**AWS – Concepts**

1. Introduction to Cloud Computing Concepts and AWS (Amazon Web Services):
2. Introduction to EC2, EBS, EFS and Amazon FSX
3. Introduction to IAM (Identity and Access management), Cloud Watch, AMI (Amazon Machine Image)
4. Elastic Load Balancer and Auto Scaling, Route 53
5. VPC (Virtual Private Cloud)
6. S3 (Simple Storage Service)
7. Database Services
8. Cloud Formation and App Services
9. AWS Lambda, Elastic Beanstalk, AWS Ops work and API (Application Programming Interface) Gateway
10. SNS (Simple Notification Services)
11. SQS (Simple Queue Services)
12. SES (Amazon Simple Email Service)
13. **Cloud Computing**

Cloud Computing is sharing a Hardware/Software resource virtually over a network.

**Cloud Service Models:**

**IaaS – Infrastructure as a Service:** Ex: AWS, Rackspace, MS Azure

**PasS – Platform as a Service:** Ex: AWS Elastic Beanstalk, Heroku etc

**SaaS – Software as a Service:** Ex: Google drive, Google docs, MS Office 365 etc

**Cloud deployment Models:**

* Public Cloud (AWS, MS Azure, Google platform, IBM blue mix etc)
* Private Cloud (HPE, VM ware, Redhat Open stack, Dell EMC etc)
* Hybrid Cloud

**Virtualization:**

Virtualization is a process of creating a virtual environment of something. It can be a Hardware or storage device or a network resources.

**Advantages of Virtualization:**

* Itsaves the space.
* It saves the operating cost.
* It enables the easy management of our data center.

**Types of Virtualization:**

* Hardware virtualization AWS\*
* Application virtualization
* Server virtualization
* Network virtualization
* Desktop virtualization

**Hypervisor:**

Hypervisor is a hardware virtualization technique that allows to run multiple operating systems on a single host system at the same time.

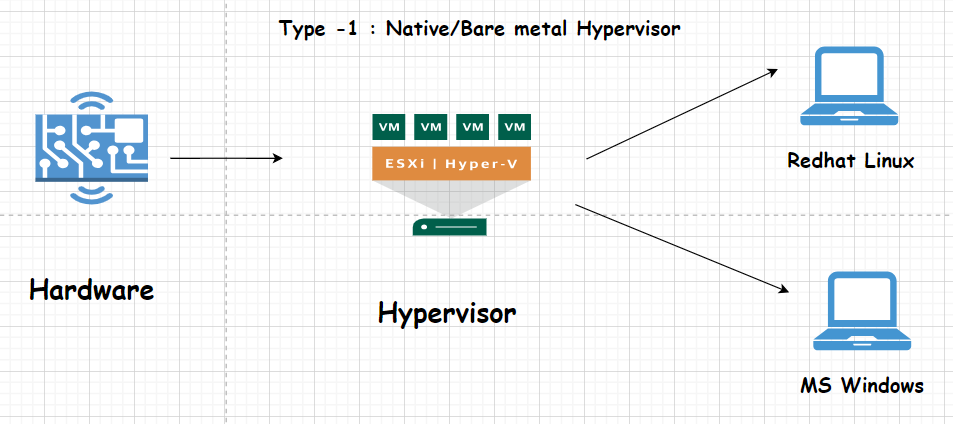
It is also called as virtual machine manager.

**Types of Hypervisor:**

We have 2 types of Hypervisors.

**Type-1 Bare metal/native hypervisor: \*AWS**

It runs directly on the hardware, there is no operating system between hardware and hypervisor. So, it is more efficient due to direct access to the hardware, CPU, memory network and physical storage. However, we should have a dedicated machine to run the instructions.



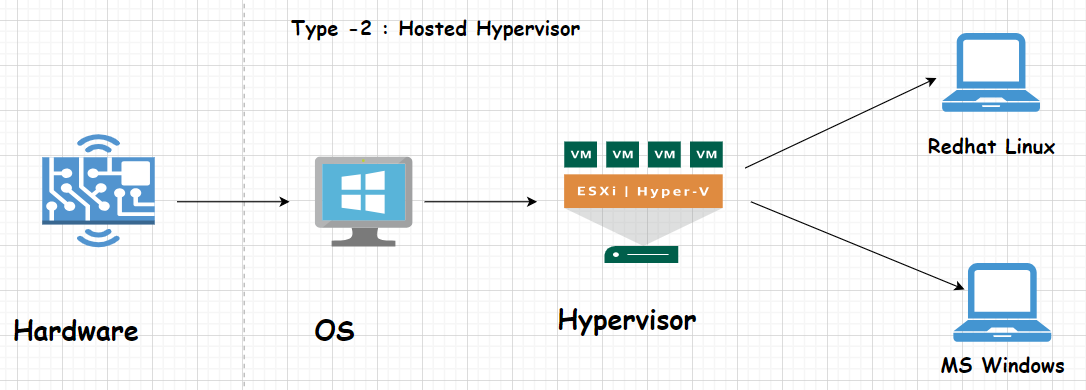
* **Xen – Hypervisor (Type-1) software used by AWS for virtualization.**

**Type-2 Hosted hypervisor:**

It doesn’t run over the hardware directly, it runs over the application as it has an Operating System.

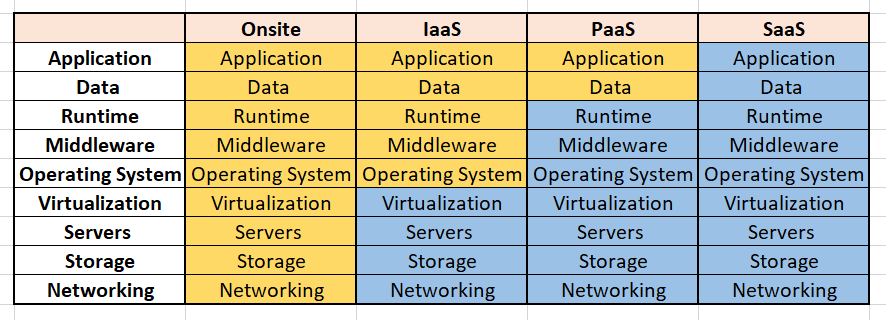
It is easy to set up.

It is useful for the engineers & Security analysis to check the malware and newly deployed applications.



The below tables shows what are the services that are managed by the customer and the services managed by the CSP (Could Service Provider) in Onsite, IaaS, PaaS, SaaS.

In Onsite, customer has to manager everything as shown. But, in SaaS, CSP manages everything.



**\* Customer manages \* CSP manages**

**1. IaaS (Infrastructure as a Service)**

**Description:**  
IaaS provides virtualized computing resources over the internet. It allows you to rent IT infrastructure like servers, storage, and networking on a pay-as-you-go basis. This means you don’t have to invest in physical hardware.

**Examples:**

* **Amazon EC2:** You can launch virtual servers and scale them up or down as needed.
* **Google Compute Engine:** Offers virtual machines and storage options for your applications.

**2. PaaS (Platform as a Service)**

**Description:**  
PaaS provides a platform that allows developers to build, deploy, and manage applications without worrying about the underlying infrastructure. It includes tools, middleware, and database management services, making it easier for developers to focus on writing code.

**Examples:**

* **Google App Engine:** A platform for building web applications without managing servers.
* **AWS Elastic Beanstalk:** Lets you deploy and manage applications in various languages without worrying about the underlying hardware.

**3. SaaS (Software as a Service)**

**Description:**  
SaaS delivers software applications over the internet on a subscription basis. Users access these applications through a web browser, eliminating the need to install or maintain software locally.

**Examples:**

* **Google Workspace (formerly G Suite):** Provides applications like Google Docs, Sheets, and Gmail online.
* **Salesforce:** A customer relationship management (CRM) tool that runs in the cloud.

**4. Onsite Cloud Services (Private Cloud)**

**Description:**  
Onsite cloud services refer to private cloud environments hosted on your own premises. This setup allows you to have dedicated resources for your organization, giving you more control over security and compliance.

**Examples:**

* **OpenStack:** An open-source software platform that allows you to create and manage your own private cloud.
* **VMware vSphere:** A virtualization platform that enables you to create a private cloud using your own hardware.

**Summary**

* **IaaS:** Rent virtual servers (e.g., EC2).
* **PaaS:** Develop applications without managing servers (e.g., Google App Engine).
* **SaaS:** Use software online (e.g., Google Workspace).
* **Onsite Cloud Services:** Build a private cloud with dedicated resources (e.g., Open Stack).

1. **Cloud Introduction to EC2, EBS, and EFS, FSX**

**EC2 (Elastic Compute Cloud):** Amazon EC2 (Elastic Compute Cloud) is a service that provides virtual servers in the cloud. Using amazon EC2 eliminates the need to invest in hardware upfront, allowing you to develop and deploy applications more quickly. Elasticity is a level at which a system is able to adapt to workload changes by provisioning and de-provisioning resources such that the resources meet the current demands as much as possible.

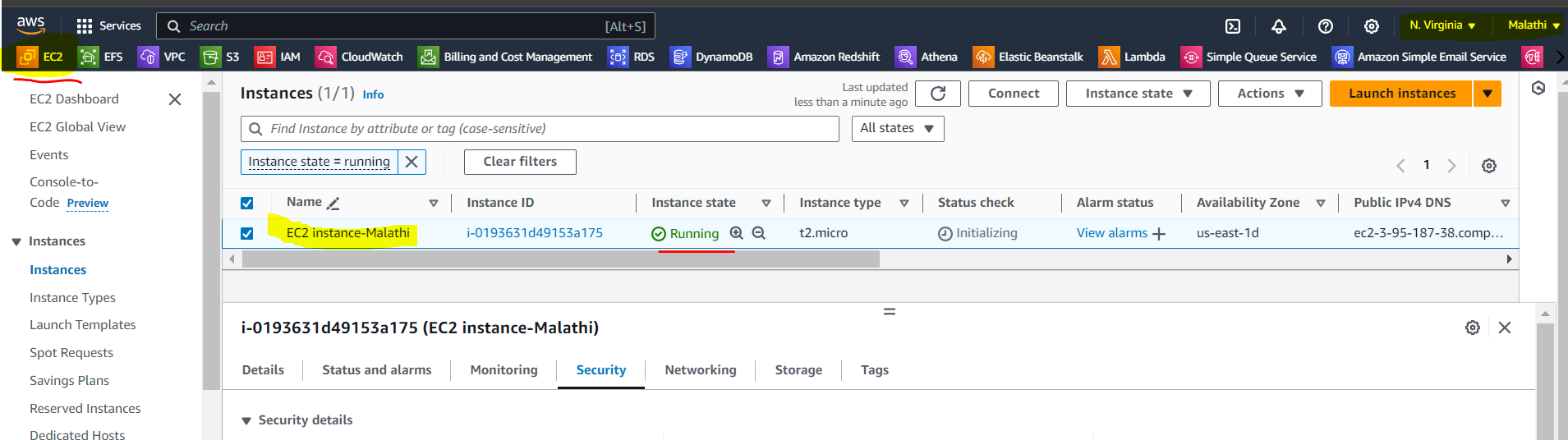
**Features of EC2:**

* Instances are virtual computing environments
* AMI (Amazon machine images) are preconfigured templates for your instances that package the bits you need for your server(including the operating system and additional software).
* Instance types are different configurations of CPU, memory and storage & networking capacity for your EC2 instances.
* Using key pairs, you can secure login information for your instances. (AWS stores the public key & you store the private key in a secured pace.)

**Instance Types:**

* General purpose, T2 Burstable, Memory optimized, Storage optimized, Accelerated Computing, Compute-optimized.

EC2 Created below in the AWS platform.



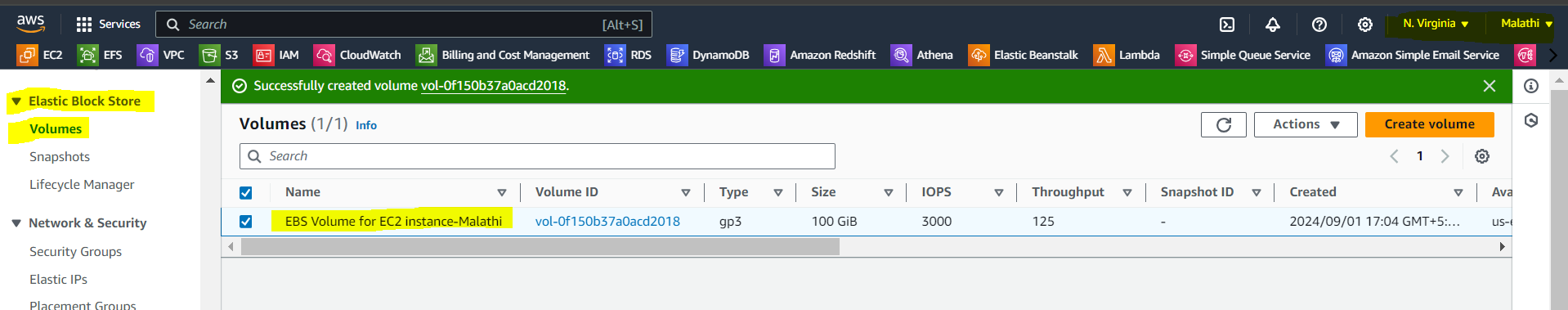
**EBS (Elastic Block Storage):** It is similar to RAM.Amazon EBS is a service from AWS that provides storage volumes with Amazon EC2instances. EBS provides storage volumes that function like virtual hard drives. These volumes can be attached to EC2 instances to store data. Unlike file storage systems, block storage divides data into blocks and stores them, which allows for efficient and flexible storage management.

One EBS can be attached to one EC2. No 2 EC2 instances can have the same EBS volume.

### Features of EBS:

### ****Flexible****: You can create EBS volumes of different sizes and performance characteristics based on your needs. For example, you might choose a volume optimized for high-speed access (SSD) or a volume optimized for cost-efficiency (HDD).

1. **Attach and Detach**: You can attach an EBS volume to an EC2 instance to use it. If you need to, you can detach it and attach it to another instance. This flexibility allows you to move data easily between instances.
2. **Snapshots**: EBS allows you to take snapshots of your volumes. These snapshots are backups that are stored in Amazon S3 (Simple Storage Service). You can use snapshots to restore data or create new EBS volumes.
3. **Performance Options**: EBS offers different performance options, such as General Purpose SSD, Provisioned IOPS SSD (for high-performance needs), and Cold HDD (for infrequent access).
4. **High Availability**: EBS volumes are automatically replicated within the Availability Zone (a data center location) to provide durability. They are designed to handle failures and ensure that your data is safe.

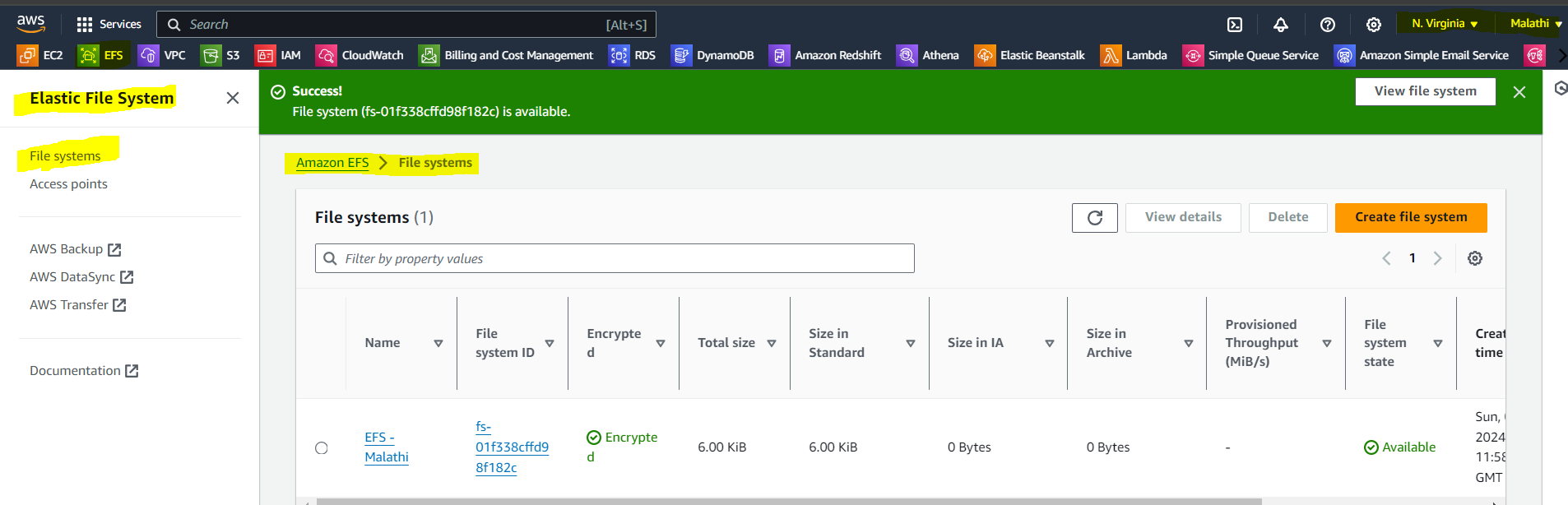


Click on the link below for more details on

Amazon EBS: <https://docs.aws.amazon.com/ebs/?icmpid=docs_homepage_storage>

**EFS (Elastic File Storage):** EFS is afile level storage service that provides a shared elastic file system with virtually limitless scalability. EFS is a highly available storage system that can be used by multiple servers at the same time.

It automatically scale up or scale down the storage depends on the work load (heavy/less workloads).



**Features:**

1. **Easy to Use**: You can quickly create and configure EFS through the AWS Management Console or APIs. It integrates easily with EC2 instances.
2. **Shared Access**: Multiple EC2 instances can access the same EFS file system simultaneously. This is useful for applications that need to share files among multiple instances.
3. **High Availability**: EFS is designed to be highly available and fault-tolerant. It replicates your data across multiple Availability Zones (data centers) within a region, which helps ensure your data is always accessible.
4. **Elastic**: EFS automatically adjusts its capacity as you add or remove files. There’s no need to provision storage ahead of time.
5. **Performance**: EFS provides high performance for a wide range of workloads, from small web apps to large data-intensive applications.

**How It Works:**

1. **Create a File System**: You start by creating an EFS file system through the AWS Management Console.
2. **Mount to EC2**: You then mount the EFS file system to your EC2 instances. This makes it appear as if it’s a local drive on those instances.
3. **Use It**: You can then store and access files on the EFS file system just like you would with any other file system. Multiple EC2 instances can read and write to the same file system concurrently.
4. **Manage**: EFS handles scaling and maintenance, so you don’t have to worry about the underlying infrastructure.

**When to Use EFS:**

* **Shared Storage**: When you need a shared file system that multiple EC2 instances can access simultaneously.
* **Dynamic Workloads**: For applications that need to scale storage automatically without manual intervention.
* **High Availability**: When you need durable storage with built-in redundancy.

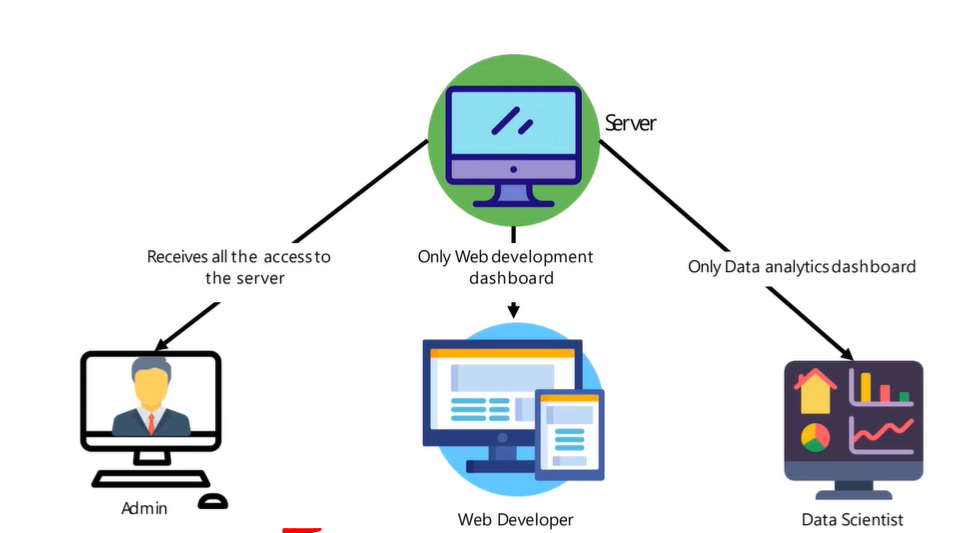
**FSX (File server):** Amazon Fsx is fully managed third-party file system solution. It makes use of SSD storage to provide fast performance with low latency.

**Use cases of FSX :**

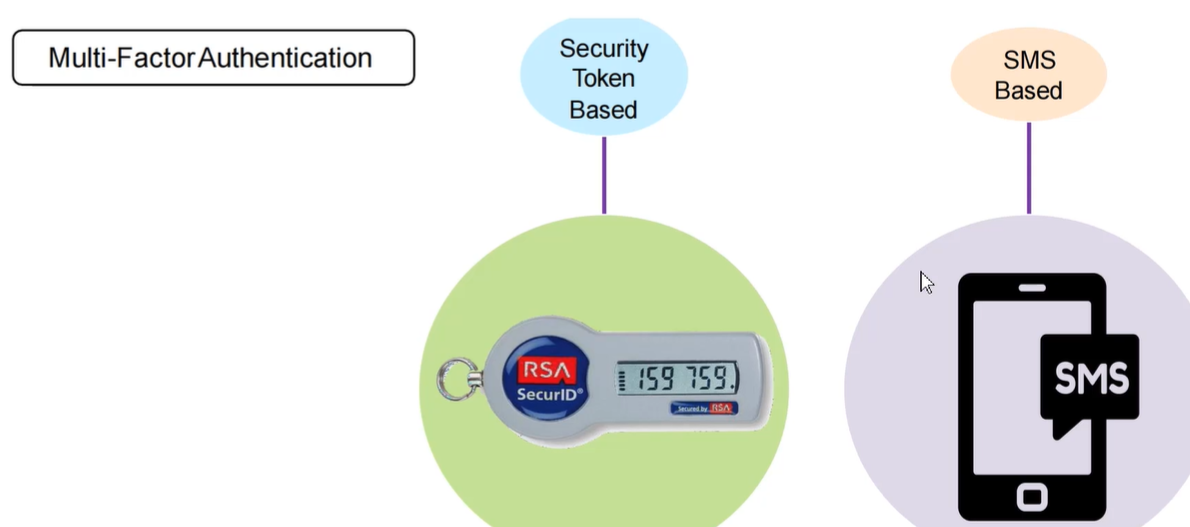
* File system that can establish permissions at the file or folder level and are accessible by multiple users.
* Application workloads that use the SMB protocol and require shared file storage provided by windows-based file systems (NTFS).
* It is compatible with the following compute services;
* Amazon elastic compute cloud, instances of amazon workspaces, instances of amazon app stream &VMWare cloud VMS running on AWS environments.
* Using FSX, we can launch and run high performing file systems with just a few clicks while avoiding tasks such as provisioning hardware, configuring software, or taking backups.
* It provides the option to choose 2 file systems. 1. FSX for windows file server. 2. FSX for luster.
* It is simple and fully managed.
* Highly available and durable.
* Secure and compliant.
* Fast Delivery.
* Pay only for the used resources.
* Easy integration with other AWS services.

1. **Introduction to IAM, Cloud watch, and AMI**

**IAM (Identity and Access Management):** AWS IAM is a web service that helps you securely control access to AWS resources. It is to control who is authenticated and authorized to use resources.



**Features of AMI:**

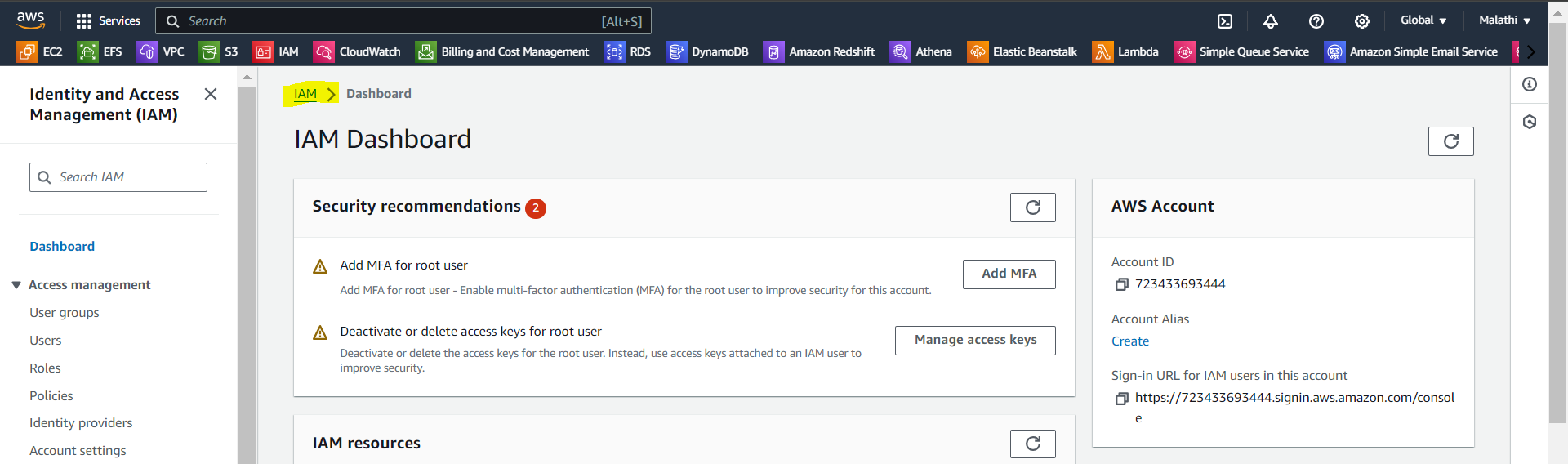


The roles and policies are created and attached to the users and groups based on the permissions requirements.

For more details, please visit the link: <https://aws.amazon.com/about-aws/whats-new/2024/06/aws-iam-access-analyzer-policy-checks/>

**CloudWatch:** Usingcloudwatch, we can create alarms on billing for the data usage in the AWS platform for all the services used.

* It monitors all the AWS resources provisioned and deployed.
* Sends notification if anything goes wrong.
* Events, Alarms, Logs and Metrics are fundamentals to cloudwatch monitoring.
* Individual data points which are monitored, all actions are based on metrics. E.g. CPU utilization %.
* All AWS services send metrics to CloudWatch by default.



**AMI (Amazon Machine Image):** An Amazon Machine Image (AMI) is an image that provides the software that is required to set up and boot an Amazon EC2 instance. Each AMI also contains a block device mapping that specifies the block devices to attach to the instances that you launch. You must specify an AMI when you launch an instance. The AMI must be compatible with the instance type that you chose for your instance. You can use an AMI provided by AWS, a public AMI, an AMI that someone else shared with you, or an AMI that you purchased from the AWS Marketplace.

An AMI is specific to the following:

* Region
* Operating System
* Processor architecture
* Root device type
* Virtualization type

You can launch multiple instances from a single AMI when you require multiple instances with the same configuration.

1. **Elastic Load Balancer and Auto Scaling**

**ELB (Elastic Load Balancer) :** Load balancer is a service that uniformly distributes the traffic and workloads across multiple servers or a cluster of servers. Load balancer increases the availability and fault tolerance of an application.

* ELB scales itself as necessary to handle the load.
* Incoming traffic is distributed across EC2 instances in multiple availability zones.
* Load balancer is the single point of contact for the clients.

**Types of load balancer:** There are 3 types: Classic Load Balancer, Network Load Balancer, and Application Load Balancer.

**Classic Load Balancer:** It resembles the traditional load balancing, but virtual devices replace the physical hardware. It distributes the incoming application traffic across EC2 instances in multiple AZs and functions at Layer 7 of the OSI model. It routes the traffic to the healthy EC2 instances only and it is evenly distributed.

**Network Load Balancer:** This is used to handle the sudden and violent traffic. It functions at the 4th layer of the OSI model. It handles the millions of requests per second and maintains low latency. Ideal for load balancing the TCP traffic and supports elastic or static IP.

**Application Load Balancer:** This identifies the incoming traffic type and directs it to the specific resources. Functions at the 7th Layer of the OSI (Open System Interconnection) model.

**7 Layers of OSI Model:**

* 1. The Physical Layer.
* 2. The Data Link Layer.
* 3. The Network Layer.
* 4. The Transport Layer.
* 5. The Session Layer.
* 6. The Presentation Layer.
* 7. The Application Layer.

**Auto Scaling:** Auto scaling is the process of scaling up and scaling down the instances based on the load. AWS Auto scaling monitors your applications and adjusts the capacity automatically to ensure consistent, predictable performance at the lowest possible cost. It is simple to set up application scaling for multiple resources across multiple services using AWS auto scaling in minutes.

* Scaling is the process of adding/removing capacity/resources as needed.
* Scale out is adding the capacity/resources.
* Scale in is removing the capacity/resources.
* Types: Vertical and Horizontal
* Auto scaling is scaling out/in automatically without any manual intervention.
* It helps ensure that the correct number of EC2 instances are available to handle the load.
* Multi-AZ instances provide high availability solutions.
* Auto scaling can dynamically increase or decrease capacity as needed.
* Load Balancer automatically registers instances in the group.
* Heath Checks: EC2 instances only: EC2 status are considered.
* EC2 and ELB Health Checks: An instance is considered unhealthy if either of the health checks fail.

**Route 53:** Route 53 is the highly available and scalable Domain Name System provided by AWS.

Example: In [www.amazon.com](http://www.amazon.com) 🡪 com: Top level domain name, amazon 🡪 Domain name, Domain name system is an Internet service that translates the domain names into IP addresses. Also, the endings like .com, .in, .gov, .ord etc represent different domains.

Hosting a Website:

* Start up and server/host where the web service will run(IP address of the server is 10.20.30.60)
* Get a domain name from the DNS (Domain Name Server) providers such as Go daddy, Freenom, etc.
* Link the domain name with the IP address from the above using domain service/system.
* Network latency is an amount of time taken to deliver some amount of data over the network.
* If an application is hosted on EC2 instances in multiple regions, user latency can be reduced by serving requests from the region where
* Network latency is the lowest
* We have to create latency resource record set for the amazon EC2 resource in each region that hosts the application latency record sets can be created for both ELB and EC2 instances.
* Latency on the internet can be changed over time due to changes in routing or any other factor.
* Geo location routing can be used to send the traffic to resources based on the geographical location of users. For example, all queries from Europe can be routed to the IP address: 10.30.50.70
* Geo location works by mapping IP addresses, irrespective of regions, to locations.

1. **VPC (Virtual Private Cloud):** A **Virtual Private Cloud (VPC)** is like having your own private network within a cloud provider (like AWS or Google Cloud), where you can safely run your applications and store data.

It’s like setting up a secure office network, but in the cloud. You control how your resources (like virtual servers, databases, etc.) connect to each other and the internet. You can also decide which parts of your network are accessible from the outside world and which parts are kept private.

For example:

* **Public-facing resources** (like a website) can be accessed by users via the internet.
* **Private resources** (like databases) can be hidden and only accessible to your application servers, making everything more secure.

**Components of VPC:**

* Network Interfaces
* Route Tables
* Internet Gateway
* Network Address Translation(NAT) Gateway
* Security Groups and NACL

Network Interface:

* Interface between computer and internet network.
* Network IO (Input/Output) happens via network interface cards.
* Network interfaces contain – Elastic IP, Public IP, Private IP, Security Groups
* It can be created to an instance, attached to an instance, detached from an instance, re-attached to another instance.
* Network interfaces can have additional secondary IP address attached to it.
* IP address can be assigned to n/w interfaces attached to a running to stopped instance.

Route Tables:

* Route Table tells a machine/network where the traffic is directed.
* Directions are defined by “routes” in Route Tables.
* Each subnet must be associated with a route.
* All the VPCs come with an implicit router and a main route table which can be modified.

Internet Gateway and NAT Gateway:

* An internet gateway is a horizontally scaled, redundant and highly available VPC component that allows communication between instances in your VPC and internet.
* Purpose of an Internet gateway:
* It can be created to an instance, attached to an instance, detached from an instance, re-attached to another instance.
* Internet cannot initiate any connection to the instance via NAT.
* NAT devices enable instances in the Private Subnet to connect to the internet and brings responses back to the instances.
* NAT devices are created in public subnet.

Security Groups and NACL(Network Access Control Lists) :

* A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

NACL(Network Access Control Lists):

* This is an optional layer of security for your VPC that acts as a firewall.
* It controls the traffic in and out of one or more subnets.

**VPC Peering:** VPC peering is the network connection between 2 VPC’s which enables traffic flow between them using private IP addresses. Peering connection can be created between VPC’s in the same or different accounts and between VPC’s in the same or different regions.

**VPC Endpoints:** VPC endpoint enables you to privately connect your VPC supported AWS services and VPC endpoint services powered by AWS Private Link without requiring an internet gateway. NAT device, VPC connection, or AWS direct connect connection.

**AWS Transit Gateway:** AWS Transit gateway is a central hub that connects your Amazon virtual Cloud (VPCs) and on-premises networks. This simplifies your network and eliminates complicated peering relationships. It functions as a cloud router, establishing new connections only once.

**Advantages of AWS Transit Gateway:** When compared to VPC peering, there is more visibility (network manager, Cloud watch metrics, and flow logs).

* Fine-grain routing is possible with TGW Route Table per attachment.
* The number of regions determines the complexity.

**Disadvantages of AWS Transit Gateway:**

* Each additional hop will add latency.
* Potential bottlenecks in regions peering connections.
* Pricing is based on hourly rates for attachments, data processing and data transfer.

**Advantages of VPC Peering:**

* Only data transfer is charged for.
* There is no bandwidth restriction.

**Benefits of AWS Transit Gateway:**

* Improved connectivity
* Improved visibility and control
* Enhanced security
* Multicasting that is adaptable

**AWS Direct Connect:** Improved The AWS direct connect cloud service that provides the quickest connection to your AWS resources. Your network traffic remains on the AWS global network and is never exposed to the public internet while in transit. This reduces the possibility of encountering bottlenecks or unexpected increases in latency.

**Benefits of AWS Direct Connect:**

* Create hybrid networks.
* Extend your current network.
* Control large datasets.

**AWS VPN:** AWS virtual private network connect your on-premises networks, remote offices, client devices, and the AWS global network in a secure manner. AWS VPN is made up of two components. AWS site-to site VPN and AWS client VPN. To protect your network traffic, each service offers a highly available, managed, and elastic cloud VPN solution.

1. **Simple Storage Service (S3):** AWS S3 is a storage that can be maintained and accessed over the internet. S3 provides the web service that can be used to store and retrieve unlimited amount of data. Same can be done programmatically using Amazon-provided API’s. We can upload files, folders, images, songs and videos from a machine and access them from anywhere in the world.

* API is a list of specifications that describe how information is exchanged between programs.
* Software that wants to access another will call the API published by the other program.
* S3 provides highly durable and available solutions by replicating all the data in multiple data centers in a region.
* Data uploaded in a particular region never leaves it.
* Read – after – write consistency.
* Eventual consistency.
* S3 follows a storage hierarchy while keeping data (documents, images, videos, files, etc).
* Management console or S3 APIs can be used to manage buckets and projects.
* By default, maximum number of buckets that can be created per account is 100. For additional buckets, one can submit a service limit increase.
* Objects, videos, images, documents, etc which are stored inside the buckets.
* While creating a bucket, a name is given and the name is the object key.
* There cannot be any sub-bucket, or sub-folder inside a bucket (physically, however, the folders can be created on the console, which provides a logical hierarchy only and are used as prefixes in the object key.)

1. **Database Services:** Database is a collection of information that is organized so that the data or the information can be easily accessed, maintained and updated. The data in a database can be added, updated, and deleted. A software system is created to perform all these operations on a database is called a database management system or DBMS.

There are several types of databases that are categorized based on various facts. Following is the list of the most used and popular types of databases that are categorized mainly on the basis of the different ways in which the data is stored in each of them. RDBMS (Relational database management system), No SQL Database, Hierarchical Database, Flat file Database.

* In RDBMS, the data stored in a tabular form, and SQL (Structured Query Language) is used to run queries for inserting, updating, deleting and searching for the data or records.
* No SQL Database doesn’t follow the normal row/column or table approach of storing data like RDBMS. Hence, it is a non-relational database system. No SQL Databases store the data in the JSON format. Data structure included in No SQL are key/value, wide column, graph, or documentations.
* In Hierarchical Database, data is stored in parent-child relationship nodes. These databases also contain the information of the respective groups of the data according to the parent-child relationships along with the data.
* There are no particular structures for indexing and recognizing relationships among the data in flat file databases. All the records follow a uniform format and are saved in a file called flat file, which can be a plain text file or even a binary file.
* AWS provided a fully managed and purpose built database services to support relational as well as non-relational database requirements of their clients.
* Relational Database: Amazon RDS, Amazon Aurora, Amazon Redshift
* Key-Value Database: Amazon Dynamo DB
* In memory: Amazon ElastiCashe.

**AWS RDS (Relational Database Service):** RDS is used to set up, manage and scale a relational database instance in the cloud. RDS is a fully managed RDBMS service. It manages backups, software patching, failure detection, and many more tasks. With RDS, CPU, memory, storage and IOPS are all independent and hence, can be scaled independently. RDS offers mainly six database engines: Amazon aurora, PostgreSQL, MySQL, MariaDB, Oracle DB and SQLserver. Performance and Scalability, High availability, Security, Backup and Restore, Maintenance and Upgrades are the features of RDS.

**Amazon Aurora:** It is a relational database engine built for the cloud, and it is completely compatible with MySQL and PostgreSQL. It is fully managed by Amazon RDS, which in turn automates administrative tasks such as database set up, patching, backups and more.

**Dynamo DB:** It is a noSQL database, which is fully managed by AWS to provide fast and predictable performance. It creates database tables that can store and retrieve any amount of data. The features are:

* Key/Value and document data models.
* Automated global replication.
* Read/Write capacity modes.
* Point-in time recovery.

In DynamoDB, the collection of items is known as table. A table in DynamoDB is not a structured table with a fixed number of cells or columns. Each table contains 0 or more items. An item is a group of attributes uniquely identifiable among all the other items.

**AWS Redshift:** It is a fully managed data warehouse in the cloud. The data in the amazon redshift data warehouse can be scaled up to petabytes or more. To create a database, a set of nodes called Amazon Redshift cluster has to be launched. It has the advantages: Faster performance, Easy to set up, Deploy and manage, Cost effective, Secure.

**AWS Elasticache:** It is a web service that provides high-performance, cost effective and scalable caching solutions. With ElastCache, it is easy to manage and scale a distributed in-memory data store or cache environment in the cloud itself.

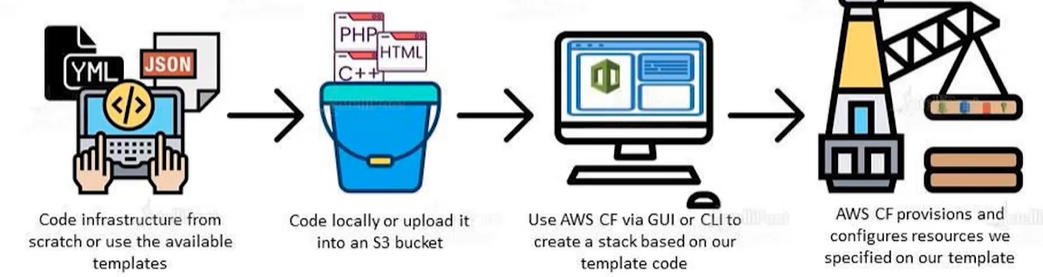
**AWS Athena:** It is a query service that allows you to easily analyze data in Amazon S3 using standard SQL. Because Athena is server less, there is no infrastructure to manage, and you only pay for the queries you run.

**Athena Benefits:**

* Begins querying immediately. It is server less, no ETL.
* It is standard, open and powerful.
* It is Presto-based application that runs standard SQL.
* It is really fast.
* Even for large databases, interactive performance is excellent.
* Pay per search.
* Only pay for data that has been scanned.

1. **Cloud Formation and App Services:** Managing all the resources of complex application on AWS becomes a problem when there are a lot of resources to be maintained.

* We can create and modify an existing AWS Cloud formation template. A template describes all of our resources and their applications.
* To make changes or modify, we can simply track the differences in our templates to check changes in our infrastructure, similar to the way developers control revision to source code.
* We can also reuse our template to set up our resources consistently and repeatedly. We can just describe our resources once and then provision the same resources over and over to the multiple regions.
* A cloud formation template is JSON or YAML formatted file. We can save these files using the extensions .json, .yaml, .txt and .template.
* We can also specify multiple resources in a single template and configure these resources to work together.
* We can manage related resources as a single unit called a stack. We can also delete the entire architecture by deleting the task. We can work with stacks by using AWS cloud formation console, API, or AWS CLI.
* To change the running resources, we need to update the stack. We can generate a change set, which is a summary of our proposed changes. If we generate a change set, we can see how changes cause our database to be replaced, and we will be able to plan accordingly.
* This is how the Cloud formation works:



* Instead of creating multiple instances, databases and other resources manually, we can create a single piece of code, which we can use to create multiple infrastructure.
* Cloud formation not only creates resources on our AWS account but also waits for them to stabilize while they start. It verifies whether provisioning was successful, and if there is a failure then it can gracefully role the infrastructure back to a past known good state.
* Templates describe the resources that we want to provision to our AWS cloud formation stacks. We can use AWS Cloud formation designer or any text editor to create and save template.
* We can author AWS cloud formation templates in JSON or YAML formats. All cloud formation features and functions, including cloud formation Designer, support both formats. If we add comments to the YAML template created in a designer, they will not be preserved while converting the template into JASON. However, we can add comments to the template we created locally.
* A stack for instance, can include all the resources required to run a web application, such as a web server, a database and a networking rules. If we no longer require a web application, we can simply delete the stack, and all of its related resources will be deleted.
* If a resource cannot be created, AWS cloud formation rolls the stack back and automatically deletes any resources that were created.

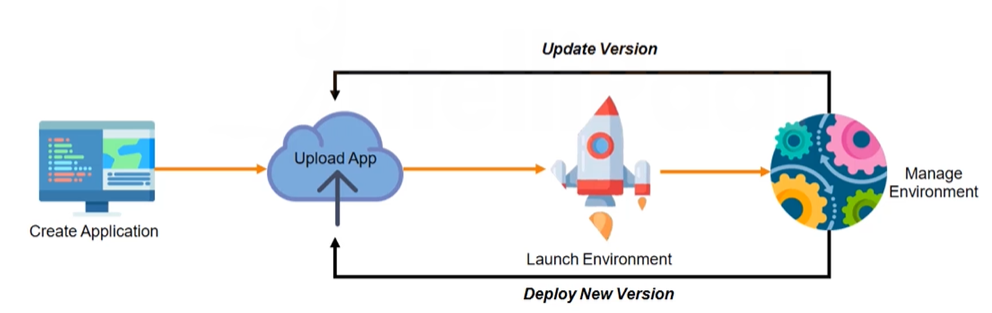
1. **AWS Lambda, Elastic Beanstalk, AWS Opsworks and API Gateway:**

**AWS Lambda:** Managing AWS Lambda is a server less compute service where your code is triggered in response to an event occurrence and it also manages the underlying compute resources.

* AWS Lambda is a Platform as a service (PaaS) with a remote platform to run and execute your backend code.
* Environment restriction, as it is restricted to few languages only. So, just choose your environment where you want to run your code and push the code into AWS Lambda. No flexibility to log into compute instances, choose customized operating system or language runtime.
* AWS Lambda is used only for running and executing the back-end code.
* We cannot select the AWS resources, like a type of EC2 instance, lambda provides an EC2 resources based on your workload. It is a stateless system.
* First, upload the code to the lambda in one or more lambda functions and it will then execute the code on your behalf.
* After the code is invoked, Lambda automatically takes care of provisioning and managing the required servers.
* Building a server less we website allows you to focus on your website code.
* You don’t have to manage and operate its infrastructure.
* Hosting your static website over S3 and then making it available using AWS lambda makes it easier for the users to keep track of their resources being used.
* One can easily schedule the lambda events and create back-ups in their AWS accounts.
* Create the back-ups, check if there are any idle resources or not, generate reports using Lambda in no time.
* One can easily use it for transforming the data between Lambda and other Amazon servers like S3, Kinesis, and Redshift and database services.
* Also, you can filter the data before sending.
* One can easily transform and load data between Lambda and these services. Next lambda concepts.

**Elastic Beanstalk:** Beanstalk is a service where you can create, run and manage application without worrying about the underlying infrastructure. It is a PaaS. You can directly upload your upload your website code to Beanstalk, and it will automatically host the application for you with a URL. You can concentrate on the code of your application rather the architecture which it is hosted on.

You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring.



* In beanstalk, an application is the folder you upload which is a logical collection of the app version, environment and configuration.
* Application version is a specific iteration of the code which is uploaded to Beanstalk. For example, if the first version is 1, then the next would be 1.1 and so on.
* An environment is a collection of AWS resources on which the application runs on. In an environment, at a time period, only one version of an application can be running. If you want to run multiple application versions at the same time, then you will have to create more environments and run another application version on it.
* When the environment is launched in the beanstalk, one of these tiers must be chosen. The tier decides the AWS resources which should be assigned for that and the type of application which it can run.
* Webserver environment: Using this environment, you can directly upload your application’s code and get it hosted. The URL will be provided as myapp.uss-west-2.elasticbeanstalk.com.
* If you have operations in your web server which might take a long time to complete, you can offload them to the worker. For ex; a task which processes videos or one which generates a ZIP archive.
* Environment configuration: These are the parameters which you enter while creating an environment. If you have selected Node.js as your runtime, then it will be completely installed on the servers which it spins up.
* You can also update your configuration whenever you need.
* Beanstalk will either apply the changes to the existing resources or deletes and deploys new resources.
* Saved configuration is a template which can be created. These templates contain the configuration details and you can directly use them to create your own unique environment. Even if you delete the environment which is running, you can create the exact same environment with this template. Platform is a combination of an operating system, programming language run time, web server, application server, and elastic beanstalk components.

**Configuration Management:** The main purpose of configuration management is to automate server management. If there are 10 similar webservers and you want to run a configuration update on all of them. Top open each server and installing a software is a time taking process. So, instead, we can create a script and run it on each server automatically to roll updates on all the servers at a time.

**AWS OPsworks:** Opsworks is a configuration management service which lets you configure and operate applications on the cloud. Opsworks gives you the benefit of choosing between Chef and puppet. Benefits are:

* It supports any application.
* Automation to sun at scale.
* Configuration as code.
* Resource organization.
* It provides businesses managed instances of chef & puppet to automate server management on your EC2 instances or non-premises instances.
* Opsworks mainly works mainly on the orchestration of servers and managing the software configuration rather than looking into the architecture.
* Use opsworks when you must manage the configuration of multiple servers which serve a similar purpose.

**API Gateway:** Amazon API Gateway is a fully managed service that enables developers to easily create, publish, maintain, monitor and secure API’s of any size. API’s serve as the “front door” through which applications access data, business logic, functionality from your backend services.

**Advantages of API Gateway:**

* Effective API development performance at any scale.
* Cost savings on a large scale.
* Monitoring is simple.
* Security controls that are adaptable.
* Options for restful API’s.

1. **SNS (Simple Notification Services):** SNS is a cloud-based notification service. It provides push-based and many –to-many messaging. It is easy to set up, operate and send notifications from the cloud.

AWS Simple Notification Service (SNS) is a fully managed messaging service that allows you to send notifications or messages between systems or directly to end-users. It is often used in distributed systems to communicate across micro services, trigger workflows, or send alerts.

### Key Concepts of AWS SNS:

1. **Topics**:
   * A "topic" is a communication channel to which subscribers can subscribe. When a message is published to the topic, it is delivered to all subscribers.
   * Topics allow you to group multiple endpoints into a single communication target.
2. **Publishers**:
   * These are the entities that send messages to a topic. Publishers can be any AWS service, like Lambda, EC2, or external systems.
3. **Subscribers**:
   * These are the endpoints that receive messages from a topic. They can be various endpoints, such as:
     + HTTP/HTTPS endpoints (e.g., web services)
     + AWS Lambda functions
     + Email addresses
     + SMS (mobile numbers)
     + SQS (Simple Queue Service)
4. **Message Format**:
   * SNS supports sending plain text messages or structured JSON, allowing you to target different protocols with different message formats in one go.
5. **Types of Notifications**:
   * **Push Notifications**: Send notifications to mobile devices (iOS, Android).
   * **Email/SMS**: Send notifications directly to users.
   * **SQS**: Deliver messages to SQS queues for processing.
6. **Message Filtering**:
   * SNS supports message filtering, which allows subscribers to receive only certain types of messages based on message attributes.
7. **Security**:
   * SNS uses AWS IAM roles and policies to control who can publish or subscribe to topics.

### Example Use Cases:

* **System Alerts**: Send an alert to administrators via email, SMS, or Lambda when something goes wrong (e.g., a server goes down).
* **Event-Driven Architecture**: Microservices can communicate by sending messages to an SNS topic, which triggers downstream services.
* **Mobile Push Notifications**: Notify users of an update or event on their mobile app.

### Workflow of SNS:

1. A publisher sends a message to a topic.
2. SNS distributes the message to all subscribers of that topic.
3. Subscribers (like email, Lambda, or HTTP endpoints) receive the message and act upon it.

### 1. ****Creating a Topic****:

A topic is the channel through which messages are published. Here's how to create one:

**Steps:**

1. Go to the [AWS SNS Console](https://console.aws.amazon.com/sns/).
2. Click **Create topic**.
3. Choose **Standard** for most use cases (FIFO topics are available if you need message ordering).
4. Enter a **name** for your topic (e.g., UserAlerts).
5. Optionally, configure permissions if you want to control who can publish/subscribe.
6. Click **Create topic**.

### 2. ****Subscribing to the Topic****:

Now that the topic is created, you need to add subscribers (e.g., email, SMS, Lambda).

**Steps**:

1. In the SNS Console, click on newly created topic.
2. Scroll down to the **Subscriptions** section.
3. Click **Create subscription**.
4. Choose the **protocol** (e.g., Email, SMS, or Lambda).
5. Enter the **endpoint**:
   * For Email: Enter the recipient's email address.
   * For SMS: Enter the phone number in international format (+1 for US, etc.).
   * For Lambda: Choose the Lambda function to trigger.
6. Click **Create subscription**.

**Verification (for Email/SMS)**:

* If you selected **Email** or **SMS**, the recipient will need to confirm the subscription by clicking on a link in the email or replying to the SMS.

### 3. ****Publishing a Message****:

Once subscribers are in place, you can send messages to the topic.

**Steps**:

1. In the SNS Console, go to the topic you created.
2. Click **Publish message**.
3. Enter a **Subject** (for email protocols) and the **Message body**.
4. You can optionally customize the message for each protocol (e.g., different messages for SMS and email).
5. Click **Publish message**.

The message will be sent to all subscribers, and if everything is set up correctly, they should receive it immediately.

### 4. ****Example Use Case: Alert System with Lambda Trigger****

Imagine you have a system that monitors the health of your web servers. When a server goes down, you want to send an alert to administrators via SNS and trigger an AWS Lambda function to automatically attempt a restart.

**Steps**:

1. **Create an SNS Topic** for "ServerAlerts".
2. **Subscribe** to the topic:
   * Add an email for administrators to get a human-readable alert.
   * Add a Lambda function to automatically restart the server.
3. **Write a Lambda function** that handles the server restart process.
   * The Lambda function can be triggered by SNS whenever an alert is published to the topic.
   * Lambda can access AWS EC2 API to restart the server.
4. **Publish an alert** (manually or through a monitoring tool) when a server goes down. Security:

* Use **AWS IAM policies** to control who can publish to or subscribe to topics.
* You can allow specific AWS services (like EC2, CloudWatch, or Lambda) or users to publish messages.
* To control subscription access, you can restrict who can subscribe to topics, ensuring only authorized endpoints can receive messages.

1. **SQS (Simple Queue Services):** SQS is a cloud-based message queuing service. It enables us to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS provides a hosted queue that is secure, durable and available, allowing you to integrate and decouple distributed software systems and components.

* Amazon SQS includes standard constructs like dead-letter queues and cost allocation tags. It provides a generic web services API that can be accessed using any programming language supported by the AWS SDK(Software development kit.

**AWS Configuration parameters:**

* Visibility Timeout
* Message Retention period
* Delivery delay
* Maximum Message Size
* Enable content based deduplication
* Receive message time
* Enable high throughput FIFO
* Redrive allow policy.

The amount of time that a message received from a queue (by one consumer) is not visible to other message consumers is called visibility timeout.

The amount of time Amazon SQS keeps messages in the queue. Messages are kept in the queue for four days by default. You can set a queue to keep messages for up to 14 days. This is called Message Retention Period. The amount of time that Amazon SQS will wait before delivering a message that has been added to the queue. Amazon SQS can automatically create duplication IDs based on the body of the message.

It is used to enable high throughput for queued messages. By selecting this option, the related options(Deduplication scope and FIFO throughput limit) are changed to the necessary settings for enabling high throughput for FIFO(First In First Out) queues.

**Key Features of AWS SQS:**

1. **Message Queuing**:
   * SQS enables the decoupling of different components or services, allowing them to interact asynchronously.
   * One service can send a message to the queue, and another can process it later, ensuring that the sender and receiver don’t need to operate at the same speed.
2. **Two Types of Queues**:
   * **Standard Queue**:
     + Provides at-least-once message delivery.
     + Messages can be delivered more than once, but message ordering is not guaranteed.
   * **FIFO (First In, First Out) Queue**:
     + Guarantees exactly-once message delivery.
     + Preserves the order in which messages are sent and received.
3. **Scalability**:
   * SQS automatically scales to handle large volumes of messages without any need for manual scaling or intervention.
4. **Fault Tolerance**:
   * SQS ensures messages are reliably stored in the queue until they are processed.
   * If a message is not processed successfully, it can remain in the queue for another attempt (configurable visibility timeout).
5. **Message Retention**:
   * Messages can be retained in the queue for up to 14 days. The retention period can be customized depending on the use case.
6. **Security**:
   * SQS integrates with AWS Identity and Access Management (IAM) to control access and permissions for different users and applications.
   * Messages can also be encrypted using AWS Key Management Service (KMS).
7. **Cost-Effective**:
   * You only pay for what you use, based on the number of API requests and the data transferred. There's no upfront cost or server management.

**Common Use Cases:**

1. **Decoupling Microservices**: SQS is often used to decouple Microservices, where one service sends a message (e.g., order placed) and another service processes it (e.g., process payment).
2. **Asynchronous Workflows**: It helps manage asynchronous workflows, such as video processing, where a video is uploaded, and then multiple steps (like transcoding) happen in sequence without immediate execution.
3. **Batch Processing**: SQS can be used to queue up a large number of tasks for batch processing, ensuring they are processed one by one or in parallel by multiple consumers.
4. **Reliable Message Delivery**: For applications that require reliable message handling, SQS ensures no message is lost or overlooked.

**Example of SQS Flow:**

1. A user uploads a photo to an S3 bucket.
2. An SQS message is sent to a queue notifying the photo processing service.
3. The photo processing service pulls the message from the queue, processes the photo, and sends back the processed data to another service (e.g., storage or further actions).

Overall, AWS SQS is an essential tool for building decoupled, scalable, and reliable distributed systems.

1. **SES (Simple Email Services):** SES is a cloud-based email sending service which is designed to help digital marketers and developers send appropriate emails. It is reliable and cost effective. A sender sends a mail, SES collects it and pushes to the receiver and the receiver finally sends it to the recipient.

**Key Features of Amazon SES:**

1. **Email Sending**:
   * SES allows sending emails to individual users or large groups, whether for transactional purposes (e.g., order confirmations, password resets) or marketing campaigns (e.g., newsletters, promotions).
2. **Scalability**:
   * SES is highly scalable, allowing you to send millions of emails per day. It can handle both low-volume and high-volume email sending needs.
3. **Deliverability**:
   * Amazon SES has high deliverability rates due to its strong relationship with internet service providers (ISPs) and its adherence to best practices (like using dedicated IP addresses and providing features like DKIM and SPF for email authentication).
   * SES allows users to monitor their sending activity and helps improve email delivery with detailed bounce, complaint, and delivery reports.
4. **Email Receiving**:
   * SES also supports email receiving, allowing you to receive emails, filter them, and store them in an S3 bucket, or trigger AWS Lambda functions or SNS notifications.
5. **Cost-Effective**:
   * SES operates on a pay-as-you-go pricing model, with no upfront costs. You pay only for the number of emails sent, received, or the amount of data processed. It’s very economical compared to traditional email delivery platforms.
6. **Flexible Integration Options**:
   * You can integrate SES with existing applications using the **SMTP interface** or through the **AWS SDKs** (for programming languages like JavaScript, Python, Java, etc.).
   * You can also trigger other AWS services (like Lambda or SNS) based on incoming emails.
7. **Security Features**:
   * **DKIM (Domain Keys Identified Mail)** and **SPF (Sender Policy Framework)**: SES supports DKIM and SPF, ensuring that your emails aren’t marked as spam or phishing by recipient email providers.
   * **IAM (Identity and Access Management)**: You can configure user permissions to control who can send emails from your SES account.
   * **TLS (Transport Layer Security)**: Ensures that the email content is encrypted when sending emails via the SES service.
8. **Email Analytics**:
   * Amazon SES provides insights into how your emails are performing. You can track bounce rates, delivery rates, complaint rates, and engagement metrics like opens and clicks by integrating with AWS CloudWatch or third-party analytics tools.
9. **Reputation Dashboard**:
   * SES offers a built-in reputation dashboard that helps monitor sender reputation. It provides insights into bounce and complaint metrics, which affect email deliverability.

**Common Use Cases:**

1. **Transactional Emails**:
   * Sending confirmation emails for orders, account registration, password resets, receipts, etc. These emails are triggered by specific user actions.
2. **Marketing and Bulk Emails**:
   * Sending newsletters, promotional offers, and product updates to large groups of recipients.
3. **Email-Based Notifications**:
   * Sending automated system alerts, notifications, and other critical messages.
4. **Inbound Email Processing**:
   * Processing and categorizing incoming emails, such as customer support inquiries or feedback, by integrating with other AWS services like S3, Lambda, or SNS for further processing.

**Steps for Using Amazon SES:**

1. **Verify Domains and Email Addresses**:
   * To send emails, you must first verify the domain (e.g., example.com) or individual email addresses that will be used as the sender. This ensures that you own the domain or have permission to send from the email addresses.
2. **Choose Sending Method**:
   * You can send emails using the **SMTP** interface or **AWS SDKs**, based on your preferred programming language or integration.
3. **Authenticate and Improve Deliverability**:
   * Configure DKIM, SPF, and DMARC records to improve the chances of your emails reaching your recipients' inboxes.
4. **Monitor and Track**:
   * Use the SES dashboard, CloudWatch metrics, or the reputation dashboard to monitor the performance of your emails, including delivery, bounce, and complaint rates.
5. **Request Production Access**:
   * When starting with SES, you are in the SES sandbox, where limits are imposed on sending emails. You’ll need to request production access to increase your sending limits and use the service for actual users.

**Example Use Case: Sending a Transactional Email**

1. A user registers on your e-commerce website.
2. Your application triggers an email confirmation using SES, where the message is constructed and sent to the user.
3. SES handles the email sending, ensuring it reaches the user's inbox and providing feedback on whether the email was successfully delivered or bounced.
4. You can track the status of the email (opened, clicked, bounced, etc.) for insights into user engagement.

**Advantages of Amazon SES:**

* **Low-Cost**: No upfront commitments, and you only pay for what you send or receive.
* **Scalability**: SES can handle large email volumes, making it ideal for growing businesses or organizations.
* **Easy Integration**: With built-in support for SMTP and AWS SDKs, it’s simple to integrate SES into most existing applications.
* **Reliable Delivery**: Amazon SES focuses on high deliverability rates, reducing the chances of your emails being marked as spam.

Overall, Amazon SES is a powerful tool for businesses that need a flexible, scalable, and reliable email sending service, particularly for high-volume use cases like marketing and transactional emails.