. Another component called Cognito Sync synchronizes user data and preferences on all devices. It is so versatile that it lets you use identity pools and user pools separately. You can also use them together. Let us discuss these components in detail now.

User Pools

Cognito User Pools are the most common use case for Cognito. It is a decentralized directory of the users to handle different actions, such as account recovery, sign-up, sign-in, and account confirmation. *Account Recovery* is just like resetting a password. In contrast, *Account Confirmation* is similar to verifying an email address after sign-up. If a new user sign-up with your mobile or web application, it will update the user pool and authenticate the credentials against the user pool's data when the user sign-in.

The user pool can connect with identity providers so that it can have its email and password form to use it. It generates a *JSON Web Token (JWT)* to preserve the connection after its authentication. If you want multi-factor authentication, the user pool can turn on MFA, which means it is very easy to integrate. You can also integrate Cognito with Pinpoint for the user campaigns, such as sending out a campaign's field. Besides, you can override various functionalities with Lambda. Thus, you can trigger Lambda at any time to perform different functions such as a sign-up, sign-in, or password recovery.

Identity Pools

Let us discuss *Identity Pools* now. It provides temporary AWS credentials to the users. These temporary credentials allow them to access other AWS resources without re-entering their credentials like Dynamo DB or S3. You may consider it as an actual mechanism that provides access to the AWS resources. Using the Identity Pool, you will select who can generate those credentials *an*d then use the SDK to create those credentials. After that, the application can access those AWS services.

You will select a provider first, which can be an authenticated or unauthenticated identity. You can choose Cognito from any authenticated providers such as Facebook, Google, or several other providers. And then, once you have created the identity pool, they have got a simple way for you to use the SDK. You can drop down your platform, have the code, and then you are ready to get your temporary credentials.

Cognito Sync

Cognito Sync allows the synchronization of user data and preferences across all devices only with a single line of code. It uses push notifications to push those updates and synchronized data. In the backend, it uses a Simple Notification Service to push this data to the devices. The data is the user data, and the preferences (key-value data) are stored with the Identity-pool.

CHAPTER 8: APPLICATION INTEGRATION SERVICES

Overview

With *message queuing service* and *automating tasks* together in the AWS application integration, you can create reliable, asynchronous, and scalable applications. You may need some additional features while building an IoT, web, or mobile application on the cloud, as they play an important role in upgrading your application's functionality and efficiency. For instance, you can use SNS to create push notifications for your web or mobile users. You can add such types of features with AWS App integration. This chapter focuses on learning Amazon's Message Queuing Services, which are considered important from an exam perspective. These services include SQS and SNS. Let us discuss them in detail.

SQS

Introduction

SQS is an abbreviation of *Simple Queue Service,* which is a completely managed queuing service for scaling and setting apart the architectures of microservices, serverless and distributed systems. Let us have a comparison between a queuing system and a streaming system, which will help you understand the SQS. So, the *Queuing System* is a messaging system that offers asynchronous communication and sets apart processes via messages. These messages are known as events from the sender and receiver point of view.

On the other hand, the sender and receiver in the streaming system are known as producers and consumers. In queuing systems, the incoming messages are generally deleted when they get consumed, which is a simple communication process. It is not good for real-time scenarios because it is reactive. It means that the sender and receiver have to pull to decide the action so that they can communicate with the queue and the messages. There are a few examples of queuing systems such as *Sidekiq, SQS,* and *RabbitMQ,* which is debatable because one may consider it as a streaming service.

In contrast, a *Streaming System* can respond to multiple consumers' events. If multiple users want to perform any event, they can do it since it will not be deleted instantly. The messages stay in the event stream for a longer time, enabling you to apply complicated operations. A big difference between both queuing systems is that one is reactive while the other is non-reactive. One allows you to perform multiple actions on messages and keeps them in the queue, whereas the other remove the messages and does not care about its operations. So, it was a comparison between streaming and queuing systems. Let us continue with SQS, which is a queuing system.

SQS is an *application integration service* that acts as a bridge of communication and connects isolated applications with the help of messaging and queues. It uses a queue, which is temporary storage for messages that need to be processed. You can understand it as going to a bank, and everyone is waiting in line, that is the queue. SQS is a pull-based service instead of push-based. You need the AWS SDK to interact with the queue and write the code that will publish messages to the queue. Then you will need AWS SDK to pull the messages to read them. Let us discuss the uses of SQS.

What is SQS used for?

In cloud applications, the most common ways of using SQS and other messaging systems are as follows:

Decouple Microservices: Messages are one of the best ways to establish contact between various components of the system in the microservice architecture. If your microservices (particularly serverless services) run in AWS, then SQS is a great option for that kind of communication.

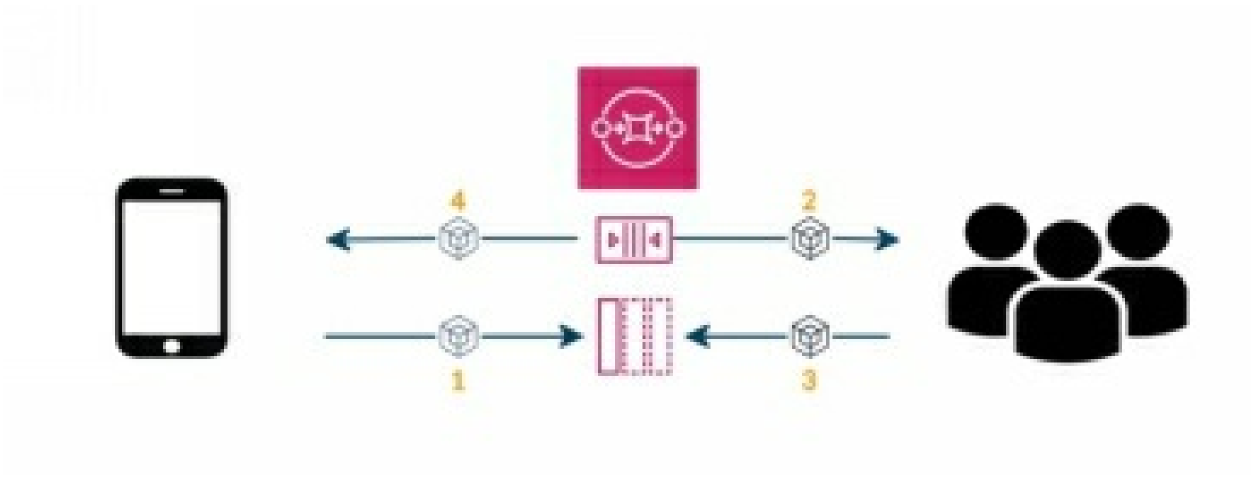
Send tasks between different parts of your system: To take advantage of SQS, you do not have to run a microservices-based application. You can also use it to communicate the tasks of one application to another system/application.

Distribute workloads from a central node to worker nodes: In wide distributed workflows like map-reduction operations, you will also find messaging systems. It is important for such operations that all tasks which need to be processed remain in the queue, distribute them quickly between the machines or functions which perform the work, and make sure that each part of the work is done only once.

Schedule Batch Jobs: SQS is an excellent choice to schedule batch jobs due to two reasons. First, it keeps a long-lasting queue for all the scheduled tasks, which means you do not have to monitor the job status. You can entirely depend on SQS to get the jobs done and to manage any retrieval even if the operation fails and the batch system return messages to the queue. Secondly, it merges with AWS Lambda to perform batch operations; if you use AWS Lambda with SQS, it automatically runs the Lambda functions for data processing once the data is available to them.

Use Case

Let us assume a use case for SQS. Consider a mobile app and a web app, and you want them to communicate with each other. The mobile app will send a message to the queue by using the AWS SDK. Now the web app will use AWS SDK and pull the queue whenever it wants. It is, therefore, up to this web application to code in how often it will check for a specific message in the queue. If a message appears, it will pull the message down, use it, and report the queue that it has consumed the message. It will inform the queue to proceed and remove this message from the queue.



Now, the mobile app wants to know if the web app has consumed the message. It will check from time to time that if the message is still in the queue or not. If it is not in the queue, the mobile app will know that the other app has consumed the message. Therefore, it is the process of using SQS between two applications.

Limits & Retention

Let us discuss some SQS limitations. Message size in SQS ranges from 1byte to 256KB. If you need to go farther than the limit, you can use *Amazon SQS extended client library for Java* which allows you to extend the message size up to 2GB. It works in a way that the message will be stored in S3, and the library will refer to that S3 object.

Message Retention is the duration in which SQS keeps a message in the queue before dropping it. It is set to four days by default. This message retention can be set from 60 seconds to 14 days maximum.

Queue Types

The Standard Queue and *FIFO* are the two different queue types. *Standard Queue* provides a nearly unlimited number of transactions per second while transacting. It is similar to messages and guarantees to deliver the message at least once. You can potentially deliver multiple copies of a message, but it can cause things to go wrong. Standard queue tries best to keep the messages in the order they are sent, but there is no surety of it. On the other hand, you need to use *FIFO* (First In, First Out) if you want the guarantee that the messages stay in their order. The message is sent exactly once in the FIFO Queue and remains visible until the consumer deletes it without introducing duplicates into the queue. It provides one-time processing where the order of sending and receiving a message is well-preserved. However, it has a limit of 300 transactions per second for a single API action.

Visibility Timeout

Visibility Timeout is the duration defined for a queue object that is hidden from the queue and other users once it is fetched and processed by a consumer. The main goal is to avoid multiple consumers (or the same consumer) who repeatedly consume the same object. With this feature's help, you can prevent an application from reading a message when some other application is already reading it, or you can avoid doing the same amount of work that someone else has already performed.

The Visibility Timeout is the duration in which the messages are not visible in the SQS queue. When a reader selects a message, you can set a visibility timeout between 0 to 12 hours for that message so that no one else can access or see it. Visibility timeout is set to 30 seconds by default. When the reader completes the processing successfully, it can delete the message from the queue; otherwise, the message would appear again in the queue for another customer to select it again. If the consumer fails to complete it in the given visibility timeout frame, he must kill the job; otherwise, it might end up with the same messages delivered twice.

Polling Types

Polling is the way of retrieving messages from the queue. *Short Polling* and *Long Polling* are the two ways of polling in SQS. The short polling is used by SQS by default. It returns the messages immediately, even if the message queue is empty. When there is no message to pull, the short polling will not come in handy because you are simply making calls without any reason. However, there are some scenarios where you need to retrieve the message immediately, and short polling is useful in such situations. In most use cases, the long polling should be used because it waits for the message to appear in the queue or till the timeout expires. As soon as the messages are available, you can retrieve them from the queue with the help of long polling. It is not set by default, so you have to set it manually. It also reduces the price because you can limit the number of empty receives. You can enable long polling within SDK and schedule the received message requests with a waiting time.

SNS

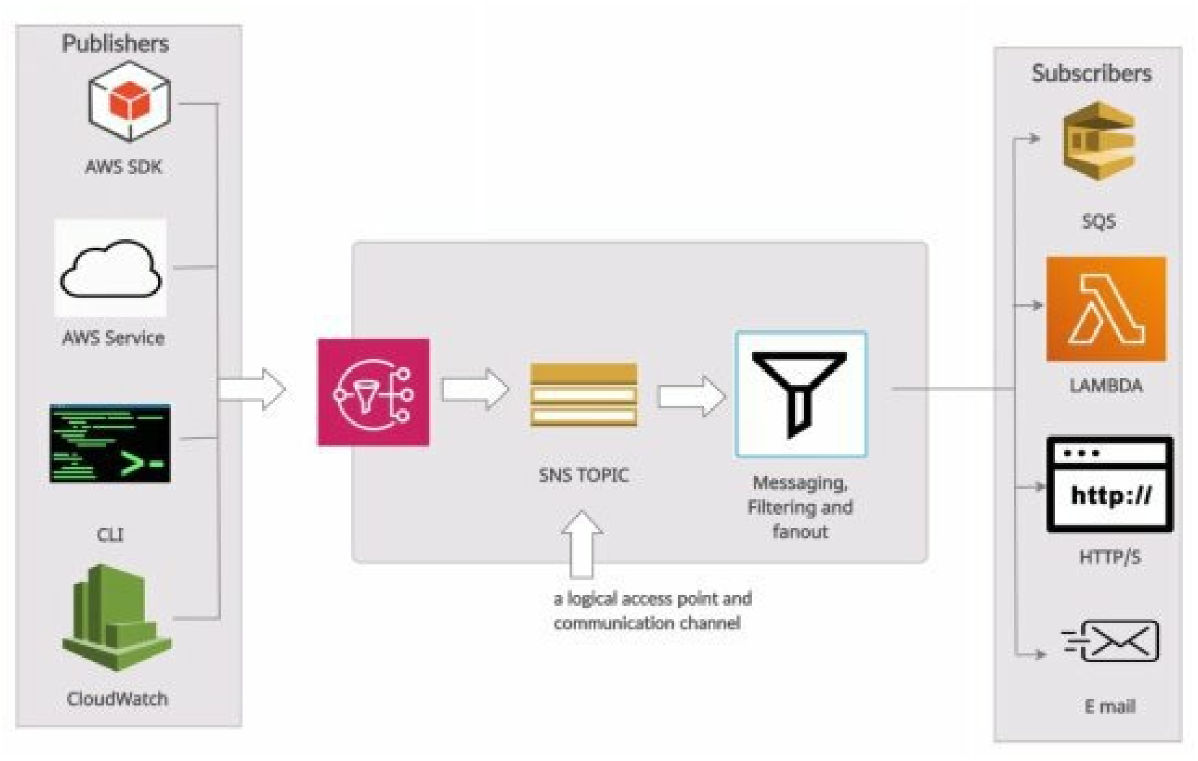
Introduction

SNS stands for *Simple Notification Service*. Using SNS, you can send notifications and subscribe via *email, text messages, Lambdas, web books, mobile notifications,* and *SQS*. Let us learn the concept of the *pub/subsystem* for a better understanding of SNS. A Pub/Sub is a *publish-subscribe* pattern widely used in messaging systems to convey messages between various components of the system without revealing their identities to each other.

In this system, *the publisher* is the sender of messages, and *the subscriber* is the one who receives them. The publisher sends a message to an *Event bus* instead of sending it to the receiver directly. Then, the Event bus categorizes the messages into different groups/classes, and the receiver of the message subscribes to these groups. So if a new message arrives within their subscription, it is immediately sent to them.

The *message buses* are the servers on which messages are sent, stored, and collected. So, a variety of topics can be included in each message bus. *Topics* are used to differentiate between various types of communications since not all subscribers need to view each message. Publishers do not know about their subscribers and vice versa. Subscribers need not pull the messages from the event bus as these messages get pushed towards subscribers automatically. Furthermore, Events and messages are interchangeable terms in the pub/sub environment.

Now, let us discuss the SNS. The SNS is a fully managed and highly available pub/sub messaging service which allows you to decouple (set apart) microservices-based architectures, serverless and distributed systems. When we say decoupling, we refer to application integration which is like a family of AWS services connecting one service to another, and SNS is an application integration service. A typical example of an AWS pub/sub system is shown in the following diagram. The publishers are on the left side, subscribers on the right side, and the event bus in the center is SNS. The publisher refers to any service which programmatically uses the AWS API. In this case, there are AWS SDK and AWS CLI as publishers that use AWS API underneath to publish their messages onto an SNS Topic. The CloudWatch can also publish to SNS because you would use them for Billy alarms. There can be another service on AWS that can publish to SNS topics.



On the other hand, the subscribers subscribe to a specific topic in the event bus. We have subscribers with several outputs, such as *SQS*, *Email*, *HTTP/S protocol*, and *Lambda*. The benefit of the pub/sub system is that you can have as many subscribers as you need and scale them all separately. It depends on the amount of time a single subscriber takes for every message to process and the traffic on the given topic they subscribe to.

Therefore, the publishers will push the messages to the SNS topic to get into the event bus. After that, the subscribers will subscribe to the SNS topic so that the events get pushed towards them. The event bus 'SNS' takes published messages as an SNS topic, filters them, and then forwards them to all the registered subscribers.

SNS Topic

A *Topic* is an access point or a communication channel where you can subscribe using a phone number, email address, or a URL to receive its relevant notifications. A topic combines multiple subscriptions together. You can deliver a topic to multiple protocols at once. When a new notification is posted, it is filtered to comply with the policies of subscribers and, if successful, sent to the specified endpoint. The publisher is not concerned about the subscribers' protocol. You can encrypt your topic with the help of KMS. Note that SNS does not guarantee the delivery of messages as the SQS does. Moreover, if the sending is unsuccessful for some reason, it will not retry. If it happens, it will delete the message.

Use cases

SNS can be used frequently for sending system notifications to the subscribers of a given topic. They include warnings when the server is down, a new user registers, or when it performs a regular check. Since different users take an interest in different topics, some of them are created to avoid unnecessary emails or texts to the entire company. There is another popular use case, which includes a ‘fan-out’ scenario that uses Amazon SQS. In this scenario, a new message is sent to the topic at every new purchase from the store. After that, SNS sends the same message among its subscribers. These messages are directed to an email address (to send a notification of a bought product for a manager), an HTTP/HTTPS endpoint to update a client record in *CRM,* and to several SQS queues as well. These queues are as follows:

- Queue 1 – send to a warehouse to allow the workers to select a product and forward it to the next team.
- Queue 2 – deliver to a purchasing department to keep them updated about remaining stock.
- Queue 3 – send to the shipping team to create a shipping tag and prepare the packaging.

Testing the internal workflows is another use case of the SNS-SQS collaboration. You can add a fourth SQS queue in the above example. It will deliver all order confirmations to the development environment of your application. After that, you can use that production data to get rid of the errors and introduce changes to the production environments. However, if you want to test the email workflows, an external tool will be good enough to test them.

Subscriptions

Subscriptions are created on a topic. So, if there is an email subscription, then the endpoint will be an email as well. For getting an email subscription, first, you enter an email address, click on '*Create Subscription,*' and fill in the options. Then you need to select your protocol from a detailed list. We have different protocols such as *platform application endpoints, HTTP/s, Email, Email-JSON, SQS, SES, SMS,* and *Lambda*. A *Platform Application Endpoint* is used for the mobile push, which enables you to get a notification system on various devices such as cell phones and laptops. The protocols *HTTPS* and *HTTP* are used for web books. HTTP/S is actually an API endpoint to the users' web applications that listens to incoming SNS messages for sending emails.

The *Email-JSON protocol* will send you JSON objects via email, where you can get email notifications that are text-based messages. After Email-JSON, there is *SQS* where you can send an SNS message to SQS. Then we have another service for sending out emails, which is the *Simple Email Service*. It is very useful for internal email alerts to the customers as you do not necessarily need your customized domain name. It can be used for billing alarms or checking any sign-up activity on your platform. You can also use *SMS protocol* to send text messages. Lastly, one can also use *SNS* to trigger the *Lambda* function for messages alert, which is a very effective feature.

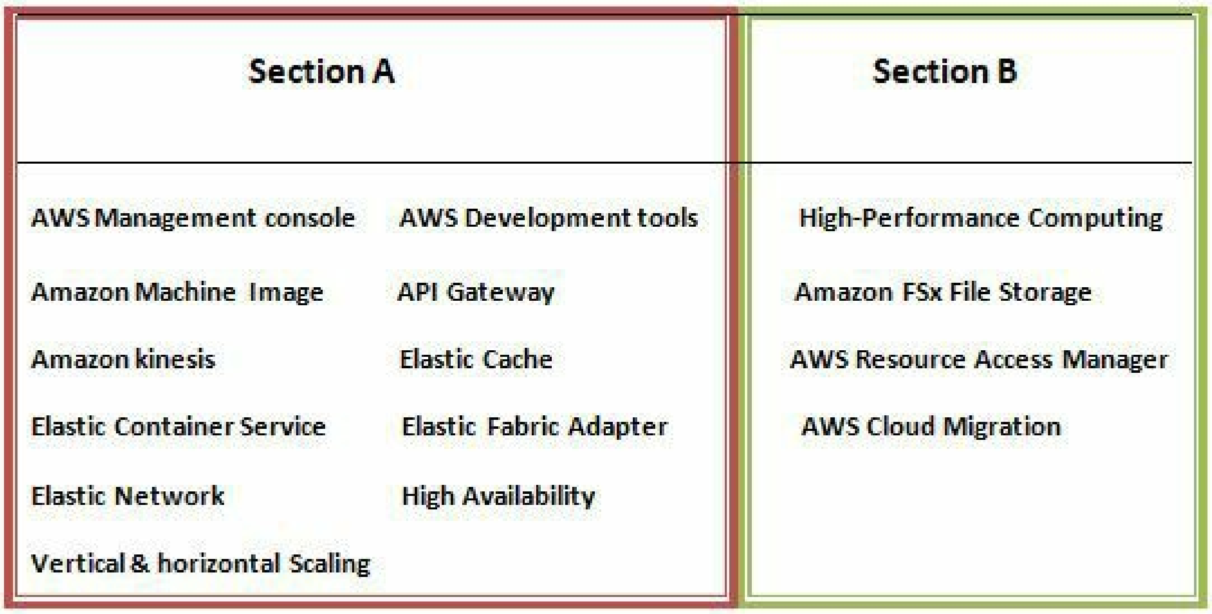
Application as Subscriber

We are going to discuss Platform-Application-Endpoint in a bit more detail as it is used for mobile push. The mobile endpoints available are *Amazon Device Messaging* (*ADM*), *Google's Firebase Cloud Messaging, Apple's APNs*, and *Baidu*. For *Microsoft*, we have got *Window Push* (*WNS*) and *Microsoft Push* (*MPNS*). Using the *Application as a Subscriber protocol,* you can send alert messages to the mobile endpoints. The advantage is that notification messages will appear as a message alert, sound alerts, or even badge updates in the mobile application when you push messages to these mobile endpoints.

CHAPTER 9: IMPORTANT SERVICES

Overview

AWS consists of several cloud services that can be used in combination according to your organization or business requirements. The rest of the AWS services are categorically discussed in this chapter. This chapter is divided into two sections.



Section A		Section B
AWS Management console	AWS Development tools	High-Performance Computing
Amazon Machine Image	API Gateway	Amazon FSx File Storage
Amazon kinesis	Elastic Cache	AWS Resource Access Manager
Elastic Container Service	Elastic Fabric Adapter	AWS Cloud Migration
Elastic Network	High Availability	
Vertical & horizontal Scaling		

Let us first look at the two developer tools and the Management Console of AWS, used for accessing almost all the Amazon web services.

AWS MANAGEMENT CONSOLE

It is a graphical method for setting up, connecting, and modifying different AWS resources. It is a GUI browser for Amazon Web Services (AWS). The console allows the customers to interface with all the AWS resources and handle their cloud storage, cloud computing, and other resources running on Amazon's cloud infrastructure. AWS customers can manage their accounts by monitoring monthly expenses. You can also launch new applications and manage the ones you already have. This console provides educational resources to adapt to the cloud, such as workflows and wizards. It is simpler to use and does not require any scripting.

You can drag and drop the service link for a customized view using the console. You can see and combine applications and resources that share common tags and make quick changes in the entire resource group using the *Tag Editor* feature. The mobile app of AWS Console helps you in performing mobile operating tasks. You can download this mobile application from the app stores of *Google*, *Amazon*, and *Apple*. AWS Management Console offers support for various browsers such as *Firefox, Safari, Microsoft Edge, Chrome,* and *Internet Explorer*. Moreover, there are separate Android and iOS consoles. Apart from the console, there is another useful tool for managing AWS services using scripts. It is known as the Command Line Interface, which we are going to discuss next.

AWS DEVELOPMENT TOOLS

Command Line Interface

It is an open-source tool for handling and automating several AWS services from a command line by using scripts. With the help of *CLI*, you can communicate with all AWS services, control their behaviors, and automate service management and AWS infrastructure resource. You can install it on *macOS, Docker container, Windows,* and *Linux*. With AWS, it is easy to manage any service for Windows, Linux, and macOS users by simply using the command line of a terminal session. Besides, the functions of console can also be used via a terminal program as it involves minimal configuration and one-step installation process. These terminal programs are:

Windows Command-Line: Windows users can run their commands in *Windows Command Prompt* or *Power Shell.*

Linux shells: One can use command shell programs such as *tsch, zsh,* and *bash* for running various commands in the operating systems, including *Linux, Unix,* and *macOS*.

Remotely: The remote terminals such as *SSH* and *PuTTY* can run the commands on the Amazon EC2 instance. With *AWS Systems Manager*, you can automate operational activities on AWS resources.

AWS CLI allows easy access to the Amazon web services' public APIs. Along with its API equivalent low-level commands, CLI provides customization for a range of services.

Since CLI allows users to use a command-line for all interactions with AWS, let us assume you are using AWS CLI on a terminal that starts with AWS, then in such a scenario, *AWS Python* script will be used to install it. You will then type AWS within the system along with some other commands after installing CLI. You can perform multiple operations, which include list buckets, start, stop, or terminate EC2 instances, upload data in S3, update security groups, and create subnets. Therefore, you can execute countless operations with this tool.

Installation of AWS Command Line Interface

It can be installed in three ways:

- Use a bundled installer
- Use pip
- Use a virtual setting

AWS Command Line Interface Installation via pip

We use *pip* for installing AWS Command Line Interface, a tool for package management for the installation and management of software packages written in *Python*.

Prerequisites

The Prerequisites for this process involves:

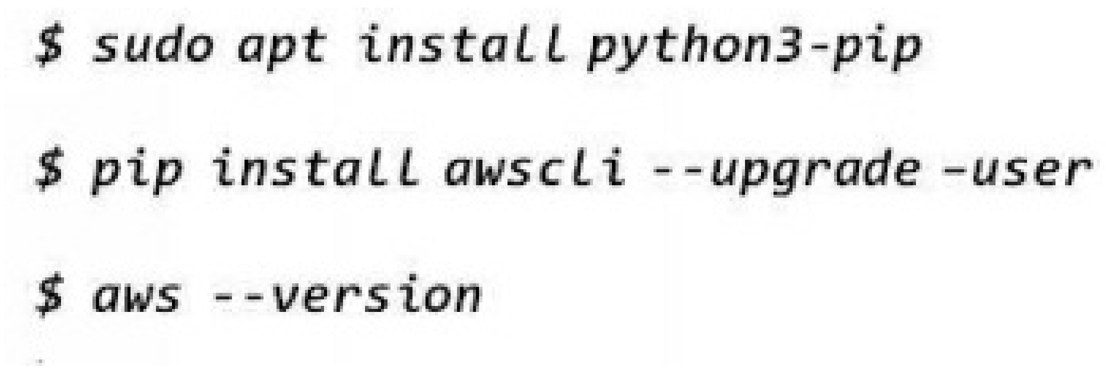
1. *Windows, macOS, Linux, or Unix OS.*
2. *Python2 /Python3*

Steps

Step 1: Install pip (on Ubuntu OS)

Step 2: Install CLI.

Step 3: Check installation.



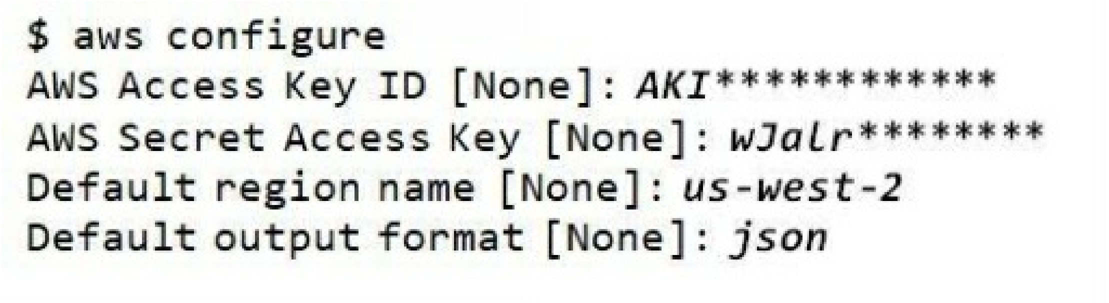
```
$ sudo apt install python3-pip

$ pip install awscli --upgrade -user

$ aws --version
```

After the installation, the next step is to configure AWS CLI to access the AWS services via the interface.

Step 4: For the configuration of AWS CLI, use the following commands:



```
$ aws configure
AWS Access Key ID [None]: AKI************
AWS Secret Access Key [None]: wJalr********
Default region name [None]: us-west-2
Default output format [None]: json
```

The users will need four sets of information as a result of the mentioned commands. Out of these commands, the first two are the AWS *Account Credentials*, and these credentials are mandatory. Regional and output format information is also required; however, users can leave it as default for now. Finally, AWS CLI is ready to use after completing the setup process.

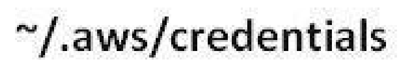
Now, our second development tool is going to be discussed next.

Software Development Kit

With the help of *SDK*, you can handle various Amazon web services using any common programming language. SDK, an important development tool, can be used to develop and manage AWS applications. This tool is mostly used with APIs attached to the programming languages to simplify the AWS service for the intended application. The Software Development Kit in AWS is a set of various libraries and tools that help users create applications for a certain software package. So it is basically a collection of API libraries that enable AWS services to be incorporated into your applications. It is offered in various languages such as *Go, C++, Java, Python, PHP, Ruby, JavaScript, .Net,* and *NodeJS.*

Programmatic Access

Programmatic access must be allowed for the users to use these two development tools. When it is being allowed, the access and secret access keys (the AWS credentials) will be provided to use these services. When these access keys are generated, you will need to store them in your home directory, where you will require a hidden directory known as '.*aws*,' and the filename is called *Credentials*. So, this is how you specify it:



```
~/.aws/credentials
```

If you do not specify any credentials while using SDK or CLI, it will use default credentials. However, one may end up with multiple credentials while dealing with multiple AWS accounts, but you can arrange them into profiles. It means that you can use the credentials file to handle multiple credentials (*profiles*).

While moving forward, a few more important AWS services and terminologies in this chapter will be discussed in detail.

AMI

Introduction

Let us discuss AMI in detail. *AMI (Amazon Machine Image*) is a packaged environment with the necessary components and software configuration for setting up an EC2 instance. It is a template for the configuration of a new EC2 instance, which provides essential data to launch it. As a result, you can turn your instance into an AMI and make replicas of your server. Moreover, these templates are configured via an operating system, application server, or any additional application necessary to provide a service.

The types of machine images depend on the launch permissions, operating systems, regions, the root devices storage such as Amazon Block Store or EC2 store, and system architecture (32/64 bit). While configuring an instance, you have to specify the source AMI. Moreover, multiple instances can be configured from one AMI when the instances with the same settings are needed, whereas you can also use different machine images if instances of different configurations are needed.

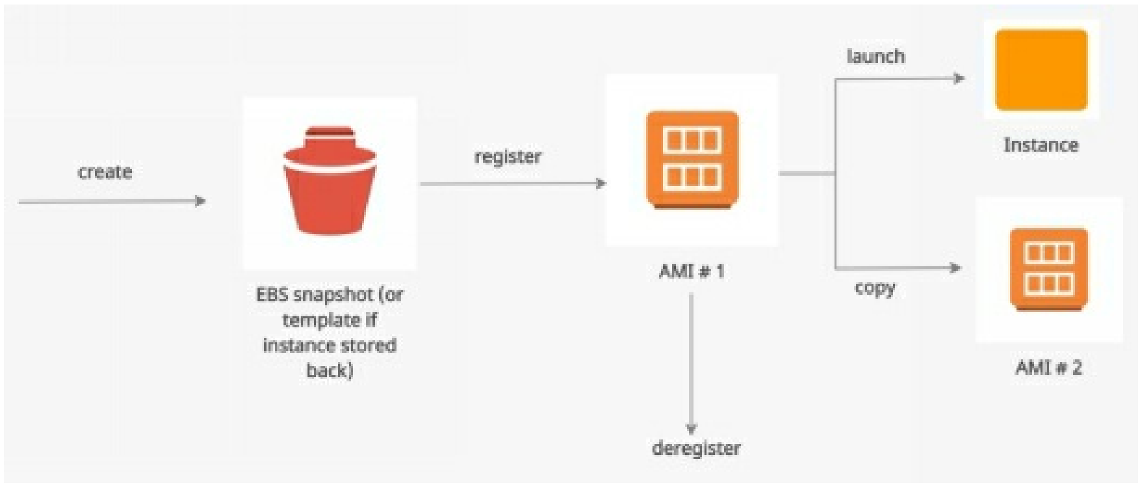
An AMI contains three elements:

Templates- It is used for the instance's root volume, such as OS or application servers.

Launching Permissions - They are used to allow AWS accounts to use AMIs for launching instances.

Block Device Mapping – When the instances are launched, this feature defines the EBS/ Instance store volumes attached to them.

The physical illustration of an AMI has EBS snapshots that are attached to it. You can either launch this AMI or replicate it for another one.



Use Cases

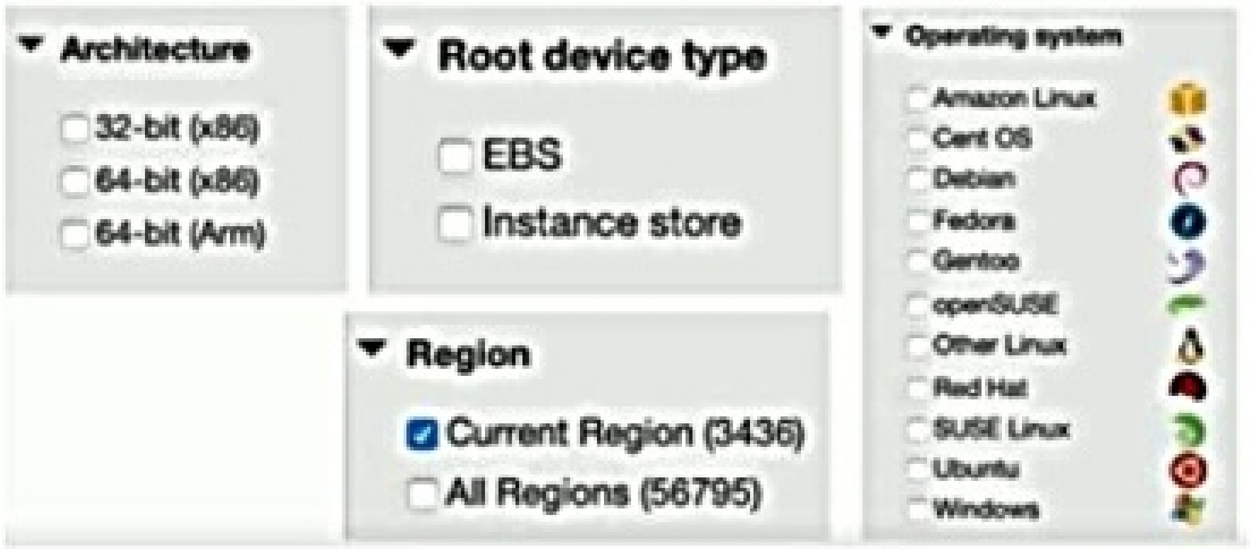
Let us talk about some AMI use cases. It allows you to keep incremental changes to your application code, system packages, and Operating System. Assume that you have built an AMI on a web server, and the AMI has some features already mounted on it. However, you needed to revisit for creating some revisions, such as installing *ImageMagick/ Redis* for image processing or *CloudWatch Agent* for log streaming from EC2 instance to CloudWatch. You will need to create those revisions at this stage, which completely depends on the names. Usually, AMI uses the service of *System Manager Automation* to frequently update security for your AMIs so that you can make those AMIs. Afterward, you can launch them instantly via *Launch Configurations*. While working with ASGs, they use launch configurations that must contain an AMI. When an AMI is attached to the launch configuration in your ASG, then updates are added to all instances. Thus, it was just to present a clearer picture of how AMIs works in AWS.

AMI Creation & Selection

The AMIs can be created from existing EC2 instances, which may be in execution or stopped. To make an AMI, you simply need to drop down to the actions, select an image, and then create one. We will now look at the selection of AMIs. There are many AMIs for you to search and select from *AWS Community AMI* and *AWS Marketplace*. Community AMI is the place for free AMIs maintained by the community, whereas at AWS marketplace, the AMIs are free and paid, which are maintained by vendors. Let us assume that you want to choose an AMI for your EC2 instance from AWS

Marketplace. By comparing the AMIs of North Virginia and Central Canada, you will find a few differences between them because although the AMIs are the same, there is a slight difference to satisfy various regions' requirements.

These AMIs are not one-to-one because they have unique *AMI IDs*, which depend on regions, making them region-specific. One cannot simply get an AMI ID from one region and launch it from some other region. You have the option to choose from a lot of AMIs with a few variations between AMIs for each region. The choice is based on multiple factors, such as your OS, the root device type, EBS or Instance Store volume, the AMI is available for the current region only or each region, and the architecture is 32 bit or 64 bit. So, you have got several filters to apply. The AMIs are categorized as *EBS-supported* or *Instance Store-supported*. You can also create replicas of your AMIs, which is another important feature as they are region-specific. You can only get an AMI from one region to the other is by using a copy command such as you need to copy an AMI and then select another region to send that AMI.



Shared AMIs

The Amazon Machine Images created by developers and available for public use are known as *Shared AMIs*. You can make and share AMIs on AWS. You need to be careful when using shared AMIs because developers or Amazon do not provide a guarantee or the security of AMIs shared by some other EC2 user. Due to security issues, AWS suggests the use of shared AMIs from trusted sources only.

AWS Marketplace

AWS Marketplace allows you to buy subscriptions for AMIs maintained by retailers. It also includes some free AMIs, but usually, they are for sale, which comes with extra charges on the users' EC2 instances. You can pay at an hourly rate if you use Microsoft's deep learning AMI, but normally users purchase from the marketplace. Among these AMIs, *Security hardened AMI* is very common. Consider that you needed to run Amazon Linux, where you have to meet the level 1 CIS requirement. In that case, this AMI will be available in the marketplace.

API GATEWAY

Introduction

API Gateway is a method of securing APIs in the user’s cloud environment at any scale. It is a completely managed service to build, maintain, publish, monitor, and secure the APIs. You may think of API Gateway as the AWS Cloud's backbone because it acts as a gateway between all of the services that are connected in the AWS system. It is written in the node.js programming language.

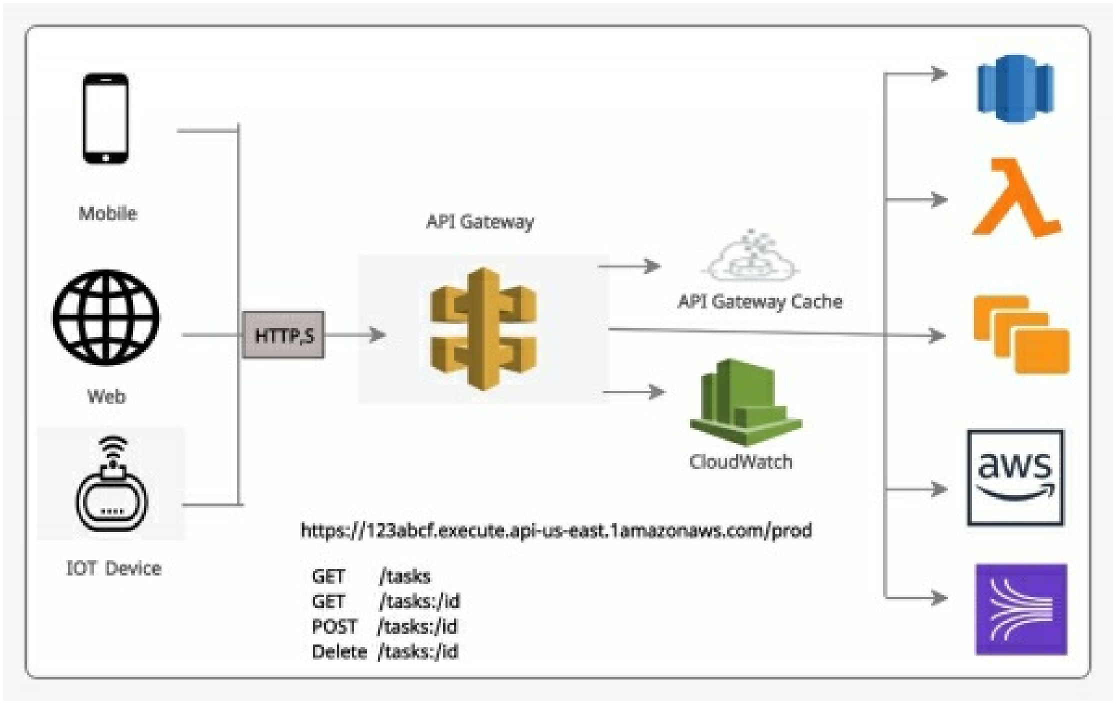
By using the API, multiple software applications can interact with one another to achieve higher functionality for a product. Within the API Gateway, you can build, maintain, and manage APIs. It will carry out all the operations which involve processing and accepting a very large number of API calls such as traffic monitoring, management, API version management, and authorization & access control. This service can be used to develop serverless apps, integrate with legacy applications, and directly proxy the HTTP requests towards other AWS services.

Using access keys with Amazon API Gateway can secure API access management. This service can interact with Amazon Cognito and IAM to authorize access to APIs. You can also pass *OAuth* tokens to operating workloads as an additional security measure. The service enables you to work with several versions of an API at the same time, which allows developers to create and launch new APIs while the already present applications continue to use their older versions. It incorporates several other AWS services, including AWS SNS, Lambda, Cognito Identity Pools, and IAM. You are only required to pay for the amount of data transferred and the API calls that you received. It offers additional data caching that is charged at per hour rate. This rate varies depending on the selected cache size.

How does it work?

Being a bridge between the users of your API and API's backend services, the API gateway handles the HTTP requests to the API endpoints and directs these requests to the accurate backends. You can manage the API definitions and the mappings between endpoints and their backend services, respectively, with the help of a set of API Gateway tools. With the help of this service, the API references can also be generated from your definitions and can be made available to the users in the API documentation.

Let us show you the working of the *API Gateway*. Assume a web app, an IoT device, or maybe a mobile app making HTTP requests to your API that is normally a *URL*. The API gateway will provide you the generated URL, and then you can create your endpoints. So, there are endpoints with different methods to perform different tasks that create virtual endpoints directed towards AWS services. The most popular service among AWS services is Lambda. The API works as a gateway for the apps to access business logic, functionality for backend service, and the data. One can say that it is a group of virtual endpoints to get linked with Amazon web services.



Configuration

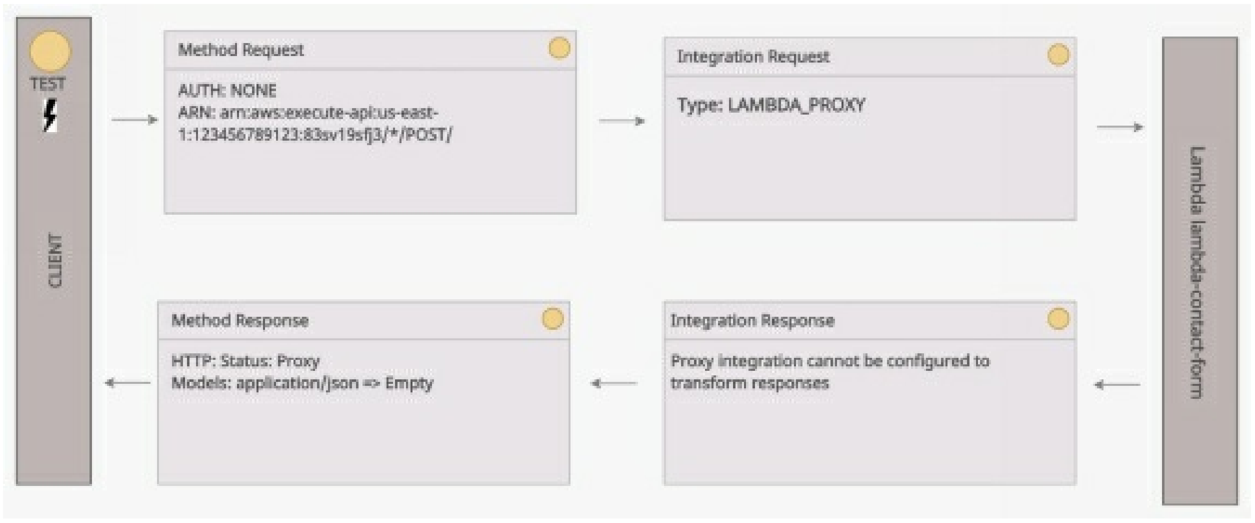
Let us explore how the API and its components are configured. *Resources* are categorized as the first and most essential element in configuring API. These are the users' defined *URLs*, such as */projects*. As you will have several endpoints in an API gateway project, you need to create several resources. Below the main resource, there is another resource *(/-id-)* that is the child resource of that parent, which creates a full URL like */projects/-id-/edit.*

After this step, you will apply *Methods* to these resources. They are the HTTP methods, which include *patch, post, put, delete, head,* and *options*. A user can do *POST/project/-id-* and *GET/project/-id-*, which are two unique endpoints, and both POST and GET will have separate functionalities. So, you can define multiple resources, and they can have several methods.

When the APIs are defined by using methods and resources, we are ready to publish the API. In order to *publish* your API, you will need a way of versioning the API for published versions, which means that you need to set up various stages. Generally, this procedure depends on the environment, so you will use *QA* for quality assurance, *Prod* for production, and *Staging* for developers. Any of these stages can be created. When a stage is created, a special URL is automatically created by AWS. This URL is known as *invoke URL*, which is the target endpoint. In this case, you have to write '/prod' and enter your endpoint. So you had */projects* in the previous example, you will rely on it and use the correct method (either POST or GET). So this is the procedure to interact with the API gateway.

Custom domains can also be used for your invoke URL in the API Gateway to be used for long-run URLs such as *api.fullhosting.co*. However, at the end of the URL, they have one for all stages such as */qa, /staging,* or */prod*. If you intend to deploy these versions to the staging, you will need your actual API and need to *Deploy API* action. The Deploy API action has to be done manually every time you make a change. It may not be very clear because one may think that the endpoints are created, but they do not work as you need to do it manually.

So far, we have discussed the definition and deployment of API. Now, the final step is to set up those endpoints. When a method is chosen for a resource, the *Integration Type* must be selected. There is a variety of them, including *HTTP*, *Mock,* and *Lambda*. You can deliver it to some other AWS service. You have also got a VPC link that goes to your local data center. The options can change once you are done with it, and the most common among them is the Lambda function. You will generally see this message: *You get to configure the request coming in and the response going out.* You can also apply authorizations such as *Auth: NONE* so that they can authorize and authenticate. After that, you have successfully set up Lambda and got a few modifications to the response outcome.


TEST
CLIENT
Method Request
AUTH: NONE
ARN: arn:aws:execute-api:us-east-1:123456789123:83sv19sfj3/*/POST/
Integration Request
Type: LAMBDA_PROXY
Lambda lambda-contact-form
Method Response
HTTP: Status: Proxy
Models: application/json => Empty
Integration Response
Proxy integration cannot be configured to transform responses

Key Features

Now we will discuss a few key features of API gateway.

- You can define strategies by using API Gateway, which restricts and meter third-party developer access to the APIs.
- You can configure a cache for your API data with personalized keys and TTL in seconds to avoid reaching the backend services every time a request comes in.
- With API Gateways, you can create, update, and delete documentation related to all portions of your API, which includes resources and methods.
- Due to the cost-effective and highly scalable nature of API Gateway, you can create, analyze, launch, and handle your APIs.
- With API Gateway, you can manage your system’s backend traffic as it enables you to configure the throttling rules that depend on the requests per second for every HTTP method within the API.

- With API Lifecycle, it allows you to execute several versions of the same API at the same time.
- It is able to execute Lambda code in your AWS account, make calls to Elastic beanstalk, ECS, or web services having public HTTP endpoints, and start Step Functions state machines.
- It provides general availability to HTTP APIs, which ensures the transfer of requests to private ELBs, Amazon EventBridge, AWS AppConfig, SQS, Step functions, Kinesis Data Stream, and ECS tasks in AWS CloudMap. Earlier, the HTTP APIs only allowed the users to create APIs for the serverless applications or proxy requests to HTTP endpoints.

Some other features of API Gateway include API Gateway Caching, Same Origin Policy, and CORS, which are going to be discussed next.

API Gateway Caching

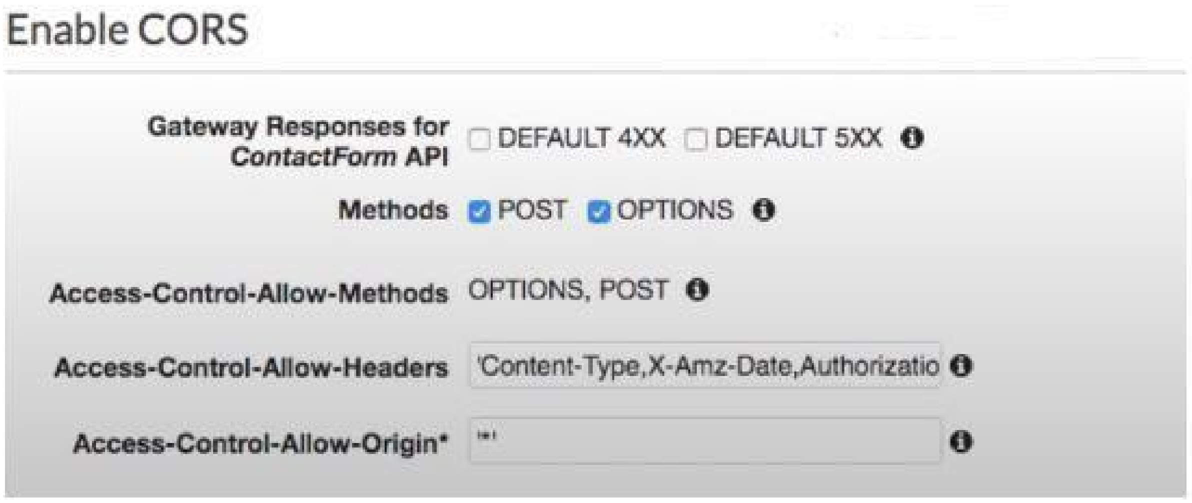
To achieve cost-effectiveness, you can enable *API Gateway Caching*. It caches the outcomes of common endpoints, which will also decrease the load on your API. When it is turned on for a stage, it caches the response from the endpoint for certain Time-to-live (TTL). Instead of sending a request to the endpoint, it addresses the request by checking the cache responses. As a result, the number of calls made to the endpoint will be decreased, ensuring cost-effectiveness and improved latency for the API requests, leading to a much better experience. Therefore, you surely want to enable the API Gateway Caching.

Same-Origin Policy

Users need the *Same-Origin Policy* when they are exposed to the *XSS attacks* (*Cross-Site Scripting*). It happens when a script tries to get executed to your website by using some other website. Usually, it holds malicious reasons, which means that when an individual tries to do something malicious, they do not use your website as you host it. So, they will surely use some other website. Therefore, to avoid such scenarios by default, a web browser will limit the ability to run those scripts that are cross-site from some other site. To execute such scripts, you will need a Same-Origin Policy, which states that certain scripts are allowed to run from another website. Again, the web browsers will impose it by default, but they will neglect the same-origin policy if one uses certain tools such as Curl and Postman.

CORS

Cross-Origin Resource Sharing (CORS) is used to deal with the problem of the same-origin policy. The same-origin policy defends the users from the XSS attack, but sometimes you want to actually access applications other than your own domain name. In such cases, CORS plays its part and enables you to do so. CORS is the header which states that a specific domain is granted access from X location to run applications. It is a header file or a document to execute scripts on. The client always imposes it. Generally, when using the API gateway, you need to enable it. Besides, you have to enable it for the entire API or certain endpoints such as CORS or in those cases where headers must be passed with the endpoints. In the following use case, *OPTIONS, POSTS,* and *Access Cross-Origin* are enabled.



Enable CORS

Gateway Responses for *ContactForm* API	☐ DEFAULT 4XX ☐ DEFAULT 5XX
Methods	✓ POST ✓ OPTIONS
Access-Control-Allow-Methods	OPTIONS, POST
Access-Control-Allow-Headers	'Content-Type,X-Amz-Date,Authorizatio
Access-Control-Allow-Origin*	'*'

AMAZON KINESIS

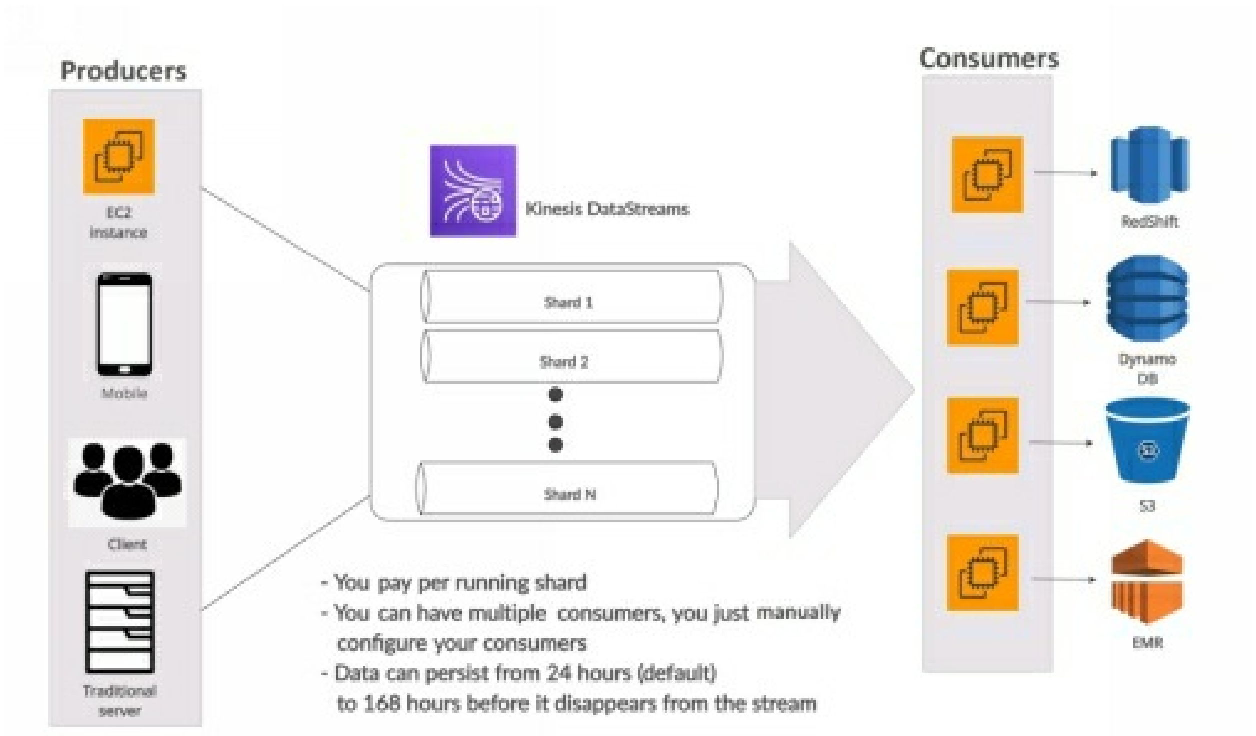
Introduction

You are now looking at a scalable and reliable real-time data streaming service known as *Amazon Kinesis*, which uses various sources to deliver and process data in real-time. It is entirely managed by AWS to collect, store, and process the streaming data in the cloud. It is used for data streaming in stock prices, gaming, geospatial, social media, and the clickstream. It provides four types of streams. Let us discuss them one by one.

Types of Kinesis Streams

Data Streams

Kinesis Data Streams consist of producers, shards, and customers. *Producers* generate data that is sent to a kinesis data stream which absorbs it. The absorbed data is then distributed among the shards, and *Consumers* receive it. A consumer is dedicated to sending data to Redshift, Dynamo DB, S3, and then EMR. You can have multiple consumers but have to manually organize those consumers with data streams by writing some code.


Producers
EC2 instance
Mobile
Client
Traditional server
Kinesis DataStreams
Shard 1
Shard 2
Shard N
- You pay per running shard
- You can have multiple consumers, you just manually configure your consumers
- Data can persist from 24 hours (default) to 168 hours before it disappears from the stream
Consumers
RedShift
Dynamo DB
S3
EMR

Before you send data, you have EC2 instances dedicated for that data processing. By default, kinesis data will remain there for 24hours after it has entered the stream. This time can be extended up to 168 hours. If you intend to run this data through several consumers, you can do it by this data stream.

In the Kinesis data stream, you will only pay for spinning up the shards. So as long as the shards are running, an individual will pay the X cost of money for X quantity of shards.

Firehose Delivery Streams

Now, we are moving onto *Kinesis Firehose Delivery Stream*, which is comparable yet easier than the data stream. It uses an approach that has a producer, which delivers your data to the kinesis firehouse. The distinction here is that the data instantly vanishes from the queue as soon as it is consumed, due to which data will not persevere. You can choose only one consumer in the Kinesis Firehose, so it lets you choose among S3, Redshift, ElasticSearch, or Splunk.

The advantage of Kinesis Firehose is that the data consumption does not require writing any code, but you have no control over-consuming the data as it is very restricted. Firehose can influence the data passing through it, so if you have something in *JSON* and need to convert it into *Parquet*, the available options are very limited. You have to push the data into the right data format. So while inserting it into S3, maybe Athena would be consuming it. Firehose is now in the *Parquet file*, which is optimized for Athena. By zipping, it can also reduce the file size as there are varieties of compression methods. Moreover, it provides security and cost-effectiveness. Being inexpensive is another benefit of Firehose as you are only paying for what you are using.

Video Streams

Kinesis Video Streams is used for streaming video data. You have producers who send data (video or audio encoded data). This data can be collected from multiple sources like security cameras, web cameras, or even mobile phones. This collected data will be sent to the kinesis video stream, which will capture and preserve the encoded data so you can get it from services used to analyze video and audio data. These services include *SageMaker, Rekognition, Tensorflow,* HLS-based video playback, and custom video processing. Therefore, you can observe and deal with output video streams to *ML* or *video processing services.*

Data Analytics

Let us have a look at *Kinesis Data Analytics* now. It operates in a way that it takes an input stream, and then it has an output stream, which can be firehose or data stream, but the information will be passed through data analytics. This service allows you to run custom SQL queries letting you go through your data in real-time. So, if you feel the need to do real-time reporting, this is the service you should prefer. It can get a bit costly as it uses two data sources, which is the only drawback. Besides this, it is great for data analytics.

ELASTIC CACHE

Introduction

Let us discuss *Elastic Cache* now. It is used to handle *Memcached* or *Redis* caching services. Elastic Cache is a *Cache In-Memory data store*, a short-term storage location where you have instant access to data. Let us go through the concepts of In-Memory data store and Caching for a complete understanding of Elastic Cache. So, the process of keeping data into the cache is known as Caching, where a cache is a short-term storage location used for instant retrieval. On the other hand, the In-Memory data store stores data in RAM. Both the cache and the in-memory data store together form the Elastic cache. The high volatility is a drawback in this case because it leads to low durability, and the risk of data loss increases.

The Elastic Cache can help you in launching, executing, and scaling any *open-source* compatible In-Memory data store. Its frequent identification of queries and keeping these queries in the cache to attain extra performance is the most interesting feature. While using it in production use cases, the only drawback is that Elastic Cache is only available to those resources that are running in the same VPC. You can only link your EC2 instance with Elastic Cache when you are in the same VPC. Otherwise, it is not possible to attach anything with Elastic Cache outside of AWS.

Redis and *Memcached* are the two in-memory data storage options available in Elastic Cache. Let us discuss the similarities and differences between these two engines. When we talk about similarities, both engines provide sub-millisecond latency, data partitioning, and ease of use for developers. They also support broad programming languages level. For differences, let us move towards the next topic.

Caching Comparison

Let us talk about the difference between the two engines. Redis can take snapshots, perform replications, pub/sub, transactions, and support geospatial support. Redis can help users in performing numerous operations on the data and the available data structure. It is ideal for tracking unreadable notifications, leaderboard, and cached data in real-time. On the other hand, Memcached is a simple key/ value store favored for caching HTML fragments. Despite being very simple, it is very fast along with a limited number of features.

There can be a debate about which one is more rapid because, according to some users, Redis is more rapid than Memcached over the internet. However, from an exam viewpoint, Memcached is considered more rapid than Redis for smaller fragments.

AMAZON ECS

Introduction

Developers can manage containers and use their software in the cloud using *Amazon ECS-Elastic Container Service* without configuring a workspace to execute the code. Using API calls or task definitions, a developer who holds an AWS account can launch scalable applications and manage them, hosted on different servers called clusters. It allows you to launch containers on EC2 servers or *Fargate* (for serverless mode). You can use Fargate (Serverless Container Management Service) by combining it with the container orchestration service, including Amazon ECS and EKS. These deployment methods can be used to take care of the orchestration and server maintenance, allowing you to focus only on scheduling and deploying the containers.

It eliminates the necessity of configuration, execution, and management of the orchestration layer for you to build an efficient, distributed, and scalable clustering system. ECS is the alternative resource for *Docker Swarm, Mesos,* and *Kubernetes*. Using ECS, you (as a developer) can utilize Docker container for multiple tasks that ranges from hosting a website to running a distributed and complex micro-service requiring several hundred containers. One may integrate it with other AWS services, including IAM, Load balancers, and VPCs. Admins are able to modify security settings at the OS level and use other management resources in order to safeguard the ECS containers. Customers are not required to pay any additional charges for ECS, though the end-users do pay for EC2 instances and EBS volumes in the cluster and other payable resources of AWS.

How does it work?

Launching an app on Amazon ECS will require the development and deployment of its components for use in containers. It enables you to define the application by taking out the important Docker resources and images from various repositories and Amazon ECR. After that, you can keep the Container images in a registry, and later you may download and run them on Amazon ECS. When all the accurate containers are gathered, then they are launched on Fargate or Amazon ECS. In last, ECS scale your application and consistently handles the container availability.

To understand the structure of Amazon ECS completely, you need to get familiar with a few important definitions.

Task Definition: It is a JSON format text file that defines the application and is one of the most important parts of Amazon ECS. It consists of container descriptions ranging up to ten. In a microservice architecture, normally, it constitutes a single container definition. However, one may treat it as a unit of microservices and proceed to add more containers (making up separate microservices).

Task: It is the instantiation of task definition, and the same task definition can be used by multiple Tasks.

Scheduler: A *Scheduler* is accountable for positioning tasks exactly on the Amazon ECS cluster. A *Service Scheduler* is responsible for long-term active applications to automate scheduling the new tasks to an ELB. On the other hand, you can use a scheduler known as *Batch Job* for temporary tasks.

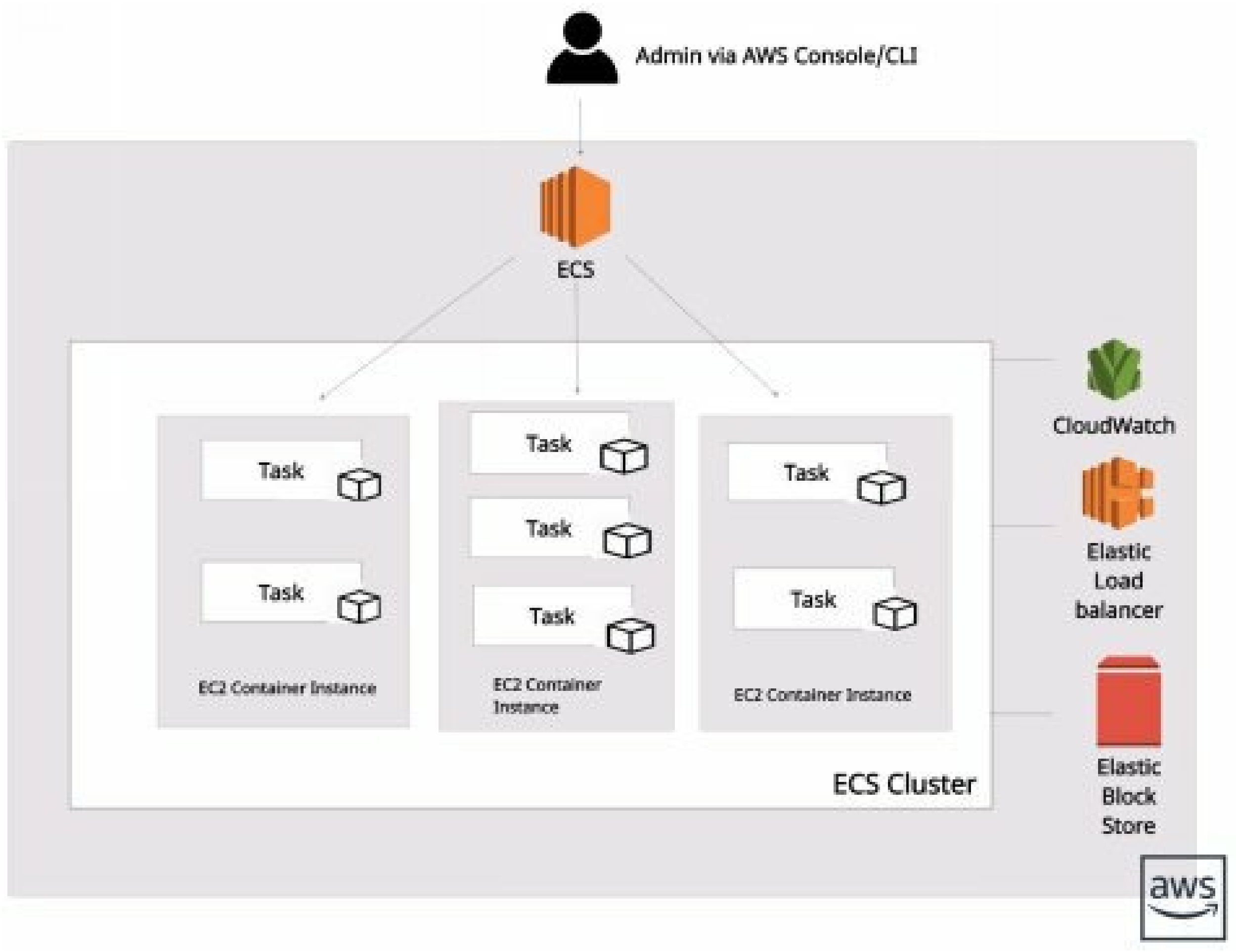
Service: On a cluster, you can run and manage a certain number of tasks or task definition instances at the same time. The service scheduler will replace the failed task by launching a new instance of the task definition in case of task failure. Depending on the selected technique of scheduling, this service maintains the anticipated number of tasks.

*Cluster***:** A *cluster* is a logical resource grouping on which you can execute tasks. Amazon ECS manages all the resources by selecting the Fargate option. On the other hand, if EC2 is the launch type, then the group of container instances will be managed by the cluster.

*Container Agent***:** It is a type of proxy between the tasks residing on the Amazon ECS and EC2 resources. In the case of the EC2 launch, it will execute on all the infrastructure resources within the cluster. It initiates and terminates the tasks when there is a request from ECS.

Elastic Container Registry: It is the AWS Docker containers' registry.

There are several tasks in the container instances that execute in the Docker containers. You can use ELB to distribute the traffic among the tasks dynamically. The EC2 container instance cluster spans several availability zones. The figure below will show the logical structure of the ECS cluster.



Uses of Amazon ECS

You can use Amazon ECS for the following use cases:

- For Inference and training purposes, ML models are easily containerized with ECS.

- It can help in the functioning of microservice applications by offering local integration into AWS and allowing the continuous deployment and integration of pipelines.
- You can migrate any legacy app to Amazon ECS and feasibly containerize them without requiring any code modifications.
- One may run batch workloads on various EC2 instances along with managed and custom schedulers.
- Linux virtual machines (which are called instances or VMs) can be launched and run in the cloud via ECS. For the isolated sets of EC2 instances, developers may define rules for improved computational efficiency and portability.

ELASTIC FIBER ADAPTOR

Introduction

This network interface designed for EC2 instances is known as *Elastic Fiber Adaptor*, which speeds up the ML and HPC workloads requiring high-level communication between the nodes on AWS, such as *water modeling, reservoir simulation*, and *computational fluid dynamics*. Along with a new custom *Scalable Reliable datagram* protocol, EFA uses an industry-standard *OS bypass* technique to improve inter-instance communication performance for scaling HPC applications.

Previously, HPC apps interacted with the transport layer of the system's network using the MPI (Message Passing Interface). In the AWS Cloud system, the application will interact with MPI that will use the ENA device driver and TCP/IP stack of OS to allow the networking between the instances. An HPC application that uses EFA communicates with the Libfabric API using NCCL or Intel MPI. To place packets on the network, the Libfabric API interacts directly with the EFA device and bypasses the OS kernel.

Along with all the functionalities of an Elastic Network Adaptor and additional characteristics of OS bypass, EFA also provides hardware support for the application to communicate with the EFA directly. Due to the advanced programming interface, it does not even involve the instance kernel. EFA was built to work with the existing network of AWS infrastructure and has the scaling ability to fulfill the application requirements. You can use it without any extra charges for a supported EC2 instance and avail it as an additional feature of EC2 networking. It is compatible with the commonly used interfaces, APIs, and libraries for the communication between the nodes to migrate HPC applications to AWS with a few changes.

Use Cases

- Tightly Coupled applications
- HPC applications
- ML and MPI use cases

Limitations

- The EFA OS bypass traffic cannot be routed. However, regular IP traffic from EFA is allowed to be routed.
- Only one EFA is able to attach to an instance.
- The OS bypass traffic of EFA is bound to a specific subnet. EFA traffic cannot be sent between subnets. One may send regular IP traffic from one subnet to another.
- A security group must be attached to EFA, which permits all the traffic to and from SG.

ENHANCED NETWORK ADAPTOR

Introduction

ENA is the new-generation network interface with the linked drivers for the enhanced networking of EC2 instances. The custom network interface was built to ensure low latencies, higher throughput, and high packet per second performance (PPS) on EC2 instances. It is intended to operate with modern processors and can use several virtual CPU technologies of these modern processors (such as processors of X1 instances). Therefore, the enhanced networking feature of ENA is compatible with X1 instances and supports a network speed of 100Gbps. Apart from C4, M4 (smaller than m4.16xlarge), and D2 instances, the latest generation of EC2 instances is using ENA for enhanced networking. In addition, network bandwidth of 24Gbps on certain EC2 instances is provided by ENA.

ENA driver needs to be installed in the system to achieve maximum network throughput from the EC2 instances that are supported by ENA. At the instance level, you must ensure that you have enabled the Enhanced Networking feature. The ENA will grow and scale as the capacity of bandwidth and the CPU count increases, which allows users to get advantages of higher bandwidth without the requirement of introducing new drivers or make any modifications in the configuration.

ENA was built to ensure the network does not throttle the customer's workload, as it provides traditional IP networking functionality required to support the VPC. It minimizes the burden on the host processor and transmits the workloads of packet processing efficiently through many virtual CPUs. As a result, you will achieve flexibility and elasticity of the AWS cloud and the application efficiency of in-house HPC clusters. It is an additional feature of the EC2 networking, supported on EC2 instances without any supplementary costs.

Use Cases

- It is better with a use case that requires higher bandwidth and lower inter-instance latencies.
- Only HMV instance types are supported by ENA.

Benefits

- ENA is compatible with devices having *Multi-Queue Interfaces*. Multiple transmit and receive queues are used by ENA to enhance the scalability and reduce the workload internally. The process to map an incoming and outgoing packet to a certain vCPU is simplified and accelerated by the presence of multiple queues.
- ENA drivers support the offload characteristics for an industry-standard TCP/IP model, which includes *TSO-Transmit Segmentation offloads* and *checksum* offloads.
- *RSS-Receive Side Scaling* is also supported by ENA. In a multi-processor system, RRS makes it possible to distribute network processing for multicore scale across multiple CPUs efficiently. *Low Latency Queue (LLQ)* is also supported by a few ENA devices as it saves some microseconds.

HIGH AVAILABILITY

The ability of a system to remain available is known as *High Availability*. To promise the high availability of a system, you must pay attention to the factors that cause service unavailability and their solutions. Let us explain the high availability of a system with the help of different scenarios. In the first case, let us say that an Availability Zone has flooded for a certain reason, and the servers are no longer running, so it becomes unavailable. In such cases, you will need an EC2 instance in some other data center. You will also need an ELB to move the traffic from one AZ to another AZ. In this way, you will achieve a *Multi-AZ environment*. In the case where two AZs become unavailable, a third AZ will be required. Therefore, one must have three AZs, as it is the least requirement for many companies.

Let us consider another use case where a region turns out to be unavailable. Assume that a meteor strikes in a certain geographical site, which is a rare situation. However, we assume that it wipes out the complete region, including every data center in that region. Therefore, in such cases, you need to have instances functioning in another region, which moves the traffic from one region to the other region. In order to do so, you will need Route 53 to direct the traffic as it is a solution for such situations.

In the third use case, assume that a web application becomes unavailable due to heavy traffic. You will need more EC2 instances to satisfy your needs according to the traffic loads in such cases. So, you need to use Auto-Scaling Groups at this stage as their scaling depends on incoming traffic intensity.

In the next use case, the unavailability of an instance is caused by instance failure**.** In such cases, hardware or software malfunction usually occurs, which is considered unhealthy. The Auto-Scaling Groups will again come to the rescue in such use cases where you set minimum instances. For instance, you need to have at least three running instances for managing the load. As if any of these instances fails, it spins up another instance, and the ELB routes the traffic to another instance and availability zone, ensuring the high availability in each use case scenario.

In the last scenario, the distance between the geographical areas causes an application to become unresponsive. Assume that you are in North America, and a user tries to access your web application from Asia. In such cases, the distance between two geographical areas will limit availability. So, there are a couple of options, such as you can use CloudFront for the content high accessibility and high availability to the users by caching the dynamic and static data to some extent. For running the servers and instances in the neighboring regions, the Geolocation Routing Policy of Route 53 can be used here. As a result, when a server resides in Asia, it will direct the traffic to those servers.

Scale-Out and Scale-up

Let us look at the important concepts of *Scale-Out* and *Scale-Up*. When the instance capacity is exceeded due to the increased demand, one can either perform vertical scaling or horizontal scaling, where vertical scaling is known as scale-up, and horizontal scaling is called scale-out. To Scale-up, you only increase the size of an instance to satisfy the capacity requirement. Although it is easy to manage because we only increase the size of an instance, however, there is a downside that you will have to cope with is lower availability. Therefore, the service will not be available if there is a failure, even in a single instance.

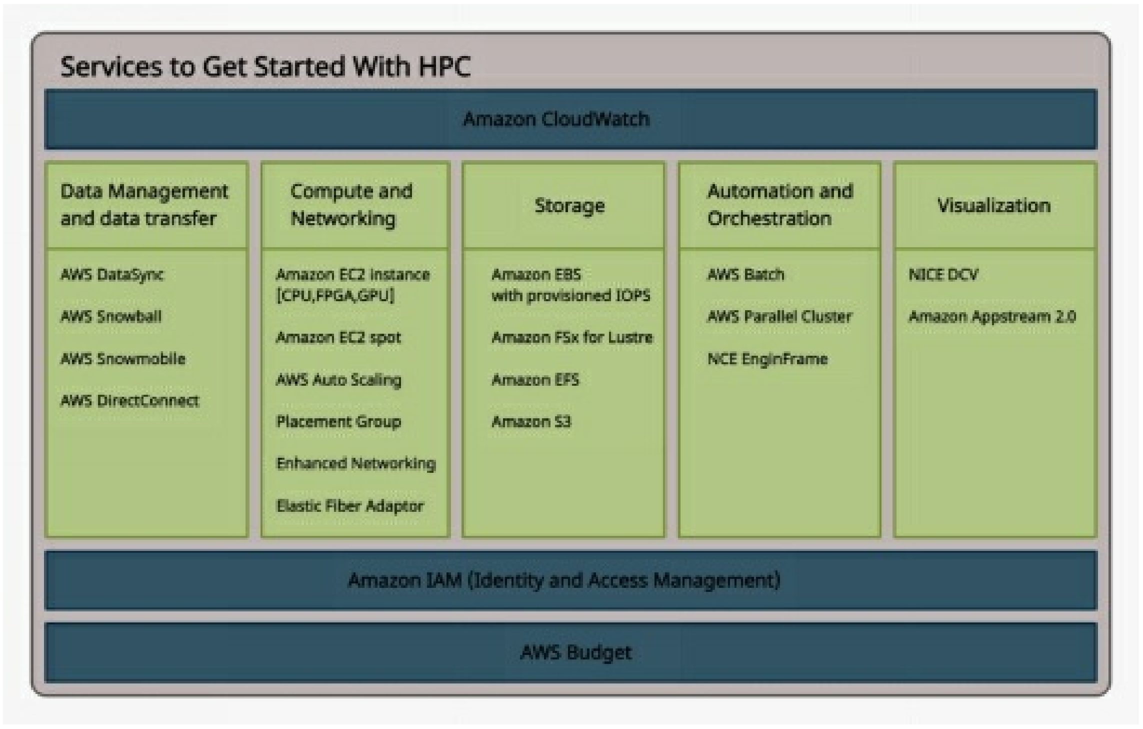
While in the case of horizontal scaling or Scale-out, all you need to do is to add more instances. An advantage of horizontal scaling is that you will achieve higher availability, which means that it will not make any difference if there is a failure in a single instance. However, managing additional servers would be difficult. So, first, you can scale out to achieve higher availability, and then you can scale up to maintain simplicity for a better experience. Therefore, you need to use both procedures as it depends on the scenarios.

HIGH-PERFORMANCE COMPUTING IN AWS

The capacity to process data and carry out quadrillions of complicated calculations in a second is known as *High-Performance Computing (HPC)*, for example, supercomputers. HPC solutions use thousands of compute nodes operating together for completing the tasks, which is referred to as *Parallel Processing*. It is the base for societal, industrial, and scientific developments. The amount and size of data that enterprises are dealing with are rapidly increasing; therefore, the HPC solution is useful for multiple purposes, including testing new products, streaming a live event of sports, analyzing stock trends, and monitoring a developing storm. In order to design an HPC architecture, the compute servers are linked together into a cluster, and then to get the output, these clusters are connected to the data storage as well. These components (storage, compute, and network) collectively work in a very smooth manner to accomplish a distinct group of tasks. All the components must maintain the speed in order to work with maximum efficiency because if any one of the components loses pace, the whole HPC infrastructure performance will suffer.

In recent years, AWS Cloud Computing turned out to be a renowned source of computational power that provides multiple options to develop the HPC platform. The traditional HPC platforms fail to scale and provide restricted hardware access. The AWS Cloud is a low-cost, effective approach and emerging tech for HPC users' requirements due to its flexibility in scaling the compute nodes based on the app needs and customers' budgets. A few major characteristics of cloud computing include the rapid growth rate with the capability of both the software and infrastructure as a service provisioning, virtualization, and flexibility of resources. With these features, a certain application of HPC users becomes more customized and flexible. So, Cloud computing facilities have been developed as a potential target by the HPC community. It is a major reason for a lot of organizations and users to port their HPC applications to the AWS cloud.

AWS offers an advanced set of services that delivers everything you require for creating and managing cloud HPC clusters very easily and rapidly so that you can manage the compute-intensive workloads in different sections of the industry. AWS HPC solution eliminates the existing on-premises cluster complications such as unsustainable technology, high capital expenditures, and fixed capacity of infrastructure. AWS grants access to the unlimited HPC capacity that is developed by using new technology. The following diagram reflects the HPC services on AWS:



Due to the arrival of Cloud networking, compute, and storage solutions, the configuration and launch procedure of HPC workloads has become very simple. AWS EC2 offers strong compute and storage resources on-request via virtualization at the hardware level, and it also aims to be globalized. Currently, EC2 can run several applications at a time that previously needed to rent out time from a supercomputer center or create a huge HPC cluster. However, as you may expect, it cannot perform all the operations as a supercomputer do, but it offers computing feasibility on virtual parallel clusters.

There is a network interface for the EC2 instances known as EFA, which allows the app execution with high-level communication between the nodes on AWS. Its Operating System is custom-made, which improves the efficiency of the communication between the nodes by avoiding the hardware interface, which is critical to scaling such applications. Both HPC applications (that use *MPI- Message Passing Interface)* and Machine Learning applications (that use *NCCL-NVIDIA Collective Communications Library*) are capable of scaling up to thousands of GPUs by simply using EFA. Thus, you can get the application efficiency of HPC clusters of local data centers along with the on-demand cloud's flexibility and elasticity.

You have different options to launch HPC storage on AWS for supporting any HPC workload virtually. To support an object interface, Amazon S3 is a highly durable and scalable platform for storing HPC applications. It has many features like lifecycle management, which allows the users to move the files (that are not accessed frequently) to storages at a lower level for more cost-effective solutions. Amazon S3 can be used as a data archive for storing the datasets and using it in the file processing system, even if the HPC applications do not support it directly. Furthermore, it is possible to process your data that is stored on Amazon S3 or an in-house data center with a highly efficient file system called Amazon FSx for Luster. FSx for Luster can be used to launch and execute a file system that offers access to the data in sub-millisecond. You can read and write the data at the speed of millions of IOPS and throughput at hundreds of Gbps. The integration of *AWS Parallel Cluster* with FSx for Lustre gives HPC customers the option of spinning clusters with thousands of compute instances and sharing storage which scales the efficiency of most challenging workloads. As a result, you can have instant answers, and the compute instances do not need to wait for the data.

AWS created Parallel Clusters to overcome the complexity of creating an HPC system. It provides a simple method to build an HPC system that scales automatically in the AWS cloud by using the services such as FSx for Lustre, Amazon EC2, or Amazon Batch. It will set up the storage and compute resources for you that are essential to run HPC workloads. Settings that are focused on a specific network, compute, and storage parameters aid in maximizing its performance.

Using job scheduling and managed resource provisioning of *AWS Batch* for

EFA, you will be able to perform distributed HPC and ML workloads. The use of High-performance computing has increased in the financial sector in various fields such as risk management, pricing, and market positioning. AWS Batch can also be used to automate the scheduling and resourcing of these jobs to save money and increase the go-to-market speed and decision-making for you. AWS was the first major cloud provider that provides services designed according to the HPC community's unique requirements. The combination of advancements in storage, compute, and networking leads AWS to become a preferred platform for HPC applications.

AMAZON FSX FILE STORAGE

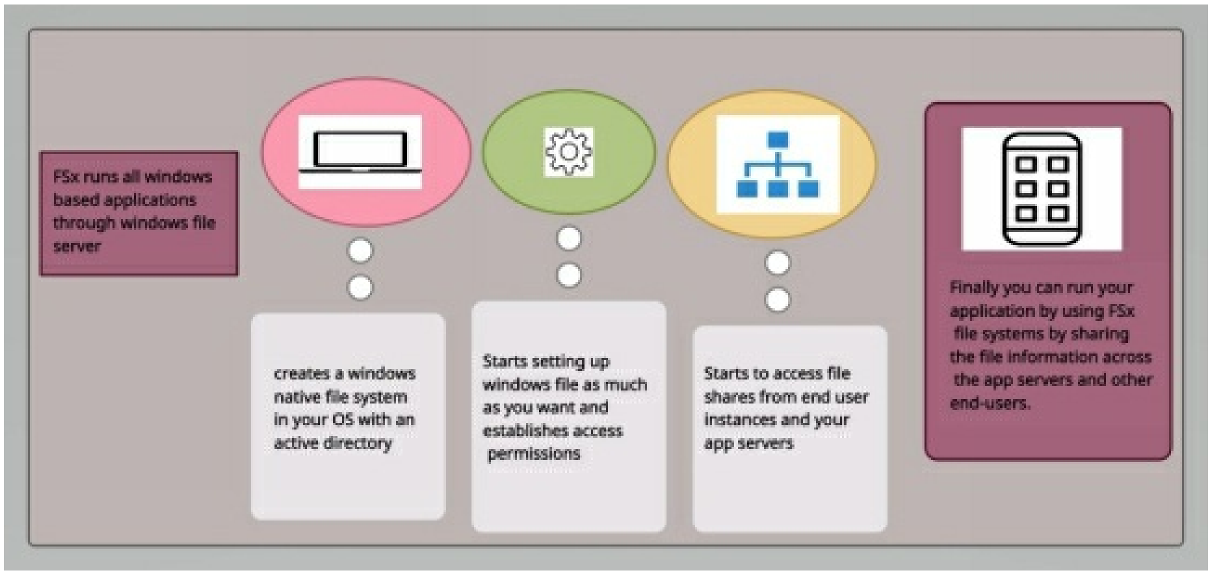
Introduction

Amazon FSx offers a network file storage solution that is more reliable, durable, business-oriented, and secure, which works perfectly with the current windows environment and applications. It focuses on file systems just like EFS. There are two forms of FSx, where one is FSx for Lustre and the other for Windows file Server. Every FSx option was created to meet a specific set of requirements. In the case of FSx for Windows, Amazon offers you a fully maintained MS-Windows file system on AWS. It is built on SSD storage, which includes a native Windows file system and fully managed Windows file servers that are accessible via the *SMB-Server Message Block* protocol. Windows-based tasks involving file storage can be quickly transferred and migrated by using this service.

On the other hand, the Amazon FSx for Lustre is used to provide a high-performance file system built for workloads that demand a lot of computation, such as ML and HPC. Using FSx for Lustre, the performance can increase up to millions of IOPS, sub-milliseconds latencies, and hundreds of Gbps of throughput because it has the ability to process large data sets. It supports cloud bursting workloads from on-premises on VPC connections and Direct Connect, and it also has integration with Amazon S3. Due to FSx for Lustre, the high-performant Input/output gets more user-friendly and accessible, particularly to the data analytics businesses.

You can launch Amazon FSx directly by using AWS CLI, AWS SDK, or AWS Management Console GUI. It is available in the majority of AWS regions, and the price is determined by the amount of throughput and the storage needed to be configured. In addition, it uses a replication feature known as *Distributed File System-DFS* to prolong the presence in a Multi-AZ environment; however, it will elevate the total cost for using this service. The enterprise-level performance and 2GBps throughput are achievable by using FSX. Global access with high accessibility simultaneously is also provided by shared storage. The Amazon FSx provides greater help for enterprise users, like providing data migration services with minimum efforts to lift and shift tasks in the cloud.

How does it work?


FSx runs all windows based applications through windows file server
creates a windows native file system in your OS with an active directory
Starts setting up windows file as much as you want and establishes access permissions
Starts to access file shares from end user instances and your app servers
Finally you can run your application by using FSx file systems by sharing the file information across the app servers and other end-users.

FSx Use Cases

Amazon FSx can be used for the following use cases:

- Software development platforms.
- Data analytics application.
- Shift and Lift applications.
- Home Directories.
- Media workflows include stream, transcode, and process.

With Amazon FSx, AWS satisfies the user's demands and requirements for the shared file support for the workloads, including *SMB* and *CIFS*. Due to the fact that FSx is developed on windows, it is completely compatible with all the applications that are windows based. You can integrate it with any workloads such as Active Directory, Distributed File System, and NTFS. The only workload that may be accessed by using the local AWS directory is AD, and without it, you cannot use this service.

Amazon FSx with HPC Use cases

Let us look at how FSx for Lustre can be used to solve business problems. Consider a scenario in which you need to discover the five years old patterns of archived client support tickets. Now, the very first problem that a non-data scientist will face while beginning a data project is to access and load the data. You have terabytes of data to analyze because you had support ticket logs and archives as objects stored in the Amazon S3 bucket. Since many people have visited and left the process over the years, the objects and bucket names have been disorganized. FSx for Lustre allows you to put a file system in front of an S3 bucket or its parts to monitor it. Lustre will take a few minutes to read the content and associated metadata within the bucket via *ImportPath*, and then display it as a file structure. No items are moved by the ImportPath 'read,' and the object data is taken and copied from the bucket only when it is required. With the assistance of security groups, you can also manage access control in FSx for Lustre. It is compatible with PCI and HIPAA. You can use granular performance reporting from CloudWatch, which means that there are potentially more metrics in assistance.

When you have created a file system, you may expect quick access to it from multiple clients or machines running on EC2 or in the local data center. AWS Direct Connect can be used instead of a VPN to ensure the best performance. You may begin with a limited number of processing clients, but when you begin to use the available compute processing, that number may grow drastically. The advantage from a business perspective is that, by using Amazon FSx for Lustre, you would not be limited by I/O or the need to scale and manage Lustre nodes.

AWS RAM

Introduction

AWS RAM (abbreviated as *Resource Access Manager*) enables you to share specific resource types securely with any AWS account or across AWS organization. Eliminating the need to duplicate the resources in all AWS accounts reduces the administrative burden between your accounts. AWS RAM enables the following resources to be shared: Resolvers' Rules, Capacity Reservation, Subnets, Transit Gateways, Aurora's DB Cluster, Traffic Mirror Targets, and License Configuration. In the case of a multi-account environment, you can create resources centrally and share these resources in all the accounts using the Resource Access Manager. You can do this by creating a Resource Share and specifying accounts and the resources in them. The resource manager is accessible from the AWS Services menu. It is available without any additional charges.

Use Case

To understand the sharing process, let us say a user wants to share a subnet from an AWS account to another AWS account. In an architectural environment, it means that when you share a subnet with an account (Q) from a source account (P), then the other account (Q) will be able to view the subnet through the AWS console or AWS CLI. It will then be possible for account Q to launch and run resources (such as EC2 instances) in that subnet. However, there is no administrative control of account Q over these shared resources for subnet deletion or modification. The only control that the account Q has is that it can attach or change the subnet tags. Therefore, for clarification, a subnet is not copied from one account and then introduced to the other. However, only a resource is shared from one account P to another account Q meaning that the subnet will remain as a part of Account P and its VPC.

Key Benefits

Some of the major benefits of AWS RAM are as follows:

Reduces operational cost- RAM can be used to share the centrally procured AWS resources, like License Manager Configurations or subnets with other accounts. There will be no need to duplicate resources in all the AWS accounts in a multi-account environment.

Enhances Safety and Visibility- RAM can use already present IAM permissions and policies to control the usage of shared resources. It offers a detailed overview of shared resources to check logs and set up alarms by integrating with AWS CloudTrail and AWS CloudWatch**.**

Cost optimization- You can use licenses in your organization's several domains for increased usage and optimized cost by sharing resources, like AWS License Manager Configurations across accounts.

AWS CLOUD MIGRATION

The majority of companies became successful when they migrated to the AWS cloud because Amazon has to offer 200 cloud services in many fields such as computing, storage, database, data analytics, AI, and continuous integration. This section focuses on the migration of on-premises workloads to AWS. We will look at the different migration phases, widely used Amazon web services during the migration, and AWS migration strategies 6R's. Let us first explain how businesses are obliged to accept cloud adoption before diving into AWS Cloud Migration details.

Phases Of AWS Migration

Data migration appears to be an easy process for shifting data from one location to another. However, the procedure includes various steps, making it very complicated. Now, the different migration phases are discussed below:

Phase 1

Phase 1 involves the creation of a migration blueprint along with the business framework. You must identify specific business goals in this phase and then assess their benefits. This starts with the creation of a migration project and the establishment of a certain initial background. This approach also takes into account the existing infrastructure and applications, as well as their drawbacks.

Phase 2

In phase 2, a proper migration plan must be developed after understanding the correlation between applications. In order to meet all your business goals, you must understand how the company's IT system works, how the applications are interconnected, and how to decide on a transfer plan. Taking the above into consideration and developing a transfer strategy accordingly will put you in a safe place during the transfer.

Phase 3 & 4

Phases 3 & 4 require applications to be designed, migrated, and validated in an app-oriented manner. It is essential to focus on every application individually when it comes to understanding the entire IT framework. Each application must be developed, moved, and validated by the users. AWS designs, migrates, and validates each application using six common application strategies. When you are convinced that migrating your applications to the cloud is vital to the company and that you have a well-planned migration strategy in mind, you can start the migration process and scale cloud operations accordingly.

Phase 5

In the last phase, you will regularly conduct your business operations on a new framework and incorporate technologies into the working model after migrating data to the cloud. You can use a working model that continuously changes as business activities are scaled on the cloud.

The 6 R’s- Application Migration Strategies

Depending on the architecture of existing applications, the complications in migrating these applications can vary. Amazon thus developed many methods (known as 6R’s) for migrating existing applications. So, let us discuss these strategies one by one.

Rehost-Once your application is up and running, then you can easily rehost it on AWS. This technique is also known as *Lift & Shift*. The services and applications can be lifted from the hosting environment using a third-party exporting tool and shifted onto the cloud.

Replatform-You will have to upgrade and rehost your application if your hosting environment has an outdated version of the application. Replatform is an improvement in the 'Lift and Shift' technique. It involves optimizing the cloud infrastructure to take advantage of without changing the core architecture of the application.

Repurchase- There will be some applications that will not run on the new architecture. You will need to buy a new application for the new architecture in that situation. AWS Marketplace offers a diverse set of services for your business requirements. This method is also termed as 'Drop and Shop.' With this technique, you will be able to adapt, update, and implement the new architecture along with changing the existing one.

Refactor- Refactor can be used in situations where you want to add new features, scale up the existing business model's boundaries, and improve efficiency that is difficult in the current setting. However, this solution is a little costly, so you reconsider your needs. Moving to a service-oriented architecture (SOA) to boost your business would help you in the long run.

Retire- After migrating to AWS, you will be able to tell the difference between useful and useless resources. Thus, you should cut off all the resources that are no longer valuable to the company and create a plan around the new ones so that the additional cost will be reduced. With fewer things to worry about, you must deal with the resources needed for the current business plan.

Retain- When you know which project components should be transferred, any of the techniques mentioned above can be used. Then, according to your business strategy, develop a plan to maintain the applications not ready to be moved to the cloud or recently updated.

We have seen the various techniques, collectively known as the 6 R's, and how to choose them carefully when migrating to AWS. Now that AWS Migration has been discussed let us look at the larger picture.

AWS Cloud Migration Tools and Services

Any technology's successful incorporation and deployment can be determined by the resources you use. AWS provides a number of well-known tools for ensuring a successful migration.

Discovery and Migration Process Tracking

AWS Migration Hub

The centralized dashboard of AWS Migration Hub allows you to track the migration process of applications. You can also use key metrics to read the migration status of your app portfolio and select the most suitable tools based on your migration strategy. You will also get to know the progress of applications individually, irrespective of the tool you are using.

AWS Application Discovery Service

This advanced service is used to plan migration projects that rely on data obtained from on-premise data centers. You can get a better understanding of workloads by using AWS Application Discovery Service, which provides information about servers' usage, setup, and behavior.

Server and Database Migration

AWS Server Migration Service

Thousands of on-premises workloads can be easily and quickly migrated to AWS using the AWS SMS. It enables you to manage, plan, and monitor gradual replications of live server volumes, making large-scale cloud migrations easier to organize.

AWS Database Migration Service

You can transfer the data from and to the most commonly used open-source and commercial databases using this service. The source database stays fully functional throughout the entire migration process, thereby reducing downtime.

VMWare Cloud on AWS (VMC)

VMC on AWS is an on-demand service that lets you run applications in vSphere-based cloud environments while also giving you access to a variety of AWS resources. The virtualization and management software of VMware easily launches and handles VMware workloads in both on-premises and AWS cloud environments.

Data Migration

S3 Transfer Acceleration

This feature accelerates the process of uploading and downloading from your S3 bucket. It uses Amazon's global network of edge locations to send files over long distances between you and an S3 bucket. You will make the most of your available bandwidth regardless of distance or internet conditions.

AWS Snowball

This approach uses secure devices to transfer huge data volumes to and from AWS on a petabyte scale.

AWS Snowmobile

This service is used for transferring very massive data volumes (up to 100PB) to AWS cloud, such as video libraries, image archives, and even the entire data centers.

AWS Direct Connect

With the help of this service, you can connect your company's network to an AWS direct connect site by establishing a direct link. This link can be divided into several virtual interfaces using standard 802.1q VLANs. It allows you to access both public and private tools using the same connection while keeping a distinction between the two environments.

Amazon Kinesis Firehose

It is an entirely controlled service with the help of which you can easily load the streaming data into AWS. It can automatically gather and load the streaming data into Redshift and S3, allowing existing dashboards and business intelligence tools to perform near real-time analytics.

Benefits of AWS Migration

AWS Migration provides an organization with a variety of advantages, including:

- Flexibility

- Enhanced Security
- Business Continuity
- Effective Cost Optimization
- Access to Innovation
- Improved Business Workflows
- Scalability

AWS CHEATSHEET

AWS EC2

- **EC2** is a web-based service that enables developers to deliver safe and resizable cloud computing capacity.
- EC2 belongs to the **compute** domain in Amazon Web Services.
- EC2 uses **Amazon machine images** (AMIs) to configure new instances.
- **AMIs** provides the necessary information to initiate EC2 instance.
- There is five **EC2 instance type**, which depends on the strength of workload.
    - **Compute organized** provide sufficient compute power to the applications that require excessive processing power.
    - **Memory-Optimized** is perfect for applications that need in-memory caches
    - **Accelerated-Optimized** is used for ML, computational finance, voice recognition, and pushover analysis.
    - **Storage Optimized** is more suitable for storage servers like NoSQL database, in-memory or transactional databases, and data warehousing.
    - The **General Purpose** instance type is used for general purpose applications.

- **EC2 instance profiles** are used to get permissions for the EC2 instances, which is like a container for IAM roles.
- Placement groups enable users to figure out the logical placement for their instances.
- Clusters bound the EC2 instances together inside an AZ.
- Partitions are used to spread the instances across the logical partitions.
- **Spreads** are used when there are critical instances that need to be isolated from each other.
- User data is a script that automatically executes when an instance is initiated.
- Metadata is the additional information about EC2 instances such as current public IP address, AMI-ID, and the instance type.
- In the EC2 pricing model, the **On-demand instances** are used for a short duration, uncertain and spiky workloads. When an instance is launched, by default, it will use an on-demand instance.
- **Reserved Instances** are designed for applications that need reserved capacity or have a stable, predictable consumption pattern.
- **Spot instances** in the EC2 pricing model offer a 90% discount for idle servers as compared to on-demand.
- **Dedicated host instances** are used to provide dedicated hardware for an EC2 instance.

AWS Lambda

- Lambda executes the code automatically without needing you to set up or manage the service.
- AWS Lambda will only run the code when necessary and scales it from fewer requests a day to thousands a second.
- You will only be charged for the computing time that you consume.
- It supports:
  - Java
  - C#
  - Go
  - Python
  - Node.js
- To handle the client requests, they are passed to the container, which executes the code provided by the user to satisfy those requests.
- A **Container** is a lightweight, self-executable package containing all the necessary components, such as codes, run times, system tools, system libraries, and any other required settings for Lambda.
- **Lambda pricing** model offers free first one million requests per month.
- A user can have a thousand Lambdas running in parallel by default.

- When you create a lambda, by default, it runs in *No VPC*.
- The delay that occurs when triggering a Lambda function is known as **Cold Start**.
- **Pre warming** allows the servers to run continuously.
- **Pre warming** is a strategy that keeps a server in a running state.

Elastic Beanstalk

- It is a **Platform-As-A-Service** that allows coders to deploy and manage their applications easily without knowing the infrastructure that runs those applications.
- **Elastic Beanstalk** minimizes management complications without limiting the control or choice.
- You need to develop and upload an application in application source bundle format (**Java .war file**) to Elastic Beanstalk and describe a few details about it.
- When you launch this application, its selected and supported platform version is deployed by Elastic Beanstalk.
- The AWS resources that are necessary for your code execution will be created and configured.
- Once you have deployed the environment, you are able to handle the environment and launch new versions of that application.
- It supports the applications built-in **Java Node.js, Python, Go, .NET, Ruby,** and **PHP**.
- It can run dockerized environments in the form of **Dockers** and **Multi-container Dockers**.
- The **Blue/Green** and **In-Place deployments** are a couple of deployment methods in Elastic Beanstalk.
- Scaling and launch issues are not simple to resolve here because of the limited information provided.
- The **worker processes** are not supported by Elastic Beanstalk.

- There is a limited set of software targets for deployment.

AWS S3

- S3 is the most affordable service among all storage options in Amazon.
- It is the object storage that can store and recover data from any platform such as websites, mobile applications, IoT sensors, etc.
- S3 offers 99.99% availability and 99.999999999% durability.
- Different security features such as Bucket Policy, Encryption, and MFA authentication are compatible with S3.
- Amazon S3 storage is accessible and managed with the help of web interfaces.
- All the objects that are uploaded to the S3 bucket are independent with respect to their characteristics and related permissions.
- Basic building blocks of S3 include **buckets** and **objects**.
- An o**bject** is a data, and **buckets** are the folders/containers where the objects are stored in S3.
- **Storage classes** in S3 include:
    - S3 Standard is suitable for low latency networks.
    - S3 Standard IA is utilized for that data that is not frequently accessed.
    - Amazon Glacier is an archival solution in the cloud.
    - S3 One Zone-IA is appropriate for the data that is not regularly viewed and stored in a single AZ.
    - Standard Reduced Redundancy is used for data that is not urgent and can reproduce quickly.

- S3 security ensures encryption by using SSL or TLS by default.
- Whenever a file is stored as an object in Amazon S3, by default, it is stored in several places simultaneously.
- Amazon S3 provides security by denying all the public access to the buckets at its creation.
- *Server-Side (SSE)* and *Client-Side Encryption* are the two encryption types.
- The original data remains static for **Server-Side Encryption** in S3, and Amazon helps the users to encrypt the object data.
- The user encrypts the files and uploads them to S3 in **client-side encryption**.
- S3 provides **consistency** of Reading-After-Writing for the PUTS operations of new objects.
- **Cross-region replication** in S3 allows the asynchronous and automatic copy of data from one destination bucket to another situated in any other AWS Regions.
- **Versioning** in S3 allows a user to set a version for objects and helps to prevent data loss by keeping a record of versions.
- **Lifecycle management** in S3 automates the process of moving objects to different storage classes.
- **Transfer acceleration** in S3 ensures fast and secure sharing of files between the end-users and S3 buckets across long distances.
- A **Pre-signed URL** offers temporary access to a private object.
- **MFA Delete** in S3 ensures that the files do not get deleted when

users access them.

- A two-level authentication is required while setting up MFA Delete on the bucket.
    - By using security credentials.
    - Six-digit code from verified authentication gadget (Google indicator).
- MFA Delete introduces a security layer for the following reasons:
    - Permanent deletion of versioned objects.
    - To modify the versioning state of the state.

Amazon Glacier

- Amazon Glacier is a cheaper storage service in the cloud environment.
- It is built to archive **cold** or **static data**.
- The **archive** is a basic storage unit in Glacier.
- It is not appropriate to store the active data (data involving frequent changes) in Amazon Glacier.
- When you copy the files to S3, Amazon will move the files from S3 to the Glacier.
- The archives in Glacier are not directly accessed.
- There is a retrieval delay of three to five hours for every restore request to the Glacier.
- For long-term Glacier storage, database backups can be moved from tape storage media to the cloud.
- You can download or upload the data from/to the Glacier by using Amazon S3.
- The process of data retrieval model from Glacier involves two steps:
    - The request for Glacier to store a short-term copy of data in Amazon S3.
    - Download the data from S3 to the target location.

Elastic Block Store

- **Elastic Block store** generates new volumes connected to EC2 instances, backed up via snapshots, and provides easy **encryption**.
- It is a **virtual hard drive** in the cloud.
- The data will be stored on AWS EBS servers even when the EC2 instances are disabled.
- A single instance can have 20 EBS volumes where each volume's size ranges from 1GB to 1TB
- EBS volumes are suitable for databases, file systems, and applications that demand granular modifications.
- Volumes exist on EBS, and Snapshots exists on S3.
- There are five volume types, and they are as follows:
  - **General Purpose SSD-**suitable for the general usage without specific requirements
  - **Provisioned IOPS SSD**-Supports fast input and output
  - **EBS Magnetic-**used for long term archival storage
  - **Cold HDD-**it is used for fewer workloads
  - The **Throughput Optimized HDD-**built for the data that is accessed frequently
- The main features of EBS are:
  - **Snapshots**- It creates a copy of the EBS volume's data to S3, where it is stored redundantly in several

    AZ.
    - **EBS-Optimized instances-** They are used for storage workloads that involve short and intensive periods of high I/O activities.

- **Root volume** can be encrypted in EBS while creating an EC2 instance.
- When an EC2 instance is launched, two types of volumes (EBS volumes and the Instance Store Volumes) can get attached to it.
- **EBS volume** is an external block device connected to a single instance, used by users all the time.
- The **Instance Store volume** is temporary (Ephemeral) storage mounted on disks and is physically attached to its host machine.

Elastic File System

- It is a scalable cloud-native **Network File System** used for **Linux-based applications**.
- You can link multiple EC2 instances to one file system with the help of **EFS**.
- It supports Network File System **version 4.1** protocol.
- The storage capacity grows up to petabytes and shrinks automatically depending on the user's data storage**.**
- Great storage solution for applications that run on multiple instances and need parallel data access.
- You can have many EC2 instances in a similar VPC mount to a single EFS volume.
- The EFS is a regional service.
- EFS costs $0.3/GB per month.
- EFS creates multiple targets in all VPC subnets.
- A user needs to install the NFSv4.1 client to mount the EC2 instances with EFS.

Storage Gateway

- Storage Gateway is used for extending and backing up on-premises storage to the cloud.
- It supports Microsoft Hyper-V and VMware ESXi.
- There are three types of gateways.
- **File gateway** supports NFS or SMB protocol to create a mount point for treating S3 as a local hard drive or local file system.
- **Volume gateway** uses the iSCSI block protocol to provide disk volume to our applications.
- **Tape Gateway** is used for backing up Virtual Tape Libraries (VTL) to AWS.
- Tape gateway is the solution to archive user's data in AWS and is pre-configured with tape drivers and a media charger.
- Tape-based backup application setup stores information on virtual tape cartridges.

AWS Snowball

- It uses a physical storage device to transfer huge volumes of data between on-premises and S3.
- It offers user-friendly interfaces to create jobs, track your data, and monitor your job progress.
- **Snowball devices** secure the **data in transit** by means of **AWS KMS**.
- It has a maximum data transfer rate of 250 to 400Mbps, which is equal to 2-5 to 3GB per link.
- 50 and 80TB Snowball data sets are available in the US region, whereas only 50TB models are available in other regions.
- The Snowball costs $200 for a 50TB transfer or $250 for the 80TB version.
- Snowball uses the **Snowball client** and reliable tools to quickly transfer the data to and from AWS services at the petabyte stage.
- Amazon uses an **E Ink label** for Snowball shipment.

AWS SnowMobile

- AWS Snowmobile is an Exabyte service that transmits large volumes of data to AWS.
- This is a shipping container that transfers up to 100PB’s of data per device.
- The transfer of data via snowmobile is quick, affordable, and secure.

- Huge data volumes, including data centres, video libraries, and image repositories, can be transferred easily to the cloud.
- Snowmobile uses several security layers to secure the data, which includes GPS monitoring, alarm tracking, 24/7 video surveillance and specialized security.

AWS RDS

- **RDS** is a cloud computing solution in AWS to provide help in configuration, launch, and scaling of a relational database instance.
- The data in relational databases is stored in tables in rows and columns where queries are applied by using SQL.
- Columns in relational databases show the attributes, whereas the rows in the tables represent the records.
- RDS manages many time-consuming database management tasks, including migration, patching and backup, and recovery, as a completely managed service.
- It provides security and backup for the user databases.
- The **automatic failover** in **multiple AZ's** can also be enabled with synchronous replication of data in RDS.
- Currently, six relational database options available are **MariaDB, Aurora, Oracle, PostgreSQL, Microsoft SQL,** and **MySQL.**
- With the help of the Multi-AZ feature, RDS can provide high availability to maintain the redundant copy of your data in a different location.
- In case the database gets overburdened with the requests, it provides a load balancer that distributes the requests equally.
- It provides two types of automated scaling: Horizontal scaling and Vertical Scaling.
- You can turn on **encryption** at rest for all RDS engines.

- AWS KMS is used for encryption where the default key or KMS key is used to turn it on.
- Automated backups and manual snapshots are the two solutions to perform RDS backups.
- **Automated backups** stores data in S3 without additional charges.
- You can have the **manual snapshots** even when the primary database or RDS gets deleted.
- **Backup data** never restore on top of an existing instance.
- **Restoring Backups** allows point-in-time recovery within a second inside the **retention period**.
- **Multi-AZ deployment** in RDS ensures that a database is available if an AZ becomes unavailable.
- **Read Replicas** in RDS allows running multiple copies of the database.

Aurora

- It is a completely maintained **Postgres** or **MySQL** compatible database service.
- It is designed to scale according to the workload and is very fast by default.
- High-end database availability, high speed is combined with the open-source database's convenience and cost flexibility in Aurora.
- It can run on engines that are compatible with MySQL or Postgres.
- It is a corporate-level database that can scale up to 64 TBs in terms of performance and availability.
- Aurora architecture's working depends on a volume cluster that handles all DB instance data in that specific cluster.
- A **volume cluster** is essentially virtual DB storage that is shared across several AZ.
- This database service offers both **Read Replicas** and **Multi-AZ deployment**.
- Aurora is five times faster than the traditional MySQL, and its Postgres version is three times more efficient.
- The cost is 1/10th the price of other databases that provide similar performance and availability.
- **Serverless mode** is another feature of Aurora in which the database scales down automatically for the lower loads and scale up for the higher loads.

Redshift

- Redshift is a fully managed Petabyte-sized data warehouse.
- Amazon Redshift is used to analyze the huge size of data by using complex SQL queries.
- Online analytical processing (OLAP) involves lengthy transactions for Redshift.
- Redshift is Single-AZ.
- The online transactional processing system (OLTP) involves small transactions for Redshift.
- The redshift **pricing model** is less than 1/10th the cost of most similar services.
- Columnar storage database is used for optimizing analytic query performance.
- In Redshift, OLAP applications fetch multiple records at the same time.
- We deal with a large amount of data in Redshift.
- Redshift uses multiple compression techniques to achieve enough compressions relative to traditional data stores.

- In Redshift configuration, **Dense compute** (DC) is optimized for computing power.
- **Dense storage** (DS) is optimized for storage in Redshift configuration.
- Redshift processing uses massively parallel processing (MPP).
- Redshift **backups** are enabled by default with one day retention time.
- Redshift can replicate the user’s snapshots asynchronously to S3.
- Redshift billing involves compute node hours.
- The data can be encrypted via AES-256 encryption to ensure the security foe data-at-rest.

DynamoDB

- **DynamoDB** is a **key/value** in the document database which manages non-relational databases.
- NoSQL databases crack the concept of keeping the data in tabular form, having rows and columns to have well distributed and well-processed data.
- Amazon Dynamo DB is also known as a Key-value Database as it stores the data as a set of key-value pairs that have a key as an ID.
- Dynamo DB does both key-value store and document-store.
- A key-value store is simply a key and a value.
- Data is structured in a document store.
- DynamoDB is designed to scale the table’s resources to hundreds and thousands of servers spread over several AZ’s.
- Dynamo DB does not compel a user into a specific model of consistency or a data model.
- DynamoDB table must contain a primary key in every table item.
- Dynamo DB guarantees consistent reads and writes.
- **Strongly consistent reads** for Dynamo DB will not return the result until all copies are consistent.

- **Eventual consistent reads** for Dynamo DB is a default function that does not provide a guarantee of consistency.
- DynamoDB is built on SSD to help optimize higher performance.
- DynamoDB supports cross-region replication, transaction, and automatic backups.
- A short key and a partition key (primary key) is made for Dynamo DB in a table structure.

CloudWatch

- **Cloud Watch** is a collection of monitoring services for logging, reacting, and visualizing log data.
- Cloud Watch covers multiple services under one name.
- **CloudWatch logs** are the core service of CloudWatch.
- CloudWatch Log files are stored **indefinitely by default**, and they do not expire.
- Most AWS services are integrated with CloudWatch logs by default.
- **CloudWatch Metrics** represents a scheduled set of data points.
- CloudWatch metrics are predefined.
- CloudWatch Events enable us to respond to our data and take action on that.
- We can make CloudWatch **Custom Metrics** by using or SDK and CLI.
- The higher the frequency for Custom Metrics, the greater it will cost.
- We can get **high-resolution metrics** is through custom metrics.
- CloudWatch does not monitor **memory utilization**.
- **CloudWatch Alarms** triggers a notification when a defined threshold is breached.
- CloudWatch dashboards allow us to create a dashboard that depends on CloudWatch metrics.

- It is easy to visualize data using the CloudWatch dashboard.
- CloudWatch agent is used to collect the information about the empty space on the server by default.
- Systems manager run command supports the installation of the CloudWatch agent.
- CloudWatch agent may be pre-installed on Amazon Linux-1 and Linux-2.

CloudTrail

- CloudTrail helps in identifying the person responsible for actions.
- It can be used to monitor the API calls between the services and the account actions of AWS.
- CloudTrail **monitoring** collects records for 90 days in Event History.
- A **trail** can be set to log in to all regions.
- CloudTrail will utilize the history of all the operations.
- CloudTrail does not have the **event history** as in GUI.
- Log File Validation will inform us how much we can rely on the log.
- CloudTrail can deliver its events to cloud watch.
- CloudTrail Logs can be encrypted.
- Operational challenges can be overhauled by using the call history of AWS API developed by CloudTrail.
- Events in the cloud trail include:
    - Data events. (Currently entertains S3 and Lambda services).
    - Management events. (Enabled by Default)
- You can easily detect abnormal activities in your AWS account by enabling the CloudTrail insights.
- CloudTrail preserves the API events in an immutable and secure manner.

- 90% of events in the cloud trail are management events.

CloudFormation

- **CloudFormation** is a simple mechanism for providing a number of AWS resources.
- It creates and designs the infrastructure and applications without performing manual operations.
- CloudFormation helps to focus more on the running application rather than waste time on resource management.
- We need to generate a **template** for any EC2 or RDS instance and then upload it to the CloudFormation portal.
- CloudFormation portal provides the resources and manages them based on a user-defined template.
- CloudFormation templates allow you to monitor all the changes employed to the infrastructure.
- When a template is reviewed and checked, cloud formation is responsible for the organized and consistent handling of the dependencies.
- The **Changeset** in CloudFormation enables you to visualize the impact of anticipated changes to the operating resources before implementation.
- A **stack** is a set of AWS resources in cloud formation managed as a single unit.
- In cloud formation, stacks are updatable, and they are not limited.
- With cloud formation templates, we can add almost all applications and the resources that we might need.

- CloudFormation templates are portable and reusable.
- CloudFormation is used to create the stack or the resources defined in the Template.
- The template and the stack collectively complete the cloud formation and thereby provide and manage the resources.
- Cloud Formation Designer helps to create or update the existing template.
- Nine main objects to build cloud formation templates include:
    - Format version -identifies the capabilities of the template depending on the version number)
    - The description -helps make random remarks about the template.
    - Metadata -includes the template's details and the template's resources.
    - Parameters section -an optional object used to customize a template.
    - Mapping-maps a key to the corresponding set of named values.
    - Conditions -includes the statements that define a resource creation or property definition.
    - Transform -an optional object explains one or more transforms that we can use in the cloud formation template.
    - Resources -declares the AWS resource required for the stack.

    - Output -an optional object that helps to declare the importable outputs into other stacks).

- **Resource Attribute** in CloudFormation template adds attributes to a resource and helps to control extra behaviour and associations.
- Few resources attributes of cloud formation template include:
    - **Depends On**- create a sequence for resource deletion.
    - **Update Policy** -manage and replace the instance's updates in an auto-scaling group.
    - **Creation Policy** -delay the resource configuration actions before the stack creation.
    - **Metadata** -associate the resources with the structured data.
    - **Deletion Policy** -preserves the back up of resources when the stack gets deleted.

- We can create, update, or delete a stack in cloud Formation.
- A cloud Formation stack holds all the necessary resources to execute Web-applications.
- In cloud formation, we can create a hierarchy of stacks by **nesting** a stack with another stack.
- **Window stacks** in cloud formation help us to use a remote desktop to access the stack and install software and update its configurations.
- Windows stacks can run on windows server instances.

- CloudFormation Stack sets are the updated form of stack level implementation.
- We can specify the list of users having access to the cloud formation template with IAM.
- **Stack Policies** are applied to the users having access to the cloud formation service.
- **IAM** provides a layer of control over the cloud formation service.

Virtual Private Cloud

- **Virtual Private Cloud** enables its users to build a virtual networking system inside the cloud in their own logically isolated area.
- VPC runs on default hardware.
- A VPC is like a standard network system that you can have in your data centre.
- Every Single VPC is entirely customizable and logically separated from the rest of the cloud's virtual networks.
- VPCs are region-oriented, and they do not span regions.
- One can create five VPCs per region.
- There is a maximum of 200 Subnets per VPC.
- Each instance that resolves an instance's private IP address is provided with a private DNS hostname.
- Creating a VPC does not cost anything.

- While creating a VPC, **DNS hostname** is not enabled by default.
- When an EC2 instance runs with DNS hostnames enabled, it will give you a public IP.
- The two types of VPCs include:
    - **Default VPC**-you can instantly launch an instance into it.
    - **Non-default VPC**- you need to generate a non-default VPC and configure it.

- Default VPC is available for every region.
- When you create a default VPC, it comes along with the following configurations such as:
    - CIDR Block size /16 with attached default subnet,
    - main Route Table,
    - internet gateway,
    - security group, NACL and DHCP options
- The range of IP addresses in a VPC is known as a subnet.
- A VPC consists of **multiple AZs**, but the *subnet* lies in each AZ.
- We cannot initiate any instance until the subnets are present in the VPC.
- VPC contains **20 netmasks** with up to **4,096 subnet addresses.**
- The resources that are linked with the internet use *Public Subnet*, whereas the resources that do not need an internet connection use *Private Subnet*.
- By default, instances launched in the VPC cannot communicate with the user's network. Users can link VPCs to an existing data centre using **VPN access**.
- **VPC Peering** enables its users to link one VPC to another across a direct network route using private IP addresses.
- If the VPCs reside in the same region, a peering connection can be established between them.
- **Internet Gateway** allows internet access to the VPC.
- Internet Gateway creates a Target in a route table for internet

traffic routing.

- It also performs network address translation on instances with public IPv4 addresses.
- **IGW** requires two steps:
    - **IGW id**
    - **Destination assignment**.
- **Route Tables** contains the route that possesses the Internet Gateway that would grant internet access to a user.
- **Custom Route-table** notifies the Internet-gateway to transmit the internet traffic towards the public subnet.
- **Main Route-table** does not let in the internet traffic since the private-subnet associates with the default Route-table.
- **Bastions Hosts** has restricted security controls that allow restrictive inbound links from reliable IP addresses.
- Bastions use **RDP** or **SSH** protocols.
- Direct Connect establishes dedicated network connections from the user’s on-premise locations to AWS.
- VPC Endpoints allow us to link a VPC with other VPC Endpoints and AWS services privately.
- The two types of VPC Endpoints include:
    - Gateway-Endpoints
    - Interface-Endpoints
- Interface-Endpoints provide Elastic Network Interface to connect

multiple AWS services with a private IP address.

- **Gateway-Endpoint** targets routes in the Route-table which use the traffic intended for AWS supported service.
- Interface Endpoints cost money, whereas Gateway Endpoints are free.
- **VPC flow logs** enable users to get incoming and outgoing IP traffic information from their network interfaces within VPC.
- Flow Logs can be turned on at the network interface level, subnet level, and VPC level.

Network Access Control List

- NACL is a security layer that serves as a **virtual firewall** at the subnet level.
- When we create a VPC, we get NACL by default.
- NACL holds **internal** and **external rules**.
- The internet traffic gets **denied** by NACL.
- Rule number determines the order of evaluation.
- We use increments of 10 or 100 in NACL rule numbers.
- The subnets must be associated with NACLs.

Transit Gateway

- **AWS Transit Gateway** (**TGW**) connects thousands of on-premise networks and VPCs together.
- The Transit Gateway serves as a central hub within the network
- TGW configuration simplifies multiple peering connections and decreases general connectivity from local locations to the VPCs.

Private Link

- **Private Link** offers a private connection between AWS VPCs, an on-premises application, and AWS services.
- It resembles Direct Connect, where it creates a private connection to the AWS cloud while Direct Connect connects the user's on-premises environments to AWS.
- Private Link secures the network traffic from the VPC environments of the users residing in AWS.
- For Private Link, the endpoints are created inside the user’s VPC utilizing IP addresses and ENIs.
- AWS Private Link allows you to create service connectivity from service provider VPC to consumer VPC in a safe and scalable way.

AWS Global Accelerator

- **AWS Global Accelerator** enhances the performance and availability of an application for both local and global users.
- Global Accelerator provides two static **IPv4** addresses, which serve as a fixed entrance for the application endpoints in a single or multiple AWS regions.
- AWS Global Accelerator tracks the health of application endpoints.
- Global Accelerator supports a few Endpoints, including EC2 Instances, Elastic IP, Internet-facing ALB, NLB and ALB.
- Global Accelerator improves the Route for all consumers and decreases the delays.
- To use of Global Accelerator, we need to:
    - Establish an accelerator that provides a couple of Anycast static IP addresses.
    - Establish a listener for the port range or protocol and port.
    - Establish an Endpoint-group for each region where he wants to direct the traffic.
    - Introduce an Endpoint such as an ALB for every Endpoint-group.
- Global Accelerator encourages gradual fluctuation of traffic among endpoints and regions.

Security Group

- It operates as a virtual firewall controlling incoming and outgoing traffic to protect EC2 instances.
- The SG contains **inbound and outbound rules** to filter out the incoming and outgoing traffic from that EC2 instance.
- All the traffic is blocked by default for security groups unless a rule explicitly permits it.
- Security groups ensure security at the instance level so they can be attached to multiple instances.
- Multiple instances across multiple subnets can belong to a single security group.
- Security groups do not care about subnets.
- Anything within the security group can gain access to inbound traffic.
- An instance can belong to multiple security groups where rules are non-restrictive.
- We can nest multiple security groups as one.
- We can have a maximum of 10,000 security groups in a region.
- By default, 2,500 security groups are allocated to users.
- The number of security groups on an instance depends on the number of ENIs attached to that security group.
- Each security group has 60 inbound and outbound rules and 16 safety groups per ENI.

NAT

- **NAT** is a method of remapping one IP address space into another.
- NAT devices are used to connect the private subnet instances with the AWS services or the internet.
- NAT helps in getting outbound access to the internet.
- We can utilize the NAT to make the addresses more appropriate for communication if there are two networks with discrete network addresses.
- NAT in AWS is introduced in two ways:
    - **NAT instance**
    - **NAT gateways**

- You need to initiate NAT instance in a public subnet as it requires internet.
- **NAT instance** cannot access the internet in a private subnet.
- NAT gateways set up the EC2 instance for the users.
- **NAT gateways** possess redundant instances for the user.
- There is a web server in the public subnet for NAT and a database server in the private server.

- We can send request from the database server to the internet, but we cannot send requests from the internet to the database server via NAT instance.
- NAT instance residing in a private subnet serves as an intermediate translator for the server.

- NAT gateway can easily replace NAT instance.
- We use NAT Gateway, where the NAT instance becomes a bottleneck.
- We can place multiple NAT instances in several AZs.
- We can create multiple NAT gateways in various availability zones to achieve higher availability.

Route 53

- It is the most scalable and available AWS domain name system.
- With **Route 53**, we can:
    - Register and managed **domains**.
    - Create various **Record Sets** on a domain.
    - Implement complicated traffic flows.
    - Continuously monitor the records through **health checks** and resolve VPCs outside of AWS.
- Route53 can manage the name servers to route users to their web applications.
- In Route 53, domain requests are resolved by a global network of DNS servers.
- By creating **Record Sets**, we can create the link for Route53 to point to certain resources.
- We can create Record Sets from different types of records.
- Route53 is good for monitoring the performance and health of web servers, web applications, and other resources.
- The **Route 53 Resolver (.2 resolver)** can route DNS queries between your VPC and network.
- Route 53 Resolver offers automated DNS resolution in the user VPCs and entertains DNS queries for domain names by default.
- **Traffic Flow** is a visual editor that allows users to create complex routing configurations within Route 53 easily.

- Seven types of Routing policies include:
    - Simple Routing Policy -helps to provide one or more IP addresses.
    - Weighted Routing Policy -helps to split the traffic depending on various assigned weights.
    - Latency Based Routing Policy -helps to direct the traffic depending on the regions with the lowest possible network latency.
    - Failover Routing Policy -helps to create an active and passive setup.
    - Geolocation Routing Policy -helps to direct the traffic based on geographical location.
    - Geo-proximity Routing Policy -helps to provide a favour to a specific location with certain boundaries.
    - Multi-Value Answer Policy -helps to provide one or more IP addresses with health checks.
- Various record types in AWS Route53 are:
    - **SOA records**- domain admins provide additional information about a domain.
    - **NS records**-Top-level domain servers use them to route traffic to the DNS server holding the DNS records authority.
    - **Address records**- the most fundamental type of DNS records that converts the domain name directly into an IP address.

- **CNAME records** - to resolve one domain name to another instead of IP addresses such as www route.
- One can consider **A** Records as IP addresses and CNAMES as domain names.
- **Alias records-** for mapping the resource record sets in your hosted area to the websites.
- **TTL** is the duration of time the DNS record uses to resolve servers or the user's local machines.

Auto Scaling Groups

- **Auto Scaling Groups** (**ASGs**) allow you to set scaling rules to automatically launch or shut down additional EC2 instances.
- The automatic scaling occurs via health check replacements, scaling policies, or capacity settings.
- **Capacity Settings** are used without any configurations by setting values of ‘Desired Capacity,' 'Min,' and ‘Max.'
- **Health Check** is another way of auto-scaling to occur with Auto-Scaling Groups.
- **ELB** and **EC2 health checks** are the two forms of health checks.
- There are three types of scaling policies:
    - **Target Tracking Scaling Policy**- it scales based on a specific metric at a target value.
    - **Simple Scaling Policy**- It scales when the alarm breaks out.
    - **Scaling Policy with Steps**- it scales based on the changing alarm value.
- **Launch configurations** can be set with the help of an ASG.
- In Launch Configuration, when an ASG launches an EC2 instance, it contains the information about the configurations required for an instance launch.
- The Launch Configurations with versioning are known as **Launch Template.**

Elastic Load Balancers (ELB)

- **Elastic Load Balancers** distributes incoming app traffic to multiple targets *such as* IP addresses, containers, Lambda functions, or EC2 instances.
- ELB includes physical hardware and virtual software, accepting incoming traffic and distributing it across multiple targets.
- ELB uses specific rules to distribute the load.
- Three components for ELB traffic flow include:
    - **Listeners**- react to the incoming traffic and test it against specific ports.
    - **Rules**- decide the actions against the traffic.
    - **Target Groups**- help gather the EC2 instances in logical groups and lies between the ASG and the ELB.
- ELB has three types including:
    - **ALB** (Application Load Balancers)
    - **CLB** (Classic Load Balancers)
    - **NLB** (Network Load Balancers)
- **ALB** balances HTTPs and HTTP traffic.
- ALB works at the Application layer of the OSI model.
- Web application firewall can be added to ALB.
- **Request Routing** is a specified feature for ALB.
- Request Routing allows you to apply the rules to an incoming request and then reroute the traffic.

- **NLB** balances TCP and UDP traffic.
- NLB works at **the transport layer** of the OSI model.
- NLB supports Multiplayer video games.
- **CLB** balances HTTP or TCP traffic.
- CLB cannot balance both traffic simultaneously.
- CLB performs one function at a time.
- CLB and NLB can perform cross-zone balancing**.**
- **Sticky Sessions** is a modified load balancer, which associates user sessions to a certain EC2 instance.
- Sticky Sessions is useful for the locally stored information on a single instance.
- **X-Forwarded-For-Header** identifies the user’s originating IP address while connecting to a web server using a load balancer or HTTP proxy.
- **ELB Health Checks** take the traffic away from an unhealthy instance and direct it to healthy instances.
- **Cross-Zone Load Balancing** is available for network load balancers and classic load balancers.

CloudFront

- **CloudFront** produces cached copies of a user website at different edge locations around the world.
- CloudFront is Amazon’s worldwide **content delivery network** (**CDN**) with massive capacity and scale.
- It has been designed for static and dynamic objects and video delivery.
- The three main components for CloudFront are:
    - An **Origin** is a place where the original files are located**.**
    - **Distribution -**set of edge locations that determine the actions of cache content**.**
    - **Edge Locations** -a server near the actual user**.**
- **Origins** describe the Amazon S3 bucket, HTTP server, or an EC2 instance from where CloudFront fetches the content.
- CloudFront uses a Signed URL to access the origin.
- One may use a **CloudFront OAI** to limit access to CloudFront from the S3 bucket.
- Origin search for the header depending on its value.
- **Distribution** is a set of edge locations that act as an indicator of the original content.
- Specifying the origin is the first thing in setting up distributions.
- CloudFront distributions have one-to-many behaviours with one default Behavior.

- The ability to perform all the configurations is known as **Behavior.**
- **Invalidation** in CloudFront configures the TTL manually so that you do not need to wait until the TTL expires.
- CloudFront allows us to restrict access depending on the requester’s geographical location at no additional cost.
- You can either blacklist or whitelist a location based on security reasons to only distribute the content to a few regions.
- **AWS WAF** is considered as a front-line defence for AWS ALB, API Gateway, and CloudFront.
- AWS WAF is the seventh layer of application protection created in each edge location of the CloudFront worldwide.
- **Lambda@Edge** involves lambda functions that override the Behavior of requests and responses that flow to and from a CloudFront.
- There are four Lambda@Edge functions: **the origin request, viewer request, the viewer response,** and **the origin response.**
- For CloudFront protection, it provides **a signed URL** to temporarily access the private cached objects.
- **Origin Identity Access (OAI)** is a virtual user identity that allows the CloudFront distribution to fetch private objects.
- The Pre- signed URLs belong to S3, and Signed URLs belong to CloudFront.

IAM (Identity and Access Management)

- It is a web server for managing access to the AWS resources securely.
- **IAM** allows you to authenticate or limit a certain set of users/resources to access the AWS account.
- A few elements that complete *IAM workflow* are:
    - Principle -is an application or a person that can request an operation /actions on AWS resources.
    - Authentication -the process of verifying the identity of the principle that wants to enter into the AWS network.
    - Request -it carries information about the actions of the principle and of three types: Put, Get or Delete request.
    - Authorization –the process where IAM uses data from **the request context** to find similar policies determines whether to accept or reject.
    - Actions - the actions that you are going to take, such as view, create, edit the content, and delete a resource.
    - Resource -An object that exists in service is called a resource.
- There are four components of IAM, which include:
    - End-User -an entity that you create in an AWS environment.
    - Group -a collection of IAM users is known as the IAM group.

    - Role - it is a set of permissions that determines whether the actions are allowed or denied by an entity in the AWS console.
    - Policy - clear documents in JSON format that manages access and sets permission to AWS resources. There are two main categories of IAM policies:
        - *Online Policy* -merged into a single user, group, or rule directly**.**
        - *Managed Policy* -default policy that you can link with multiple entries in the AWS account**.**
- IAM allows us to create unique usernames and passwords for individual users or resources and representative access.
- You can apply restrictions on requests using IAM.
- IAM supports Multi-Factor Authentication.
- Additional users, groups, and policies are available for free.
- You can reset or rotate a password locally with the IAM password policy.

KMS (Key Management Service)

- KMS is used to create and control cryptographic keys and monitor their usage across different platforms and applications.
- With KMS, you can securely manage keys across AWS services and hosted applications.
- It uses envelop encryption and maintains keys in Hardware Security Modules (HSMs).
- In Envelop encryption, the data is encrypted and stored locally in the AWS service or application along with the key.
- With KMS, the administrators can create, control, and delete the keys.
- AWS KMS is reachable within AWS Identity and Access Management.
- KMS provides centralized control over encryption keys to define the user data.
- KMS is a **multi-tenant** hardware security module.
- There is a **master key** for the data encryption key, which acts as an additional layer of security.
- KMS uses **CMKs** to encrypt the data keys and data keys to encrypt the plain text.
- CMK (Customer Master Key) is a logical representation of a master key.
- You can define, create, and provide a list of master keys using

AWS KMS.

- It is possible to import cryptographic material into the AWS KMS master keys.

Amazon Cognito

- **Amazon Cognito** is a decentralized way of managing authentication using credentials.
- One may consider Amazon Cognito as a Log-on/Log-in integration with the user applications and other social identity providers.
- Amazon Cognito gathers the user's profile attributes into the folders known as **user pools**.
- Amazon Cognito combines data sets with identities and stores encrypted data as key or value pair.
- You can store up to 20MB of data with a limit of 1 MB of individual data set.
- A developer can set Amazon Cognito to take a stream of events.
- Identity Provider is a reliable provider for the user identity.
- The user who receives short-term AWS credentials with limited access will have a new Cognito ID.
- There are two key components in Amazon Cognito:
    - **User Pools**-a decentralized record of the users to handle different actions.
    - **Identity Pools** provides temporary AWS credentials to the users.
- **Cognito Sync** allows the synchronization of user data and preferences across all devices with a single line of code.
- It uses push notifications such as SNS to push those updates and

synchronized data to the devices.

SQS (Simple Queue Service)

- It is a completely managed **queuing service** for scaling and setting apart the architectures of microservices, serverless and distributed systems.
- It is an **application integration** service that acts as a bridge of communication and connects isolated applications with the help of messaging and queues.
- SQS is a **pull-based** service.
- It uses a **queue**, which is temporary storage for messages that need to be processed.
- SQS can be used for:
    - Decoupling microservices,
    - Send tasks between different parts of your system,
    - Distribute workloads from a central node to worker node,
    - Schedule batch jobs.
- In SQS, message size ranges from 1byte to 256KB.
- **Message Retention** is the duration in which SQS keeps a message in the queue before dropping it.
- You can set this message retention from 60 seconds to a maximum of 14 days.
- The two different queue types are:
    - **Standard Queue** provides a nearly unlimited number of transactions per second while transacting. It is similar to messages and guarantees to deliver the message at least once.
    - **FIFO (First In, First Out**) has a limit of 300

transactions per second for a single API action and guarantees that the messages stay in their order without duplicates.

- **Visibility Timeout** is the duration defined for a queue object that is hidden from the queue and other users once it is fetched and processed by a consumer.
- **Polling** is the way of retrieving messages from the queue.
- **Short Polling** and **Long Polling** are the two ways of polling in SQS.

SNS (Simple Notification Service)

- It is a completely managed and highly available **pub/sub messaging service** to decouple microservices-based architectures, serverless and distributed systems.
- Using **SNS**, you can send notifications and subscribe via email, text messages, Lambdas, web books, mobile notifications, and SQS.
- In the pub/sub system environment, a sender is also known as a **publisher**.
- A publisher sends a message to an event bus instead of sending it to the receiver directly.
- The **Event bus** categorizes the messages into different groups/classes.
- A receiver is known as a **subscriber** that subscribes to these groups.
- Publishers do not know about their subscribers.
- **SNS Topic** allows you to combine multiple subscriptions together.
- A Topic is an access point or a communication channel where you can subscribe.
- You can encrypt your topic with the help of KMS.
- The delivery of messages is not guaranteed by SNS.
- If the message sending is unsuccessful for some reason, SNS will delete the message.

- **Subscriptions** are created on a topic.
- The **Application as a Subscriber protocol** can be used to send alert messages to the mobile endpoints.

Command Line Interface

- Multiple AWS services are managed by **AWS CLI** from a command line by using scripts**.**
- CLI is installed via **Python Script**.
- It also enables the users to control the behaviours and manage all AWS services and resources.
- One can install CLI on Linux, macOS, Windows, and Docker container.
- You can organize CLI in profiles.
- Direct access is provided by CLI to the public APIs of AWS services.
- To use CLI, you can enable programmatic access.

Software Development Kit or SDK

- It is a set of API libraries that enable us to integrate AWS services into the applications.
- By using a common programming language, SDK enables a user to manage several AWS services.
- It is used with APIs attached to the programming languages to simplify the AWS service for the intended application.
- You can organize SDK in profiles.

AMI (Amazon Machine Image)

- AMI **is a template** for the configuration of a new EC2 instance, which provides essential data to launch it.
- It is a packaged environment with the necessary components and software configuration for setting up an EC2 instance.
- While configuring an instance, you have to specify the source AMI.

- Many instances can be configured from a single machine image.
- One may use several machine images to set up the instances if Instances with different configurations are required.
- An AMI consist of:
    - Template
    - Permission to Launch
    - Block Device Mapping
- The AMIs can be created from existing EC2 instances.
- Hundreds of AMIs are available in AWS Community AMI and AWS Marketplace, which can be chosen and scanned.
- These AMIs have unique *AMI IDs*.
- Shared AMIs are available for public use and developed by the developers.

API Gateway

- It is a completely managed service to build, maintain, publish, monitor, and secure the APIs in your cloud environment.
- It acts as a bridge between the users of your API and API's backend services while handles the HTTP requests to the API endpoints and directs these requests to the accurate backends.
- Multiple software applications can interact with one another using **API Gateways**.
- With API Gateways, you can create, update, and delete documentation related to all portions of your API.
- With **API Lifecycle**, several version of your API can be managed at the same time.
- You can manage the backend traffic with API Gateway.
- The access keys provide security to manage API access.
- You can define strategies and manage API definitions by using API Gateway.
- There are few steps for Configuring APIs:
    - Create **several resources** as you will have several endpoints.
    - Apply **HTTP methods** to your resources.
    - **Publish the API** by setting up different stages such as:
        - **Staging** for developers
        - **Prod** for production
        - **QA** for quality assurance

    - When a stage is created, AWS automatically creates a **invoke URL** which is the target endpoint.
    - Finally, **set up** the endpoints.
- **API Gateway caching** reduces the load on an API by caching the outcomes of common endpoints.
- Users need the **Same-Origin Policy** when they are exposed to the *XSS attacks.*
- **Cross-Origin Resource Sharing (CORS)** is used to deal with the problem of the same-origin policy.

Amazon Kinesis

- Amazon Kinesis is a scalable and durable data streaming services for delivering and processing data in real-time.
- It is used for data streaming in **stock prices, gaming, geospatial, social media,** and **the Clickstream.**
- Kinesis has four types of streams:
    - Firehose Delivery Streams
    - Data Streams
    - Video Streams
    - Data Analytics
- The Firehose Delivery Stream is similar to but simpler than the data streams.
- The data in Firehose Stream instantly vanishes from the queue as soon as it is consumed, due to which data will not persevere.
- You can choose only one consumer in Firehose.
- Data consumption in Firehose does not require writing any code.
- **Data Streams** consist of producers, shards, and customers.
- A consumer is dedicated to sending data to Redshift, Dynamo DB, S3, and then EMR.
- By default, kinesis data will remain there for 24hours after it has entered the stream.
- **Video Streams** are used for streaming video data.
- **Data Analytics c**onsists of an input and an output stream.

- Data Analytics uses two streams (Firehose or data stream) which make it a bit expensive.

ElasticCache

- **Elastic Cache** is a **Cache In-Memory data store**, a short-term storage location where you have instant access to data.
- It is used to manage the **Memcached or Redis caching** services.
- The process of storing data in a cache is known as **Caching**.
- **A cache** is a short-term storage location used for instant retrieval.
- When the data gets stored in RAM, it is known as an **In-Memory data store.**
- Both the cache and the in-memory data store together form the Elastic cache.
- Elastic Cache is highly volatile.
- It frequently identifies queries and keeps these queries in the cache to attain extra performance.
- **Redis** and **Memcached** are the two in-memory data storage options available in Elastic Cache.
- Redis can support geospatial support, take transactions, snapshots, pub/sub, and replication.
- It is ideal for tracking unreadable notifications, leaderboard, and cached data in real-time.
- For caching HTML fragments, Memcached is used as a simple value/key store.
- Memcached is considered more rapid than Redis for smaller fragments.

Amazon ECS

- **Amazon ECS** is a cloud computing service that handles the containers.
- With the help of ECS, a developer is able to run an application in a cloud environment by configuring an environment to execute the code.
- ECS allows you to launch **containers** on EC2 server containers.
- For the **serverless mode**, containers can be launched on **Fargate** by using ECS.
- ECS is an alternative tool for Swarm, Kubernetes, Mesos, or Docker.
- ECS eliminates the need to configure, administer, and execute the orchestration layer for creating an efficient and rigid clustering system.
- There are no additional charges for AWS clients to use ECS.
- The components of an application must be designed and implemented to operate in containers in order to launch applications on Amazon ECS.
- Amazon ECS scales the application and constantly handle the availability of containers.
- The structure of Amazon ECS consists of the following terms:
- The instantiation of a task definition is known as a **Task**.
- **Task Definition** of Amazon ECS describes an application.
- The positioning of tasks in Amazon ECS is a responsibility of the

ECS **Scheduler**, particularly on the ECS Cluster.

  - The temporary tasks are handled by **the Batch job scheduler**.
  - Permanent tasks are handled by a **service scheduler.**

- The ECS **Service** means to carry out a specified number of tasks on a cluster simultaneously.
- A **Cluster** is a logical resource grouping on which you can execute tasks.
- The **Container Agent** serves as a proxy for the Amazon ECS and the EC2 resource task.

EFA (Elastic Fabric Adaptor)

- It is a network interface designed for EC2 instances.
- It speeds up the **ML** and **HPC** workloads requiring high-level communication between the nodes.
- EFA has all **ENA functions** and optional **OS-Bypass** features.
- An HPC application that uses EFA communicates with the **Libfabric API** using **NCCL** or **Intel MPI**.
- It is compatible with the commonly used interfaces, APIs, and libraries to migrate HPC applications to AWS.
- EFA was built to work with the existing network of AWS infrastructure and has the scaling ability to fulfil the application requirements.
- Only one EFA per instance can be attached.
- One cannot route EFA OS-bypass traffic.
- A security group must be attached to EFA, which permits all the traffic to and from SG.

ENA (Enhanced Network Adaptor)

- To guarantee **enhanced networking** for the EC2 instances, **ENA** is used as a new-generation network interface with the linked drivers.
- This custom network interface was built to ensure **low latencies, higher throughput**, and **high packet per second performance** (**PPS**) on EC2 instances.
- ENA is intended to operate with **modern processors** and can use several virtual CPU technologies of these modern processors (such as processors of **X1 instances**).
- ENA provides 24Gbps network bandwidth for specific instance types.
- ENA is used for improved networking for the current windows EC2 generation.
- ENA scales and increases with increasing bandwidth abilities and raising CPU count.
- Suitable for applications with higher bandwidth and low latency.
- It is supported on devices with **multi-queue interfaces**.
- Only **HMV instance** types are supported by ENA.
- **RSS-Receive Side Scaling** is also supported by ENA.

HPC on AWS

- The capacity to process data and carry out quadrillions of complicated calculations in a second is known as **High-Performance Computing.**
- **A supercomputer** is an example of HPC.
- HPC solutions use thousands of compute nodes operating together for completing the tasks, which is referred to as **Parallel Processing**.
- It is a base for industrial, societal, and scientific progress.
- HPC solutions can be used for
    - Testing new products,
    - Streaming a live event of sports,
    - Analyzing stock trends,
    - Monitoring a developing storm.
- The **AWS Cloud** is a low-cost, effective approach and emerging tech for HPC users' requirements due to its flexibility in scaling the compute nodes based on the app needs and customers' budgets.
- **AWS EC2** offers strong compute and storage resources on-request via virtualization at the hardware level, and it also aims to be globalized.
- The **EFA** allows the app execution with high-level communication between the nodes on AWS.

- AWS offers HPC storage options to support practical HPC workloads.
- To support an object interface, **Amazon S3** is a highly durable and scalable platform for storing HPC applications.
- **FSx for Luster** can be used to launch and execute a file system that offers access to the data in sub-millisecond for HPC.
- AWS created Parallel clusters to build complicated HPC systems.
- Using job scheduling and managed resource provisioning of **AWS Batch** for EFA, you will be able to perform distributed HPC and ML workloads.

Amazon FSx

- It provides a network file storage solution that is more reliable, durable, business-oriented and secure.
- It works perfectly with the current windows environment and applications.
- There are two forms of FSx:
    - **FSx for Lustre**
    - **FSx for Windows File Server**
- FSx for Windows offers a fully maintained MS-Windows file system on AWS.
- FSx for Windows is built on SSD storage and uses SMB protocol for sharing files.
- FSx for Lustre is used to provide a high-performance file system built for workloads that demand a lot of computation.
- FSx for Lustre is used for ML and HPC applications.
- FSx for Lustre has the ability to process large data sets.
- The performance can increase up to millions of IOPS, sub-milliseconds latencies, and hundreds of Gbps of throughput.
- The replication feature known as **the distributed-file-system** (**DFS**) is used by FSx in order to maintain its presence in a **multi-AZ** environment.
- The Amazon FSx can achieve 2 Gbps Throughput and enterprise-level performance.
- Amazon FSx can be integrated with the workloads such as **NTFS**

in windows, **DFS**, or **active directory**.

- FSx can be used for the following use cases:
    - Software development platforms.
    - Data analytics application.
    - Shift and Lift applications.
    - Home Directories.
    - Media workflows include stream, transcode, and process.

AWS Cloud Migration

- **Data migration** is a very simple process in which the data is transferred from one location to another.
- Data migration includes multiple stages.
    - One must specify business goals and evaluate the types of advantages while creating Migration Blueprint.
    - Creating a migration strategy allows users to be located accurately during the transfer.
    - Applications must be designed, migrated, and validated in an app-oriented manner.
    - One will frequently work on the latest platform and implement innovation into a working model by migrating applications to the cloud.
- Amazon developed many methods (known as **6R's**) for migrating existing applications.
- You must select from 6 R's wisely during AWS migration.
    - **Rehost-** rehost your application on AWS when it is ready and running.
    - **Replatform-** upgrade and rehost your application if the hosting environment has an outdated version of the application.
    - **Repurchase-** buy a new application for the new architecture.
    - **Refactor** - add new features, scale up the existing business model's boundaries, and improve efficiency.
    - **Retire-** cut off all the resources that are no longer

    valuable to the company and create a plan around the new ones.
  - **Retain-** develop a plan to maintain the applications not ready to be moved to the cloud or recently updated.
- AWS provides a number of well-known tools for ensuring a successful migration:
- **Discovery and Migration Process Tracking**
  - **AWS Migration Hub-** allows you to track the migration process of applications.
  - **AWS Application Discovery Service-** used to plan migration projects that rely on data obtained from on-premise data centres.
- **Server and Database Migration**
  - **AWS Server Migration Service-** Thousands of on-premises workloads can be easily and quickly migrated to AWS.
  - **AWS Database Migration Service -** You can transfer the data from and to the most commonly used open-source and commercial databases.
  - **VMWare Cloud on AWS (VMC)-** run applications in vSphere-based cloud environments while also giving access to many AWS resources.
- **Data Migration**
  - **S3 Transfer Acceleration-** accelerates the process of uploading and downloading from your S3 bucket.
  - **AWS Snowball-** uses secure devices to transfer huge data volumes to and from AWS on a petabyte-scale.

- **AWS Snowmobile-** transfers very massive data volumes (up to 100PB) to AWS.
- **AWS Direct Connect-** connects
  your company's network to an AWS direct connect site by establishing a direct link.
- **Amazon Kinesis Firehose-** easily loads the streaming data into AWS.

References

1. (n.d.). *Amazon Elastic Compute Cloud User Guide for Linux Instances*. Retrieved from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ ug.pdf
2. (n.d.). *Amazon Elastic Compute Cloud User Guide for Windows Instances*. Retrieved from https://docs.aws.amazon.com/AWSEC2/latest/WindowsG wg.pdf
3. (n.d.). *Amazon Relational Database Service User Guide Amazon Relational Database Service: User Guide*. Retrieved from https://docs.aws.amazon.com/AmazonRDS/latest/UserGui ug.pdf#Welcome
4. (n.d.). *Amazon Simple Storage Service User Guide*. Retrieved from

https://docs.aws.amazon.com/AmazonS3/latest/userguide/
userguide.pdf#UsingServerSideEncryption

5. (n.d.). *AWS CloudTrail User Guide*. Retrieved from https://docs.aws.amazon.com/awscloudtrail/latest/userguic ug.pdf#cloudtrail-event-reference-user-identity
6. (n.d.). *AWS General Reference Reference guide Version 1.0*. Retrieved from https://docs.aws.amazon.com/general/latest/gr/aws-general.pdf#aws_tagging
7. (n.d.). *AWS Identity and Access Management User Guide*. Retrieved from https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-ug.pdf#introduction
8. (n.d.). *Financial Services Industry Lens AWS Well-Architected Framework*. Retrieved from https://docs.aws.amazon.com/wellarchitected/latest/financ services-industry-lens/wellarchitected-financial-services-

industry-lens.pdf#cost-optimization-pillar?cv=1

9. . (n.d.). *Amazon EMR Management Guide*. Retrieved from https://docs.aws.amazon.com/emr/latest/ManagementGuic mgmt.pdf#emr-overview

10. . (n.d.). *AWS Elastic Beanstalk Developer Guide*. Retrieved from https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/aw dg.pdf

11. . (n.d.). *AWS IoT Greengrass Developer Guide, Version 1*. Retrieved from https://docs.aws.amazon.com/greengrass/v1/developergui dg.pdf

12. 2020. (2020). Retrieved May 9, 2020, from Amazon Web Services, Inc. website: https://aws.amazon.com/about-aws/whats-new/2020/

13. Abbasi, A. (2020). *AWS Certified Data Analytics Study*

Guide. https://doi.org/10.1002/9781119649489

14. Advanced Amazon Virtual Private Cloud (Amazon VPC). (2018). *AWS® Certified Advanced Networking Official Study Guide*, 57–92. https://doi.org/10.1002/9781119549000.ch3
15. Alteen, N., Fisher, J., Gerena, C., Gruver, W., Jalis, A., Osman, H., Roth, M. (2019). *AWS® Certified Developer Official Study Guide*. https://doi.org/10.1002/9781119549451
16. Amazon Aurora | MySQL PostgreSQL Relational Database | Amazon Web Services. (2021). Retrieved March 30, 2021, from Amazon Web Services, Inc. website: https://aws.amazon.com/rds/aurora/?aurora-whats-new.sort-by=item.additionalFields.postDateTime&aurora-whats-new.sort-order=desc
17. Amazon Cognito Identity Pools (Federated Identities) -

Amazon Cognito. (2021). Retrieved February 5, 2021, from Amazon.com website: https://docs.aws.amazon.com/cognito/latest/developerguid identity.html

18. Amazon Cognito user pools - Amazon Cognito. (2021). Retrieved February 3, 2021, from Amazon.com website: https://docs.aws.amazon.com/cognito/latest/developerguid user-identity-pools.html
19. Amazon EBS volume types - Amazon Elastic Compute Cloud. (2018). Retrieved February 6, 2021, from Amazon.com website: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ volume-types.html
20. Amazon Elastic Block Store (Amazon EBS) - Amazon Elastic Compute Cloud. (2021). Retrieved January 19, 2021, from Amazon.com website: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/

21. Amazon Elastic File System (EFS) | Cloud File Storage. (2021). Retrieved March 10, 2021, from Amazon Web Services, Inc. website: https://aws.amazon.com/efs/

22. Amazon Simple Storage Service (S3) — Cloud Storage — AWS. (2015). Retrieved April 27, 2020, from Amazon Web Services, Inc. website: https://aws.amazon.com/s3/faqs/

23. Amazon Virtual Private Cloud (Amazon VPC) and Networking Fundamentals. (2018). *AWS® Certified Advanced Networking Official Study Guide*, 15–56. https://doi.org/10.1002/9781119549000.ch2

24. Anvy. (2016). CISSP - PDF Free Download. Retrieved January 5, 2021, from qdoc.tips website: https://qdoc.tips/cissp-pdf-free.html

25. Aoanan, J. (2021, March 2). AWS Glue | Noise. Retrieved April 10, 2020, from Getoto.net website: https://noise.getoto.net/tag/aws-glue/

26. AWS PrivateLink and VPC endpoints - Amazon Virtual Private Cloud. (2021). Retrieved March 18, 2021, from Amazon.com website: https://docs.aws.amazon.com/vpc/latest/userguide/endpoir services-overview.html

27. AWS Snow Family | Physical Devices to Migrate Data into and out of AWS | Amazon Web Services. (2021). Retrieved January 8, 2021, from Amazon Web Services, Inc. website: https://aws.amazon.com/snow/

28. Barr, J. (2016, September 2). improvements | Noise | Page 10. Retrieved April 11, 2020, from Getoto.net website: https://noise.getoto.net/tag/improvements/page/10/

29. Beach, B., Armentrout, S., Bozo, R., & Tsouris, E. (2019). *Pro PowerShell for Amazon Web Services*. https://doi.org/10.1007/978-1-4842-4850-8

30. Botmetric | Cloud Cost Management, Cloud Security Compliance & DevOps Automation Platform for AWS

and Azure Cloud. (2017, June 22). Retrieved February 4, 2021, from Botmetric website: https://www.botmetric.com/blog/monthly-aws-cloud-usage-cost/

31. Chauhan, S., Devine, J., Halachmi, A., Lehwess, M., Matthews, N., Morad, S., & Seymour, S. (2018). *AWS® Certified Advanced Networking Official Study Guide*. https://doi.org/10.1002/9781119549000

32. Creativity in Intelligent Technologies and Data Science. (2019). In A. G. Kravets, P. P. Groumpos, M. Shcherbakov, & M. Kultsova (Eds.), *Communications in Computer and Information Science*. https://doi.org/10.1007/978-3-030-29750-3

33. Data Storage. (2020). *AWS Certified Data Analytics Study Guide*, 93–142. https://doi.org/10.1002/9781119649489.ch3

34. Domain Name System and Load Balancing.

(2018). *AWS® Certified Advanced Networking Official Study Guide*, 155–206. https://doi.org/10.1002/9781119549000.ch6

35. Elastic Load Balancing - Amazon Web Services. (2019). Retrieved March 15, 2021, from Amazon Web Services, Inc. website: https://aws.amazon.com/elasticloadbalancing/?whats-new-cards-elb.sort-by=item.additionalFields.postDateTime&whats-new-cards-elb.sort-order=desc
36. Getting started with Amazon Redshift - Amazon Redshift. (2021). Retrieved February 1, 2021, from Amazon.com website: https://docs.aws.amazon.com/redshift/latest/gsg/getting-started.html
37. Gillis, A. S. (2017). Amazon Web Services (AWS). Retrieved January 19, 2021, from SearchAWS website:

https://searchaws.techtarget.com/definition/Amazon-Web-Services

38. Hello, Storage. (2019). *AWS® Certified Developer Official Study Guide*, 85–174. https://doi.org/10.1002/9781119549451.ch03

39. How to Understand SOP: Same-origin Policy Whitepaper | Netsparker. (2016). Retrieved April 2, 2021, from Netsparker.com website: https://www.netsparker.com/whitepaper-same-origin-policy/

40. Isaac Computer Science. (2021). Retrieved March 25, 2021, from Isaac Computer Science website: https://isaaccomputerscience.org/concepts/gcse_sys_secor

41. Jakóbczyk, M. T. (2020). *Practical Oracle Cloud Infrastructure*. https://doi.org/10.1007/978-1-4842-5506-3

42. Load balancer types - Amazon Elastic Container Service.

(2021). Retrieved March 15, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonECS/latest/develope balancer-types.html

43. McLaughlin, B. (2019). *AWS Certified Solutions Architect Practice Tests*. https://doi.org/10.1002/9781119558453
44. McLaughlin, B. (2021). AWS Certified SysOps Administrator Study Guide. Retrieved April 6, 2021, from Perlego.com website: https://www.perlego.com/book/1382746/aws-certified-sysops-administrator-study-guide-associate-soac01-exam-pdf
45. Mishra, A. (2019). *Machine Learning in the AWS Cloud*. https://doi.org/10.1002/9781119556749
46. Mitra Group. (2020, September 22). Eight types of AWS storage services explained - Mitra Innovation. Retrieved

September 7, 2020, from Mitra Innovation website: https://mitrai.com/tech-guides/eight-types-of-aws-storage-services-explained/

47. Nadon, J. (2017). *Website Hosting and Migration with Amazon Web Services*. https://doi.org/10.1007/978-1-4842-2589-9

48. Patel, J. M. (2020). Introduction to Cloud Computing and Amazon Web Services (AWS). *Getting Structured Data from the Internet*, 85–133. https://doi.org/10.1007/978-1-4842-6576-5_3

49. Piper, B., & Clinton, D. (2019). *AWS Certified Solutions Architect Study Guide*. https://doi.org/10.1002/9781119560395

50. Santana, G., Neto, M., Sapata, F., Muñoz, M., Moraes, A. M. S. P., Morais, T., & Goldfarb, D. L. (2021). *AWS Certified Security Study Guide*. https://doi.org/10.1002/9781119658856

51. Scott, S. (2016, February 10). S3 Lifecycle Policies, Versioning & Encryption: AWS... Retrieved January 2, 2021, from Cloud Academy website: https://cloudacademy.com/blog/s3-lifecycle-policies-versioning-encryption-aws-security/

52. Service Requirements. (2018). *AWS® Certified Advanced Networking Official Study Guide*, 345–362. https://doi.org/10.1002/9781119549000.ch11

53. Shackelford, A. (2015). *Beginning Amazon Web Services with Node.js*. https://doi.org/10.1007/978-1-4842-0653-9

54. Sharma, V. (2018). *The Cloud-Based Demand-Driven Supply Chain*. https://doi.org/10.1002/9781119477792

55. Stateless Application Patterns. (2019). *AWS® Certified Developer Official Study Guide*, 663–795. https://doi.org/10.1002/9781119549451.ch14

56. Surianarayanan, C., & Chelliah, P. R. (2019). Essentials of Cloud Computing. In *Texts in Computer Science*.

https://doi.org/10.1007/978-3-030-13134-0

57. Understanding how IAM works - AWS Identity and Access Management. (2021). Retrieved March 27, 2021, from Amazonaws.cn website: https://docs.amazonaws.cn/en_us/IAM/latest/UserGuide/i structure.html

58. Updated whitepaper available: Encrypting File Data with Amazon Elastic File System | Amazon Web Services. (2021, February 23). Retrieved March 10, 2021, from Amazon Web Services website: https://aws.amazon.com/blogs/security/updated-whitepaper-available-encrypting-file-data-with-amazon-elastic-file-system/

59. Vayngolts, I., Korobkin, D., Fomenkov, S., & Kolesnikov, S. (2019). The Software and Information Complex Which Uses Structured Physical Knowledge for Technical Systems Design. *Communications in Computer*

and Information Science, 42–51.
https://doi.org/10.1007/978-3-030-29750-3_4

60. Venner, J. (2009). *Pro Hadoop*.
https://doi.org/10.1007/978-1-4302-1943-9

61. What is a transit gateway? - Amazon Virtual Private Cloud. (2021). Retrieved March 24, 2021, from Amazon.com website:
https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html

62. What is Amazon API Gateway? - Amazon API Gateway. (2021). Retrieved March 2, 2021, from Amazon.com website:
https://docs.aws.amazon.com/apigateway/latest/developer

63. What is Amazon Aurora? - Amazon Aurora. (2021). Retrieved January 14, 2021, from Amazon.com website:
https://docs.aws.amazon.com/AmazonRDS/latest/Aurora

64. What is Amazon CloudFront? - Amazon CloudFront.

(2021). Retrieved March 8, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonCloudFront/latest/D

65. What Is Amazon CloudWatch Events? - Amazon CloudWatch Events. (2021). Retrieved February 7, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonCloudWatch/latest/e
66. What Is Amazon CloudWatch Logs? - Amazon CloudWatch Logs. (2021). Retrieved February 10, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonCloudWatch/latest/l
67. What Is Amazon CloudWatch? - Amazon CloudWatch. (2021). Retrieved February 18, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonCloudWatch/latest/m
68. What Is Amazon DynamoDB? - Amazon DynamoDB. (2021). Retrieved January 18, 2021, from Amazon.com

website:

https://docs.aws.amazon.com/amazondynamodb/latest/dev

69. What is Amazon Elastic Container Service? - Amazon Elastic Container Service. (2021). Retrieved January 5, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonECS/latest/develope

70. What is Amazon ElastiCache for Redis? - Amazon ElastiCache for Redis. (2021). Retrieved January 21, 2021, from Amazon.com website: https://docs.aws.amazon.com/AmazonElastiCache/latest/r ug/WhatIs.html

71. What Is Amazon Kinesis Data Analytics for SQL Applications? - Amazon Kinesis Data Analytics for SQL Applications Developer Guide. (2021). Retrieved March 5, 2021, from Amazon.com website: https://docs.aws.amazon.com/kinesisanalytics/latest/dev/w is.html

72. What Is Amazon Kinesis Data Firehose? - Amazon Kinesis Data Firehose. (2021). Retrieved March 4, 2021, from Amazon.com website: https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

73. What Is Amazon Kinesis Data Streams? - Amazon Kinesis Data Streams. (2021). Retrieved March 4, 2021, from Amazon.com website: https://docs.aws.amazon.com/streams/latest/dev/introducti

74. What Is Amazon Kinesis Video Streams? - Amazon Kinesis Video Streams. (2021). Retrieved March 4, 2021, from Amazon.com website: https://docs.aws.amazon.com/kinesisvideostreams/latest/d is-kinesis-video.html

75. What is Amazon Route 53? - Amazon Route 53. (2021). Retrieved March 16, 2021, from Amazon.com website: https://docs.aws.amazon.com/Route53/latest/DeveloperGu

76. What is Amazon Simple Queue Service? - Amazon Simple Queue Service. (2021). Retrieved March 27, 2021, from Amazon.com website: https://docs.aws.amazon.com/AWSSimpleQueueService/l
77. What is Amazon SNS? - Amazon Simple Notification Service. (2021). Retrieved March 30, 2021, from Amazon.com website: https://docs.aws.amazon.com/sns/latest/dg/welcome.html
78. What is Amazon VPC? - Amazon Virtual Private Cloud. (2021). Retrieved March 21, 2021, from Amazon.com website: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html
79. What is AWS CloudFormation? - AWS CloudFormation. (2021). Retrieved February 14, 2021, from Amazon.com website: https://docs.aws.amazon.com/AWSCloudFormation/latest

80. What Is AWS CloudTrail? - AWS CloudTrail. (2021). Retrieved February 24, 2021, from Amazon.com website: https://docs.aws.amazon.com/awscloudtrail/latest/userguic user-guide.html

81. What is AWS Database Migration Service? - AWS Database Migration Service. (2021). Retrieved March 1, 2021, from Amazon.com website: https://docs.aws.amazon.com/dms/latest/userguide/Welcor

82. What is AWS Direct Connect? - AWS Direct Connect. (2021). Retrieved March 9, 2021, from Amazon.com website: https://docs.aws.amazon.com/directconnect/latest/UserGu

83. What is AWS Global Accelerator? - AWS Global Accelerator. (2021). Retrieved March 14, 2021, from Amazon.com website: https://docs.aws.amazon.com/global-accelerator/latest/dg/what-is-global-accelerator.html

84. What is AWS Key Management Service? - AWS Key Management Service. (2021). Retrieved February 8, 2021, from Amazon.com website: https://docs.aws.amazon.com/kms/latest/developerguide/o
85. What is AWS Lambda? - AWS Lambda. (2021). Retrieved January 10, 2021, from Amazon.com website: https://docs.aws.amazon.com/lambda/latest/dg/welcome.h
86. What is AWS RAM? - AWS Resource Access Manager. (2021). Retrieved February 19, 2021, from Amazon.com website: https://docs.aws.amazon.com/ram/latest/userguide/what-is.html
87. What is AWS Storage Gateway? - AWS Storage Gateway. (2021). Retrieved January 27, 2021, from Amazon.com website: https://docs.aws.amazon.com/storagegateway/latest/userg
88. What is Elastic Load Balancing? - Elastic Load

Balancing. (2015). Retrieved March 11, 2021, from Amazon.com website: https://docs.aws.amazon.com/elasticloadbalancing/latest/uis-load-balancing.html

89. What is the AWS Command Line Interface? - AWS Command Line Interface. (2021). Retrieved February 15, 2021, from Amazon.com website: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
90. What is the AWS Management Console? - AWS Management Console. (2021). Retrieved February 16, 2021, from Amazon.com website: https://docs.aws.amazon.com/awsconsolehelpdocs/latest/gwhats-new.html
91. Whitepaper, A. (n.d.). *AWS Storage Services Overview*. Retrieved from website: https://docs.aws.amazon.com/whitepapers/latest/aws-

storage-services-overview/aws-storage-services-overview.pdf

92. Zomaya, A. Y., & Sakr, S. (Eds.). (2017). *Handbook of Big Data Technologies*. https://doi.org/10.1007/978-3-319-49340-4