**AWS Services**

**What is AWS?**

AWS stands for Amazon Web Services.

The AWS service is provided by the Amazon that uses distributed IT infrastructure to provide different IT resources available on demand. It provides different services such as infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS).

AWS started contributing to their highly available infrastructure platform in 2006, based on the pay-as-you-go model. After that, whatever they have garnered as Services and customers till now is remarkable because they have thousands and Thousands of customers across 191 countries who use AWS platform services for their initiatives, and the number is increasing in the AWS customer bucket.

**Uses of AWS**

A small manufacturing organization uses their expertise to expand their business by leaving their IT management to the AWS.

A large enterprise spread across the globe can utilize the AWS to deliver the training to the distributed workforce.

An architecture consulting company can use AWS to get the high-compute rendering of construction prototype.

A media company can use the AWS to provide different types of content such as ebox or audio files to the worldwide files.

**Pay-As-You-Go**

Based on the concept of Pay-As-You-Go, AWS provides the services to the customers.

AWS provides services to customers when required without any prior commitment or upfront investment. Pay-As-You-Go enables the customers to procure services from AWS.

**Security:** For every organization, security is a very vital element.

The security features of AWS are as follows:

° Certifications and accreditations

° Identity and Access Management

**Global infrastructure:** AWS provides a fully-functional, flexible technology infrastructure platform worldwide with managed services with certain characteristics, for example:

° Multiple global locations for deployment

° Low-latency CDN service

° Reliable, low-latency DNS service

**Compute:** AWS offers a huge range of various Cloud-based core computing services, including a variety of compute instances, which can be automatically scaled to justify the needs of your users and application; a fully managed elastic load balancing service; and more fully managed desktop resources on the pathway of AWS Cloud. Some of the common characteristics of computer services include the following:

° Broad choice of resizable compute instances

° Flexible pricing opportunities

° Great discounts for compute resources are always on

° Lower hourly rates for elastic workloads

° Wide-range of networking configuration selections

° A widespread choice of operating systems

° Virtual desktops

° One can save as one grows, with the tiered pricing model

**Storage:** AWS offers low cost with high durability and availability with their storage services. The pay-as-you-go pricing model with no commitment provides more flexibility and agility in services and processes for storage with a highly secure environment. AWS provides storage solutions and services for backup, archive, disaster recovery, and many more. They also support block, file, and object kind of storages with highly available and flexible infrastructures. A few major characteristics of storage are the following:

° Cost-effective, high-scale storage varieties

° Data protection and data management

° Storage gateway

° Choice of instance storage options

**Content delivery and networking Application services**: AWS offers a wide set of networking services, which enable us to create a logical isolated network that network architects define and, creates a private network connection to the AWS infrastructure, with fault-tolerant, scalable and highly available DNS services. It also provides delivery services to your end users for content by very low latency and with high data transfer speed with the AWS CDN service. A few major characteristics of content delivery and networking include the following:

° Application and media files delivery

° Software and large file distribution

° Private content

° Device detection

**Deployment & management:** AWS offers the management of credentials to explore AWS services such as monitor services, application services, and updating stacks of AWS resources. They also have deployment and security services alongside the AWS API activity. A few major characteristics of deployment and management services include the following:

° IAM

° CloudWatch

° Elastic Beanstalk

° CloudFormation

° Data pipeline

° OpsWorks

° CloudHSM

° Cloud Trail

**Databases**: AWS offers fully managed, distributed relational, and NoSQL types of database services. Moreover, database services are capable of in-memory caching, sharing, and scaling with/without data warehouse solutions. A few major characteristics for databases include the following:

° RDS

° SimpleDB and DynamoDB

° Redshift

° ElastiCache

**Advantages of AWS**

**1) Flexibility**

We can get more time for core business tasks due to the instant availability of new features and services in AWS.

It provides effortless hosting of legacy applications. AWS does not require learning new technologies and migration of applications to the AWS provides the advanced computing and efficient storage.

AWS also offers a choice that whether we want to run the applications and services together or not. We can also choose to run a part of the IT infrastructure in AWS and the remaining part in data centres.

**2) Cost-effectiveness**

AWS requires no upfront investment, long-term commitment, and minimum expense when compared to traditional IT infrastructure that requires a huge investment.

**3) Scalability/Elasticity**

Through AWS, autoscaling and elastic load balancing techniques are automatically scaled up or down, when demand increases or decreases respectively. AWS techniques are ideal for handling unpredictable or very high loads. Due to this reason, organizations enjoy the benefits of reduced cost and increased user satisfaction.

**4) Security**

AWS provides end-to-end security and privacy to customers.

AWS has a virtual infrastructure that offers optimum availability while managing full privacy and isolation of their operations.

Customers can expect high-level of physical security because of Amazon's several years of experience in designing, developing and maintaining large-scale IT operation centers.

AWS ensures the three aspects of security, i.e., Confidentiality, integrity, and availability of user's data.

**Regions & Availability Zones**

**What are AWS Regions?**

AWS Regions are separate geographic areas that AWS uses to house its infrastructure. These are distributed around the world so that customers can choose a region closest to them in order to host their cloud infrastructure there. The closer your region is to you, the better, so that you can reduce network latency as much as possible for your end-users. You want to be near the data centers for fast service.

**Best practices for choosing AWS Regions**

Follow these best practices when you choose a region, to help ensure top performance and resilience:

* **Proximity**: Choose a region closest to your location and your customers’ location to optimize network latency.
* **Services:** Try and think about what are your most needed services. Usually, the newest services start on a few main regions then pop up in other regions later.
* **Cost:** Certain regions will cost more than others, so use built-in AWS calculators to do rough cost estimates to inform your choices.
* **Service Level Agreement (SLA):** Just as with cost, your SLA details will vary by region, so be sure to be aware of what your needs are and if they’re being met.
* **Compliance:** You may need to meet regulatory compliance needs such as GDPR by hosting your deployment in a specific — or multiple regions.

**What are AWS Availability Zones?**

An AWS Availability Zone (AZ) is the logical building block that makes up an AWS Region. There are currently 84 AZs, which are isolated locations— data centers — within a region. Each region has multiple AZs and when you design your infrastructure to have backups of data in other AZs you are building a very efficient model of resiliency, i.e. a core concept of cloud computing.

<https://aws.amazon.com/about-aws/global-infrastructure/regions_az>

**AWS EC2**

* EC2 stands for Amazon Elastic Compute Cloud.
* Amazon EC2 is a web service that provides resizable compute capacity in the cloud.
* Amazon EC2 reduces the time required to obtain and boot new user instances to minutes rather than in older days, if you need a server then you had to put a purchase order, and cabling is done to get a new server which is a very time-consuming process. Now, Amazon has provided an EC2 which is a virtual machine in the cloud that completely changes the industry.
* You can scale the compute capacity up and down as per the computing requirement changes.
* Amazon EC2 changes the economics of computing by allowing you to pay only for the resources that you actually use. Rather than you previously buy physical servers, you would look for a server that has more CPU capacity, RAM capacity and you buy a server over 5 year term, so you have to plan for 5 years in advance. People spend a lot of capital in such investments. EC2 allows you to pay for the capacity that you actually use.
* Amazon EC2 provides the developers with the tools to build resilient applications that isolate themselves from some common scenarios.

**What Are the Different Types Of AWS Instances?**

We mentioned instances are basically virtual servers. As with any virtual server, AWS instances run on a host machine. AWS offers nearly limitless virtual host machines with various computing resources (CPU, RAM, storage, and bandwidth).

Different instances have different levels of computing resources, so some are better suited to specific workloads than others. AWS instances are grouped into different families. You can usually recognize instance types by observing their names, which comprise a letter, number, and sometimes an extra letter.

There are five different types of instance types available in AWS. The name of each instance family describes its recommended use case, as shown below:

* General Purpose
* Compute Optimized
* Memory Optimized
* Accelerated Computing
* Storage Optimized

**Use cases for instance categories**

The various instance categories are as follows:

• **General purpose instance**: Data processing, small size databases, enterprise applications/portals like SAP, SharePoint, and so on.

• **Compute-optimized instances**: Batch processing, websites that have very high traffic, GNOME analysis, ads and media serving, computational fluid serving, and video encoding.

• **GPU instances:** Application/software and 2D/3D application streaming, rendering and engineering design, and so on.

• **Memory-optimized:** Applications with larger deployment and analysis suchas SAP, GNOME assemble analysis, distributed memory caches, and so on.

• **Storage-optimized:** Scale out transactional databases, data warehousing, and Hadoop.

• **t1.micro instances:** Low traffic sites, getting hands on with AWS EC2 or for some free tier stuff.

**EC2 Pricing Option**

On demand, Reserved Instances, Spot Instances, Dedicated Hosts.

* **On Demand**

It allows you to pay a fixed rate by the hour or even by the second with no commitment.

Linux instance is by the second and windows instance is by the hour.

On Demand is perfect for the users who want low cost and flexibility of Amazon EC2 without any up-front investment or long-term commitment.

It is suitable for the applications with short term, spiky or unpredictable workloads that cannot be interrupted.

It is useful for the applications that have been developed or tested on Amazon EC2 for the first time.

On Demand instance is recommended when you are not sure which instance type is required for your performance needs.

* **Reserved**

It is a way of making a reservation with Amazon or we can say that we make a contract with Amazon. The contract can be for 1 or 3 years in length.

In a Reserved instance, you are making a contract means you are paying some upfront, so it gives you a significant discount on the hourly charge for an instance.

It is useful for applications with steady state or predictable usage.

It is used for those applications that require reserved capacity.

Users can make up-front payments to reduce their total computing costs. For example, if you pay all your up fronts and you do 3 years contract, then only you can get a maximum discount, and if you do not pay all up fronts and do one year contract then you will not be able to get as much discount as you can get If you do 3 year contract and pay all the up fronts.

**Types of Reserved Instances:**

* Standard Reserved Instances
* Convertible Reserved Instances
* Scheduled Reserved Instances

**Standard Reserved Instances**

It provides a discount of up to 75% off on demand. For example, you are paying all up-fronts for 3 year contract.

It is useful when your Application is at the steady-state.

**Convertible Reserved Instances**

It provides a discount of up to 54% off on demand.

It provides the feature that has the capability to change the attributes of RI as long as the exchange results in the creation of Reserved Instances of equal or greater value.

Like Standard Reserved Instances, it is also useful for the steady state applications.

**Scheduled Reserved Instances**

Scheduled Reserved Instances are available to launch within the specified time window you reserve.

It allows you to match your capacity reservation to a predictable recurring schedule that only requires a fraction of a day, a week, or a month.

* **Spot Instances**

It allows you to bid for a price whatever price that you want for instance capacity, and providing better savings if your applications have flexible start and end times.

Spot Instances are useful for those applications that have flexible start and end times.

It is useful for those applications that are feasible at very low compute prices.

It is useful for those users who have an urgent need for large amounts of additional computing capacity.

EC2 Spot Instances provide less discounts as compared to On Demand prices.

Spot Instances are used to optimize your costs on the AWS cloud and scale your application's throughput up to 10X.

EC2 Spot Instances will continue to exist until you terminate these instances.

* **Dedicated Hosts**

A dedicated host is a physical server with EC2 instance capacity which is fully dedicated to your use.

The physical EC2 server is the dedicated host that can help you to reduce costs by allowing you to use your existing server-bound software licenses. For example, Vmware, Oracle, SQL Server depending on the licenses that you can bring over to AWS and then they can use the Dedicated host.

Dedicated hosts are used to address compliance requirements and reduces host by allowing to use your existing server-bound server licenses.

It can be purchased as a Reservation for up to 70% off On-Demand price.

**Limitations**

In general, you are limited to run a total of 20 on-demand or reserved instances and can request 100 Spot instances per region. If you are a new customer, the instance limit can be lower than the limit described.

**Billing and pricing**

On AWS, you will be charged for what you use and there are no minimum charges; in other words, they follow the pay-as-you-go model. Charges will be based on per hour usage of an instance type. Partial or semi partial instance hours consumed will be counted as full hours. For example, if you are using an instance for 55 minutes, you will be charged for 60 minutes

**Ephemeral versus persistent storage**

(**Temporary vs Permanent)**

**EBS volume** is network attached drive which results in slow performance but data is persistent meaning even if you reboot the instance data will be there.

**Instance store** provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer.

**EC2 Storage Overview**

EC2 instances support two types for block level storage: EC2 Instances can be launched using either Elastic Block Store (EBS) or Instance Store volume as root volumes and additional volumes.

EC2 instances can be launched by choosing between AMIs backed by EC2 instance store and AMIs backed by EBS. However, AWS recommends use of EBS backed AMIs, because they launch faster and use persistent storage

**Instance Store**

* Instance store backed instance is an EC2 instance using an Instance store as root device volume created from a template stored in S3.
* An instance store is ephemeral storage that provides temporary block level storage for your instance. Instance store is ideal for temporary storage like buffers, caches, and other temporary content.
* Instance store volumes accesses storage from disks that are physically attached to the host computer.
* When an Instance stored instance is launched, the image that is used to boot the instance is copied to the root volume (typically sda1).
* Instance store provides temporary block-level storage for instances.
* Data on an instance store volume persists only during the life of the associated instance; if an instance is stopped or terminated, any data on instance store volumes is lost.

**Key points for Instance store backed Instance**

* Boot time is slower than EBS backed volumes and usually less than 5 min
* Can be selected as Root Volume and attached as additional volumes
* Instance store backed Instances can be of maximum 10GiB volume size
* Instance store volume can be attached as additional volumes only when the instance is being launched and cannot be attached once the Instance is up and running
* The data in an instance store persists only during the lifetime of its associated instance. If an instance reboots (intentionally or unintentionally), data in the instance store persists
* Instance store backed Instances cannot be stopped, as when stopped and started AWS does not guarantee the instance would be launched in the same host and hence the data is lost
* AMI creation requires usage on AMI tools and needs to be executed from within the running instance
* Instance store backed Instances cannot be upgraded
* For EC2 instance store-backed instances AWS recommends to:

1. Distribute the data on the instance stores across multiple AZs

2. Back up critical data from the instance store volumes to persistent storage on a regular basis

* Data on Instance store volume is LOST in following scenarios:

1. Underlying disk drive fails

2. Instance stops

3. Instance terminates

4. Instance hibernates

Therefore, do not rely on instance store for valuable, long-term data.

**Amazon Elastic Block Store (EBS)**

* An “EBS-backed” instance means that the root device for an instance launched from the AMI is an EBS volume created from an EBS snapshot
* An EBS volume behaves like a raw, unformatted, external block device that can be attached to a single instance and are not physically attached to the Instance host computer (more like a network attached storage).
* Volume persists independently from the running life of an instance. After an EBS volume is attached to an instance, you can use it like any other physical hard drive.
* EBS volume can be detached from one instance and attached to another instance.
* EBS volumes can be created as encrypted volumes using the EBS encryption feature.
* EBS is block store which is separately attached to EC2. Also its design such a way that it will be replicated within its availability zone so it provides high availability and durability.
* And the additional advantage of it is, you can have back-ups for EBS by creating Snapshots which is not possible instance store. So that whenever you want to retrieve the data you can just create the EBS volume from the snapshot.

**Key points for EBS backed Instance**

* Boot time is very fast usually less than a min.
* Can be selected as Root Volume and attached as additional volumes.
* EBS backed Instances can be of maximum 16TiB volume size depending upon the OS.
* EBS volume can be attached as additional volumes when the Instance is launched and even when the Instance is up and running.
* When EBS-backed instance is in a stopped state, various instance– and volume-related tasks can be done for e.g. you can modify the properties of the instance, you can change the size of your instance or update the kernel it is using, or you can attach your root volume to a different running instance for debugging or any other purpose.
* EBS volumes are AZ scoped and tied to a single AZ in which created.
* EBS volumes are automatically replicated within that zone to prevent data loss due to failure of any single hardware component.
* AMI creation is easy using a Single command.
* EBS backed Instances can be upgraded for instance type, Kernel, RAM disk and user data.
* Data on the EBS volume is LOST in following scenarios:

1. For EBS Root volume, if Delete on termination flag is enabled (enabled, by default).

2. For attached EBS volumes, if the Delete on termination flag is enabled (disabled, by default).

* Data on EBS volume is NOT LOST in following scenarios:

1. Reboot on the Instance.

2. Stopping an EBS-backed instance.

3. Termination of the Instance for the additional EBS volumes. Additional EBS volumes are detached with their data intact.

**Scalability, elasticity, and Bootstrapping**

**Scalability**, each and every Cloud is designed to gain infinite scalability. Even though the Cloud is scalable, you cannot take advantage of the scalability if your infrastructure is not scalable at any point. You have to conclude your components, which requires scalability, and you have to find those parts in which on-demand scaling won't work for your business. You have to design your application in order to get maximum output to capture the market and leverage the scalable infrastructure of the Cloud.

The characteristics of a proper scalable application are as follows:

• Growing capitals result in a comparative upsurge in the recital of performance

• A scalable service is accomplished because of conduct heterogeneity

• A climbable service is operationally effective

• An ascendable service is robust

• In general, a scalable service should get more budget with respect to actual one when it is required

These are characteristics that must be converted into an intrinsic part of the business application and if the architecture design is built with the preceding characteristics, then both the architecture and infrastructure will work in an organized manner to give you the scalability you are looking for.

**Elasticity** is one of the essential properties of Cloud computing in today's fast-paced demand-growing world. It is the power to scale computing capitals up and down in a straightforward manner and with nominal chafing. It is important to comprehend that elasticity will eventually initiate maximum of the paybacks of the Cloud. As a Cloud engineer, you need to adopt the concept of elasticity and implement it in your application architecture in order to get the best value from the Cloud and its services.

**Bootstrapping -** when you boot or spin the instances from an AMI at runtime on air. At the time of booting the instance, we will download and configure the scripts and codes necessary as per the role requirement and work automatically as defined in the scripts.

The following are the advantages of bootstrapping your instances:

• Reconstruct the (development, test, and production) environment with little snaps and nominal struggle

• The preceding instances govern your abstract Cloud-based capitals

Condense human-induced deployment mistakes

• Generates a self-healing and self-discoverable working environment, which can be more robust to hardware failure

**Identity and Access Management**

AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS resources. It enables you to create and control services for user authentication or limit access to a certain set of people who use your AWS resources. You can create groups and allow those users or groups to access some servers, or you can deny them access to the service.

**With IAM**, Organizations can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.

**Without IAM**, Organizations with multiple users must either create multiple user accounts, each with its own billing and subscriptions to AWS products or share an account with a single security credential. Without IAM, you also don't have control about the tasks that the users can do.

IAM includes the following features:

• Complete control of users, groups, and security credentials

• Control of access based on user role, and security tokens

• Control over shared AWS resources

• Authorizations based on organizational groups and users

• Centralized networking controls

**Accessing IAM**

You can work with AWS Identity and Access Management (AWS IAM) by using the following methods:

• Using the AWS Management Console

• Using the AWS command-line interface

• Using the AWS API

Using any of the preceding access methods, one can accomplish IAM capitals, by:

• Creating users/groups and assigning permissions to them

• Creating security credentials (roles and policies) for your users

• Assigning passwords to your users and restricting them for particular services

**How Does IAM Work?**

The IAM workflow includes the following six elements:

**A principal** is an entity that can perform actions on an AWS resource. A user, a role or an application can be a principal.

**Authentication** is the process of confirming the identity of the principal trying to access an AWS product. The principal must provide its credentials or required keys for authentication.

**Request:** A principal sends a request to AWS specifying the action and which resource should perform it.

**Authorization**: By default, all resources are denied. IAM authorizes a request only if all parts of the request are allowed by a matching policy. After authenticating and authorizing the request, AWS approves the action.

**Actions** are used to view, create, edit or delete a resource.

**Resources:** A set of actions can be performed on a resource related to your AWS account.

**Components of IAM**

There are other basic components of IAM. First, we have the user; many users together form a group. Policies are the engines that allow or deny a connection based on policy. Roles are temporary credentials that can be assumed to an instance as needed.

**Users**

An IAM user is an identity with an associated credential and permissions attached to it. This could be an actual person who is a user, or it could be an application that is a user. With IAM, you can securely manage access to AWS services by creating an IAM user name for each employee in your organization. Each IAM user is associated with only one AWS account. By default, a newly created user is not authorized to perform any action in AWS. The advantage of having one-to-one user specification is that you can individually assign permissions to each user.

**Groups**

A collection of IAM users is an IAM group. You can use IAM groups to specify permissions for multiple users so that any permissions applied to the group are applied to the individual users in that group as well. Managing groups is quite easy. You set permissions for the group, and those permissions are automatically applied to all the users in the group. If you add another user to the group, the new user will automatically inherit all the policies and the permissions already assigned to that group. This lessens the administrative burden.

**Policies**

An IAM policy sets permission and controls access to AWS resources. Policies are stored in AWS as JSON documents. Permissions specify who has access to the resources and what actions they can perform. For example, a policy could allow an IAM user to access one of the buckets in Amazon S3.

The policy would contain the following information:

Who can access it

What actions that user can take

Which AWS resources that user can access

When they can be accessed

* There are two types of policies: managed policies and inline policies.

**A managed policy** is a default policy that you attach to multiple entities (users, groups, and roles) in your AWS account. Managed policies, whether they are AWS-managed or customer-managed, are stand-alone identity-based policies attached to multiple users and/or groups.

**Inline policies** are policies that you create that are embedded directly into a single entity (user, group or role).

**Roles**

An IAM role is a set of permissions that define what actions are allowed and denied by an entity in the AWS console. It is similar to a user in that it can be accessed by any type of entity (an individual or AWS service). Role permissions are temporary credentials.