**Brief Notes AWS (SAA-02)**

**Cloud Computing**

* Cloud computing is the on-demand availability of [computer](https://en.wikipedia.org/wiki/Computer" \o "Computer) [system resources](https://en.wikipedia.org/wiki/System_resource" \o "System resource), especially data storage ([cloud storage](https://en.wikipedia.org/wiki/Cloud_storage" \o "Cloud storage)) and [computing power](https://en.wikipedia.org/wiki/Computing_power" \o "Computing power), without direct active management by the user. The term is generally used to describe [data centers](https://en.wikipedia.org/wiki/Data_center" \o "Data center) available to many users over the [Internet](https://en.wikipedia.org/wiki/Internet" \o "Internet). Large clouds, predominant today, often have functions [distributed](https://en.wikipedia.org/wiki/Distributed_computing" \o "Distributed computing) over multiple locations from central [servers](https://en.wikipedia.org/wiki/Server_(computing)" \o "Server (computing)). If the connection to the user is relatively close, it may be designated an [edge server](https://en.wikipedia.org/wiki/Edge_server" \o "Edge server).
* Clouds may be limited to a single [organization](https://en.wikipedia.org/wiki/Organization" \o "Organization) (enterprise clouds), or be available to multiple organizations (public cloud).
* The availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of [hardware virtualization](https://en.wikipedia.org/wiki/Hardware_virtualization" \o "Hardware virtualization), [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture" \o "Service-oriented architecture) and [autonomic](https://en.wikipedia.org/wiki/Autonomic_computing" \o "Autonomic computing) and [utility computing](https://en.wikipedia.org/wiki/Utility_computing" \o "Utility computing) has led to growth in cloud computing.

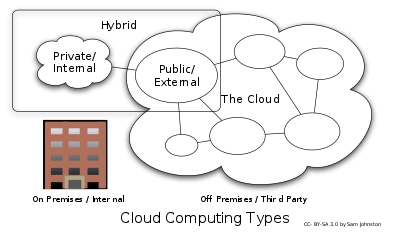
**Service Models**

Though [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture" \o "Service-oriented architecture) advocates "Everything as a service" (with the acronyms EaaS or XaaS or simply [aas](https://en.wikipedia.org/wiki/As_a_service" \o "As a service)), cloud-computing providers offer their "services" according to different models.

* **Infrastructure as a service (IaaS) -** refers to online services that provide high-level [APIs](https://en.wikipedia.org/wiki/Api" \o "Api) used to [abstract](https://en.wikipedia.org/wiki/Abstraction_(computer_science)" \o "Abstraction (computer science)) various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup, etc.
* **Platform as a service (PaaS) -** provides a [platform](https://en.wikipedia.org/wiki/Computing_platform" \o "Computing platform) allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.
* **Software as a service (SaaS) -** is a [software licensing](https://en.wikipedia.org/wiki/Software_licensing" \o "Software licensing) and [delivery](https://en.wikipedia.org/wiki/Software_delivery" \o "Software delivery) model in which [software](https://en.wikipedia.org/wiki/Software" \o "Software) is licensed on a [subscription](https://en.wikipedia.org/wiki/Subscription" \o "Subscription) basis and is centrally [hosted](https://en.wikipedia.org/wiki/Internet_hosting_service" \o "Internet hosting service). It is sometimes referred to as "on-demand software", and was formerly referred to as "software plus services" by [Microsoft](https://en.wikipedia.org/wiki/Microsoft" \o "Microsoft). SaaS applications are also known as on-demand software and Web-based/Web-hosted software.
* **Mobile "backend" as a service (MBaaS) -** also known as backend as a service (BaaS), [web app](https://en.wikipedia.org/wiki/Web_app" \o "Web app) and [mobile app](https://en.wikipedia.org/wiki/Mobile_app" \o "Mobile app) developers are provided with a way to link their applications to [cloud storage](https://en.wikipedia.org/wiki/Cloud_storage" \o "Cloud storage) and cloud computing services with [application programming interfaces](https://en.wikipedia.org/wiki/Application_programming_interface" \o "Application programming interface) (APIs) exposed to their applications and custom [software development kits](https://en.wikipedia.org/wiki/Software_development_kit" \o "Software development kit) (SDKs). Services include user management, [push notifications](https://en.wikipedia.org/wiki/Push_technology" \o "Push technology), integration with [social networking services](https://en.wikipedia.org/wiki/Social_networking_service" \o "Social networking service) and more.
* **Serverless computing -**  is a cloud computing code [execution](https://en.wikipedia.org/wiki/Execution_(computing)" \o "Execution (computing)) model in which the cloud provider fully manages starting and stopping [virtual machines](https://en.wikipedia.org/wiki/Virtual_machines" \o "Virtual machines) as necessary to serve requests, and requests are billed by an abstract measure of the resources required to satisfy the request, rather than per virtual machine, per hour
* **Function as a service (FaaS) -** is a service-hosted remote procedure call that leverages serverless computing to enable the deployment of individual functions in the cloud that run in response to events. FaaS is included under the broader term [serverless computing](https://en.wikipedia.org/wiki/Serverless_computing" \o "Serverless computing), but the terms may also be used interchangeably.

**Deployment models**

* **Private cloud -** Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third party, and hosted either internally or externally. Undertaking a private cloud project requires significant engagement to virtualize the business environment and requires the organization to revaluate decisions about existing resources. It can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities. Self-run [data centers](https://en.wikipedia.org/wiki/Data_center" \o "Data center) are generally capital intensive.
* **Public cloud -** Cloud services are considered "public" when they are delivered over the public Internet, and they may be offered as a paid subscription, or free of charge.[[99]](https://en.wikipedia.org/wiki/Cloud_computing" \l "cite_note-99) Architecturally, there are few differences between public- and private-cloud services, but security concerns increase substantially when services (applications, storage, and other resources) are shared by multiple customers. Most public-cloud providers offer direct-connection services that allow customers to securely link their legacy data centers to their cloud-resident applications.
* **Hybrid cloud -** is a composition of a public cloud and a private environment, such as a private cloud or on-premises resources, that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources. [Gartner](https://en.wikipedia.org/wiki/Gartner" \o "Gartner) defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers.



**AWS Identity and Access Management (IAM)**

* AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.
* IAM is a feature of your AWS account offered at no additional charge. You will be charged only for use of other AWS services by your users.
* 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**

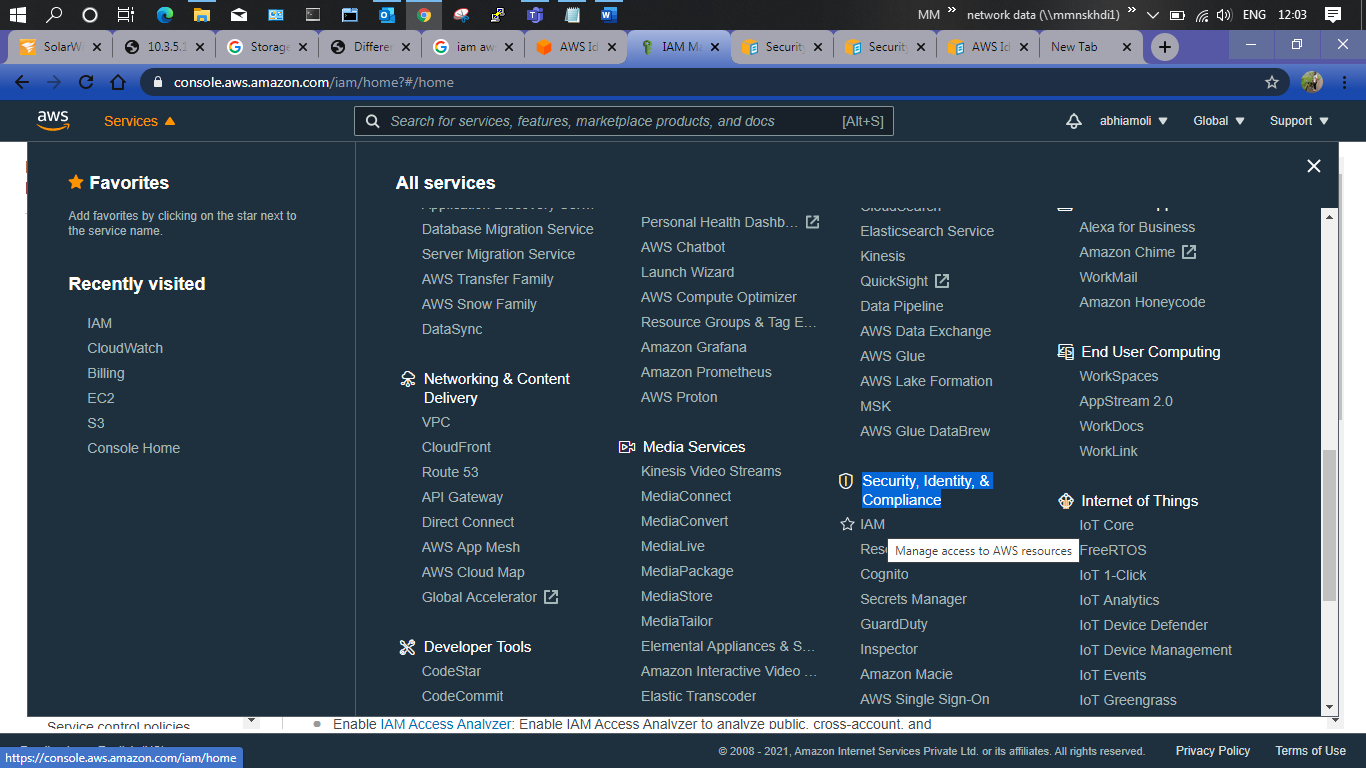
* **Users**- An IAM user is an identity with an associated credential and permissions attached to it. This could be an actual person who is a user, or it could be an application that is a user. With IAM, you can securely manage access to AWS services by creating an IAM username for each employee in your organization. Each IAM user is associated with only one AWS account.
* **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.
* **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. 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 like a user in that it can be accessed by any type of entity (an individual or AWS service). Role permissions are temporary credentials.

**Features of IAM**

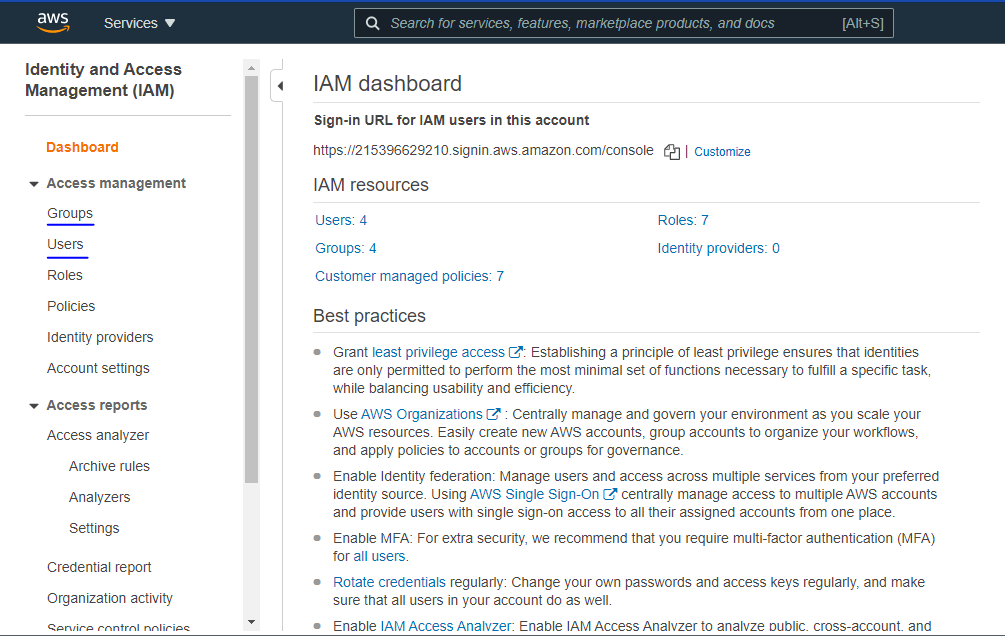
* **Shared access to the AWS account**. The main feature of IAM is that it allows you to create separate usernames and passwords for individual users or resources and delegate access.
* **Granular permissions**. Restrictions can be applied to requests. For example, you can allow the user to download information, but deny the user the ability to update information through the policies.
* **Multifactor authentication (MFA)**. IAM supports MFA, in which users provide their username and password plus a one-time password from their phone—a randomly generated number used as an additional authentication factor.
* **Identity Federation**. If the user is already authenticated, such as through a Facebook or Google account, IAM can be made to trust that authentication method and then allow access based on it. This can also be used to allow users to maintain just one password for both on-premises and cloud environment work.
* **Free to use**. There is no additional charge for IAM security. There is no additional charge for creating additional users, groups, or policies.
* **PCI DSS compliance**. The Payment Card Industry Data Security Standard is an information security standard for organizations that handle branded credit cards from the major card schemes. IAM complies with this standard.
* **Password policy**. The IAM password policy allows you to reset a password or rotate passwords remotely. You can also set rules, such as how a user should pick a password or how many attempts a user may make to provide a password before being denied access.

**Creating an IAM user:**

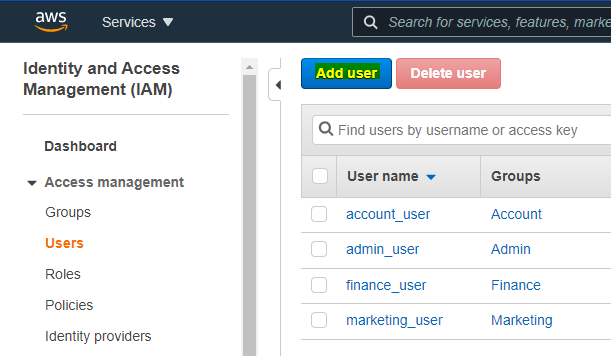
1. **Services 🡪Security, Identity, & Compliance 🡪 IAM**



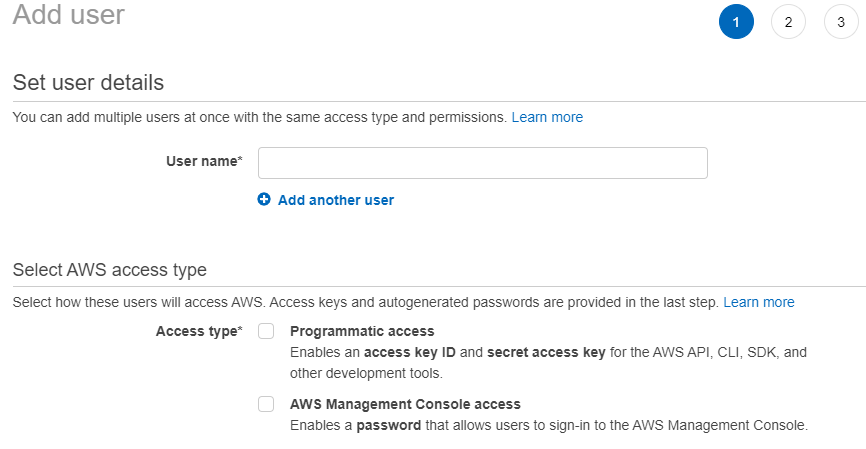
1. **IAM Dashboard**-



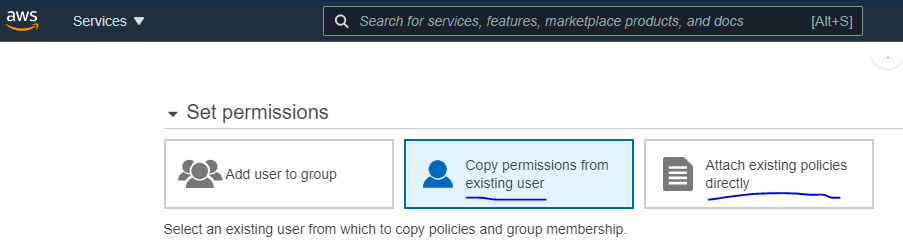
1. **Creating IAM User 🡪 Click on Users 🡪 Add User 🡪**



1. **In Add User Console 🡪 Enter Username 🡪 Select AWS Access Type**
   1. Programmatic Access – Used for AWS API, CLI and other services.
   2. AWS Management Console Access – AWS Console User



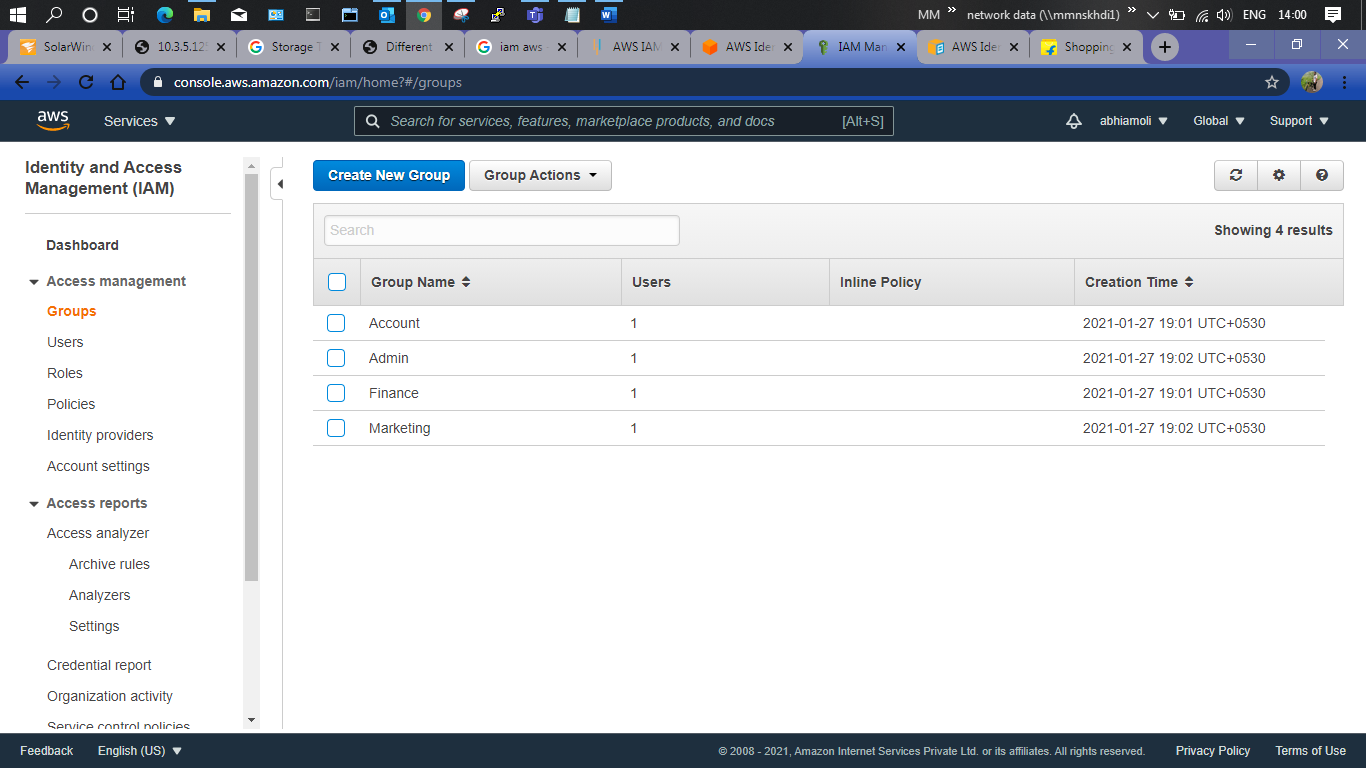
1. Select **AWS Access Type 🡪Enter Console Password 🡪 Click Next: Permissions**
2. Set specified permissions to the users 🡪Use Existing Policy or create new policy using AWS Policy Generator.



1. If this user needs to be added in some group, either **create new group** or add the user in **existing group 🡪 Than click Next 🡪Add Tags** (this is optional) **🡪Create user**.

**Creating an IAM Group:**

1. **Click on Groups from Left Panel 🡪 Click on Create New Group**



1. **Enter Group Name 🡪Click Next 🡪 Attach the Policies equivalent to the group🡪Click Next 🡪Click Create Group.**

**AWS Simple Storage Service (S3)**

* Amazon S3 (Simple Storage Service) provides object storage, which is built for storing and recovering any amount of information or data from anywhere over the internet.
* It provides this storage through a web services interface. While designed for developers for easier web-scale computing, it provides 99.999999999 percent durability and 99.99 percent availability of objects.
* It can also store computer files up to 5 terabytes in size.

**AWS S3 Benefits:**

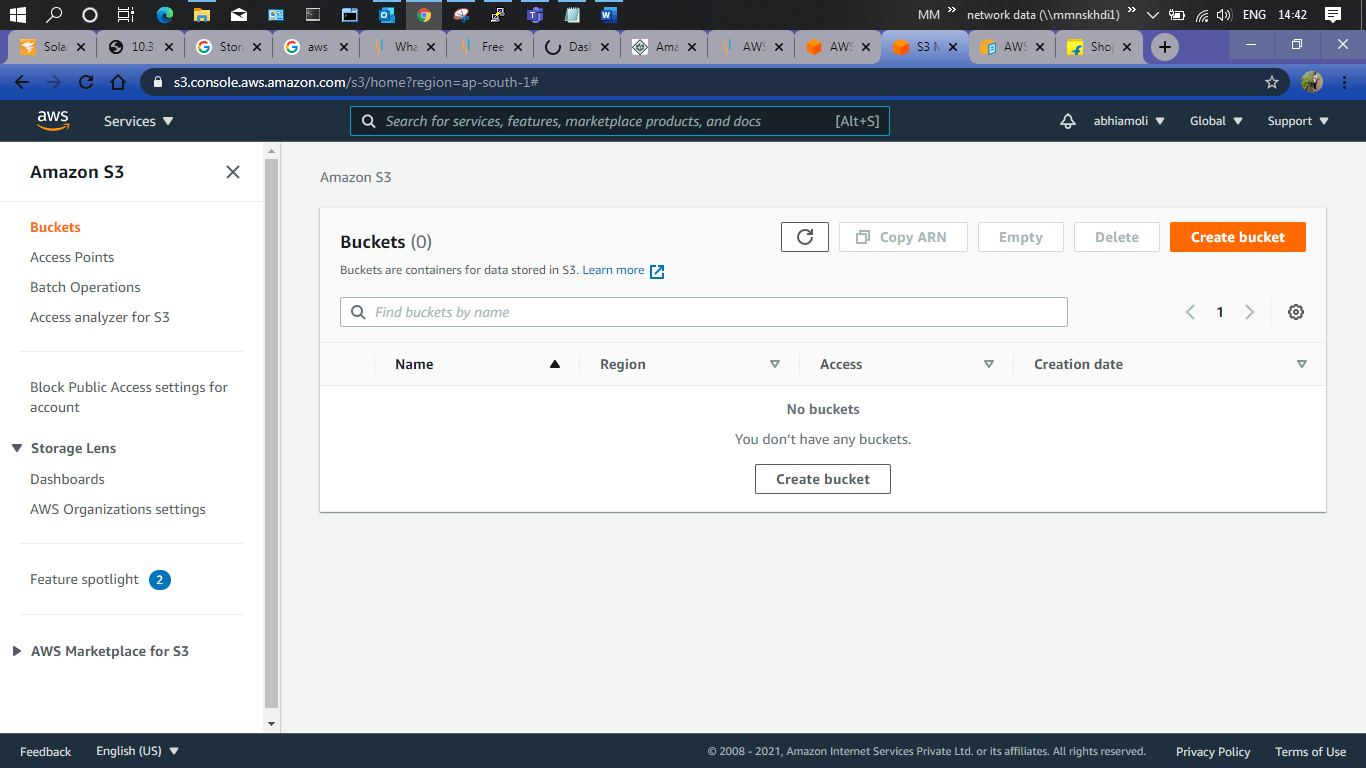
* Durability
* Low cost
* Scalability
* Availability
* Security
* Flexibility
* Simple data transfer

**AWS Buckets and Objects:**

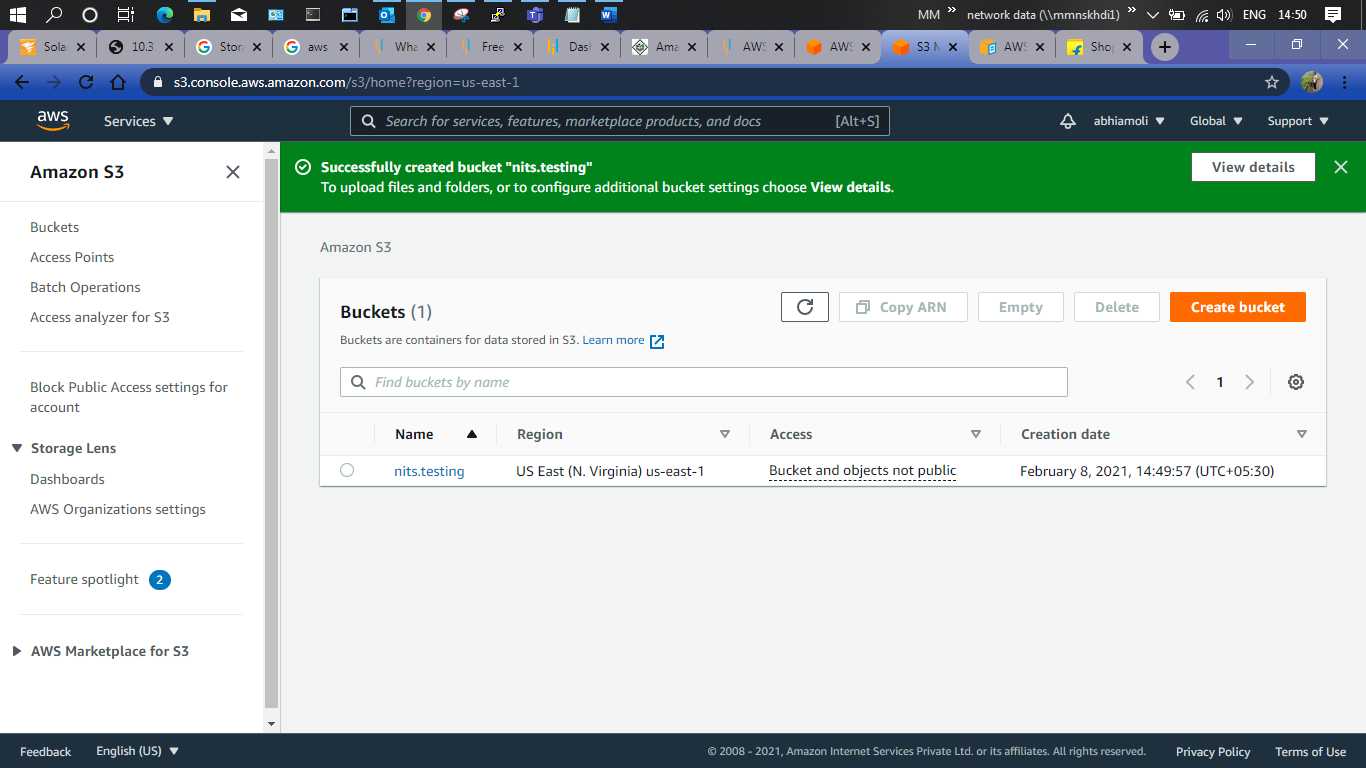
* An object consists of data, key (assigned name), and metadata. A bucket is used to [store objects](https://aws.amazon.com/s3/features/" \t "_blank). When data is added to a bucket, Amazon S3 creates a unique version ID and allocates it to the object.

**Creating Buckets:**

1. **Services 🡪 Storage 🡪 S3 🡪 Buckets Dashboard**



1. **Click Create Bucket 🡪 Enter Unique Bucket Name (check the rules for bucket naming) 🡪Choose the Region 🡪Leave Remaining setting as default 🡪 Click Create Bucket.**



1. Click on Bucket Name 🡪 Here you can create folder or directly Add Files to your bucket.
2. To access the file from public or any interface, you need to provide necessary permissions to bucket as well as file.

**Amazon S3 Storage Classes:**

1. **S3 Standard –** 
   1. Ideal for performance-sensitive use cases and frequently accessed data
   2. Default storage class if none of the storage class is specified during upload.
   3. Default storage class if none of the storage class is specified during upload.
   4. Default storage class if none of the storage class is specified during upload.
   5. Most expensive storage class among all others.
2. **Intelligent-Tearing –** 
   1. Designed to optimize costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead**.**
   2. Automatically moves objects between two access tiers based on changing access patterns. It moves objects that have not been accessed for 30 consecutive days to the infrequent access tier. If an object in the infrequent access tier is accessed, it is automatically moved back to the frequent access tier.
   3. Delivers same low latency and high throughput performance of S3 Standard.
   4. Designed for 99.99% availability and 99.999999999% (11 9's) durability (same as Standard).
   5. There are no retrieval fees when using the S3 Intelligent-Tiering storage class, and no additional tiering fees when objects are moved between access tiers.
3. **S3 Standard-Infrequent Access (S3 Standard-IA) –**
   1. Optimized for long-lived and less frequently accessed data. i.e. It is for data that is accessed less frequently but requires rapid access when needed. Objects are available for real-time access.
   2. It has data stored redundantly across multiple geographically separated AZs and are resilient to the loss of an Availability Zone (3 AZs).
   3. Delivers same low latency and high throughput performance of S3 Standard.
   4. Designed for 99.99% availability and 99.999999999% (11 9's) durability (same as Standard).
   5. It is less expensive than S3 Standard storage, but you will be charged for a retrieval fee. So, they are most suitable for infrequently accessed data.
4. **S3 One Zone-Infrequent Access (S3 One Zone-IA) –** 
   1. Optimized for long-lived and less frequently accessed data. i.e. It is for data that is accessed less frequently but requires rapid access when needed. It is designed for long-lived and infrequently accessed data (like the Standard and Standard-IA storage class).
   2. Unlike other S3 Storage Classes which store data in a minimum of three Availability Zones (AZs), S3 One Zone-IA stores data in a single AZ and costs 20% less than S3 Standard-IA, which makes it less expensive than Standard-IA.
   3. Data stored in this storage class will be lost in the event of Availability Zone destruction.
   4. Ideal for a low-cost storage option for infrequently accessed data but do not require the availability and resilience of S3 Standard or S3 Standard-IA.
   5. Designed for 99.5% availability and 99.999999999% (11 9’s) durability in a single Availability Zone (same as Standard).
5. **Glacier**
   1. Low-cost design is ideal for long-term archiving.
   2. Configurable retrieval times, from minutes to hours.
   3. Data is resilient in the event of one entire Availability Zone destruction.
   4. Designed for 99.99% availability and 99.999999999% (11 9’s) durability in a multiple Availability Zone.
   5. Charges are levied for both the archive (Glacier rate) and the copy restored temporarily (RRS rate)
   6. Vault Lock feature enforces compliance via a lockable policy
   7. It has a minimum storage duration period of 90 days and can be accessed in as little as 1–5 minutes using expedited retrieval.
6. **Glacier Deep Archive**
   1. Lowest-cost storage class and supports long-term retention and digital preservation for data that will be retained for 7–10 years and may be accessed once or twice in a year.
   2. Lowest cost storage option in S3. Storage costs for Glacier Archive is less expensive than using the Glacier storage class.
   3. Designed for 99.99% availability and 99.999999999% (11 9’s) durability in a multiple Availability Zone (same as Glacier).
   4. Ideal alternative to magnetic tape libraries.
   5. It has a minimum storage duration period of 180 days and can be accessed in at a default retrieval time of 12 hours.
   6. Retrieval costs can be reduced by using bulk retrieval, which returns data within 48 hours.

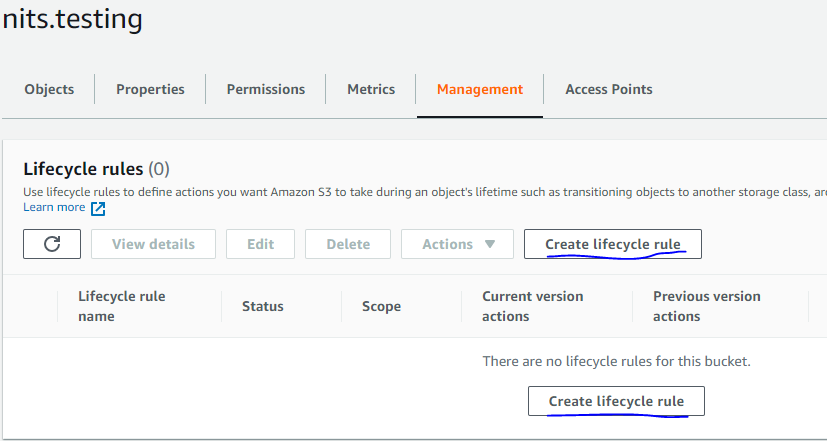
**S3 Glacier provides three types of Retrieval Options:**

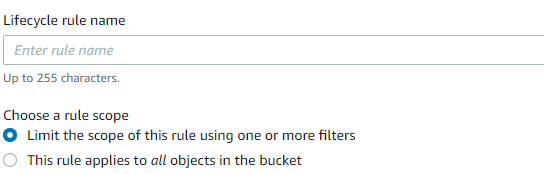
* **Expedited Retrieval —** It allows you to quickly access your data when occasional urgent requests for a subset of archives are required. For all but the largest archives (250 MB+), data accessed using Expedited retrievals are typically made available within 1–5 minutes for Glacier. It is not available for objects stored in the Glacier Deep Archive storage class.
* **Standard Retrieval** — It allows you to access any of your archives within several hours. This is the default option for retrieval requests that do not specify the retrieval option. Typically, retrievals of this type are implemented within 3–5 hours for objects stored in the Glacier storage class and within 12 hours for objects stored in the Deep Archive storage class respectively.
* Standard Retrieval — It allows you to access any of your archives within several hours. This is the default option for retrieval requests that do not specify the retrieval option. Typically, retrievals of this type are implemented within 3–5 hours for objects stored in the Glacier storage class and within 12 hours for objects stored in the Deep Archive storage class respectively.

**NOTE: All storage classes are designed for 99.999999999% (11 9’s) durability in a multiple Availability Zone (≥ 3 AZs). (except One Zone-IA, which is Single AZ)**

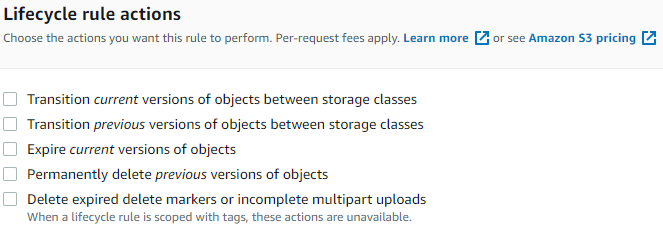
**AWS S3 Features**

* **Lifecycle Management -** Amazon S3 applies a set of rules that define the action to a group of objects. You can manage and store objects in a cost-effective manner. There are two types of actions:
  + **Transition Action -** With this action, you can choose to move objects to another storage class. With this, you can configure S3 to move your data between various storage classes on a defined schedule.
  + **Expiration Actions -** Here, S3 removes all objects within the bucket when a specified date or time in the object’s lifetime is reached.
* **Creating Lifecycle Rule –**

1. **Select the bucket 🡪 Go to Management TAB 🡪Select the Option Create Lifecycle Rule 🡪**
2. **Enter Rule Name 🡪 Select the Rule Scope**



1. **Select Lifecycle Rule Actions 🡪 Click Create Rule**



**Bucket Policy**

Bucket policy is an IAM policy where you can allow or deny permission to your Amazon S3 resources. With bucket policy, you also define security rules that apply to more than one file within a bucket. For example: If you do not want a user to access the “Test.Bucket” bucket, then with the help of JSON script, you can set permissions.

**Data Protection**

Amazon S3 protects your data using below methods:

* **Data encryption** - This refers to the protection of data while it’s being transmitted and at rest. It can happen in two ways, client-side encryption (data encryption at rest) and server-side encryption (data encryption in motion). **Select the Bucket 🡪Go to Properties Tab 🡪 Scroll Down to Default Encryption 🡪 Here you can make changes to Encryption settings.**
* **Versioning** - It is utilized to preserve, recover, and restore an early version of every object you store in your AWS S3 bucket. Unintentional erases or overwriting of objects can easily be managed with versioning. For example, in a bucket, it is possible to have objects with the same key name but different version IDs. **Select the Bucket** 🡪 **Go to Properties Tab 🡪 Scroll down to Bucket Versioning 🡪** Here you can enable Versioning.
* **Cross-region Replication** - Cross-region replication provides automatic copying of every object uploaded to your buckets (source and destination bucket) in different AWS regions. Versioning needs to be turned on to enable CRR.
  + **Select the Bucket** 🡪 **Go to Management Tab 🡪 Scroll down to Replication Rules 🡪 Create Replication Rule**
  + **Enter Replication rule name 🡪 Check the source bucket details and rule scope 🡪 Than enter Destination Bucket details 🡪 Select IAM Role 🡪 Check Additional Replication Options 🡪Save the Rule.**
* **Transfer Acceleration** - This enables fast, easy, and secure transfers of files over long distances between your client and S3 bucket. The edge locations around the world provided by Amazon CloudFront are taken advantage of by transfer acceleration. It works by carrying data over an optimized network bridge that keeps running between the AWS Edge Location (closest region to your clients) and your Amazon S3 bucket.
  + **Go to Services 🡪 Under Networking & Content Delivery 🡪 Select Cloud Front**
  + **Create Distribution 🡪 Get Started 🡪 Enter Origin Domain Name (Your Bucket Name) 🡪 Check other setting and do the changes as per your requirement 🡪 Click Create Distribution**

**AWS Elastic Compute Cloud (EC2)**

* Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.
* You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
* Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

**Features of EC2**

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

instance, known as instance store volumes.

* Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as

Amazon EBS volumes.

* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones.
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your

instances using security groups.

* Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses.
* Metadata, known as tags, that you can create and assign to your Amazon EC2 resources.
* Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs).

**Why is AWS EC2 important?**

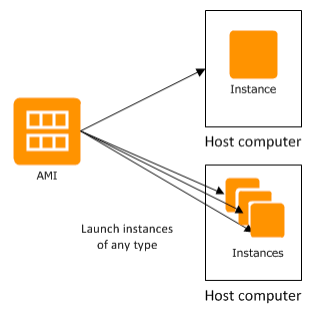
* You don’t require any hardware units
* Easily scalable (up or down)
* You only pay for what you use
* You have complete control
* Highly secure
* You can access your assets from anywhere in the world

**Pricing for Amazon EC2**

* When you sign up for AWS, you can get started with Amazon EC2 for free using the AWS Free Tier.
* Amazon EC2 provides the following purchasing options for instances:
  + **On-Demand Instances** - Pay for the instances that you use by the second, with no long-term commitments or upfront payments.
  + **Savings Plans** - You can reduce your Amazon EC2 costs by making a commitment to a consistent amount of usage, in USD per hour, for a term of 1 or 3 years.
  + **Reserved Instances -** You can reduce your Amazon EC2 costs by making a commitment to a specific instance configuration, including instance type and Region, for a term of 1 or 3 years.
  + **Spot Instances -** Request unused EC2 instances, which can reduce your Amazon EC2 costs significantly

**Instances & AMIs**

* An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications).
* From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud. You can launch multiple instances of an AMI, as shown in the following figure-



* Your instances keep running until you stop, hibernate, or terminate them, or until they fail. If an instance fails, you can launch a new one from the AMI.

**Instances**

* An instance is a virtual server in the cloud. Its configuration at launch is a copy of the AMI that you specified when you launched the instance.
* You can launch different types of instances from a single AMI. An instance type essentially determines the hardware of the host computer used for your instance. Each instance type offers different compute and memory capabilities. Select an instance type based on the amount of memory and computing power that you need for the application or software that you plan to run on the instance. For more information about the hardware specifications for each Amazon EC2 instance type.

**Storage for your Instance**

* The root device for your instance contains the image used to boot the instance. The root device is either an Amazon Elastic Block Store (Amazon EBS) volume or an instance store volume.
* Your instance may include local storage volumes, known as instance store volumes, which you can configure at launch time with block device mapping.
* After these volumes have been added to and mapped on your instance, they are available for you to mount and use. If your instance fails, or if your instance is stopped or terminated, the data on these volumes is lost; therefore, these volumes are best used for temporary data.
* To keep important data safe, you should use a replication strategy across multiple instances, or store your persistent data in Amazon S3 or Amazon EBS volumes.

**Stop & Terminate Instances**

* **Stop an instance** - When an instance is stopped, the instance performs a normal shutdown, and then transitions to a stopped state. All its Amazon EBS volumes remain attached, and you can start the instance again later. When an instance is in a stopped state, you can attach or detach Amazon EBS volumes. You can also create an AMI from the instance, and you can change the kernel, RAM disk, and instance type.
* **Terminate an instance -** When an instance is terminated, the instance performs a normal shutdown. The root device volume is deleted by default, but any attached Amazon EBS volumes are preserved by default, determined by each volume's **deleteOnTermination** attribute setting. The instance itself is also deleted, and you can't start the instance again later.

**Amazon Machine Images (AMIs)**

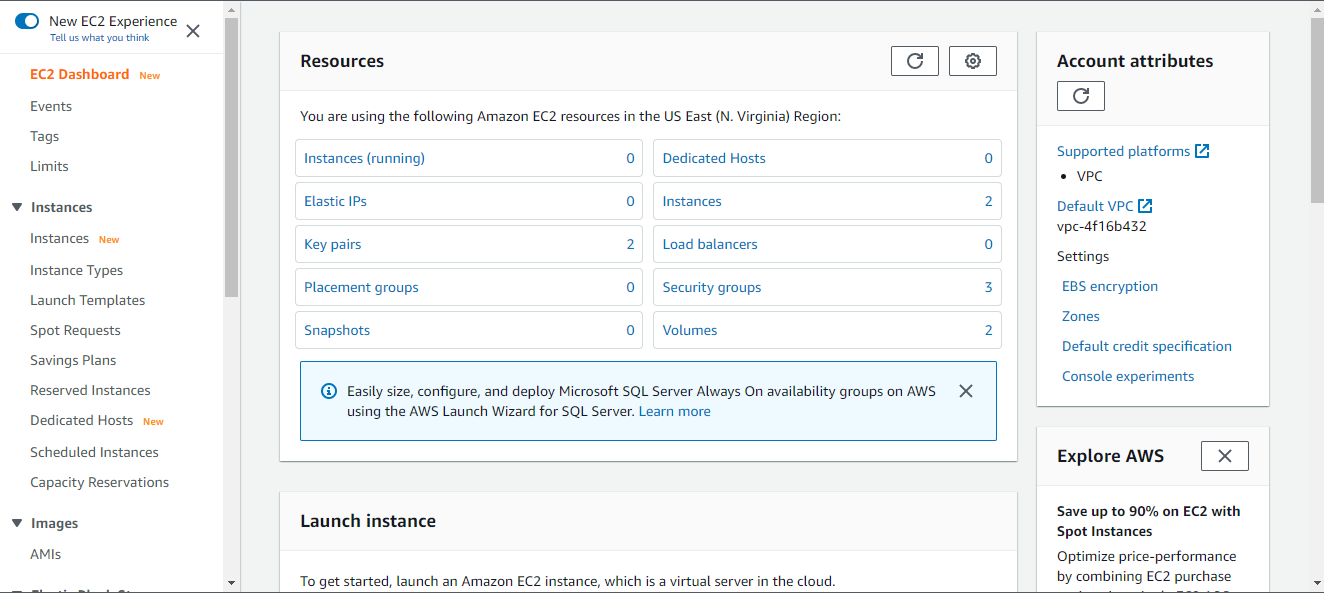
* Amazon Web Services (AWS) publishes many Amazon Machine Images (AMIs) that contain common software configurations for public use. In addition, members of the AWS developer community have published their own custom AMIs. You can also create your own custom AMI or AMIs; doing so enables you to quickly and easily start new instances that have everything you need.
* All AMIs are categorized as either backed by Amazon EBS, which means that the root device for an instance launched from the AMI is an Amazon EBS volume, or backed by instance store, which means that the root device for an instance launched from the AMI is an instance store volume created from a template stored in Amazon S3.

**Region & Zones**

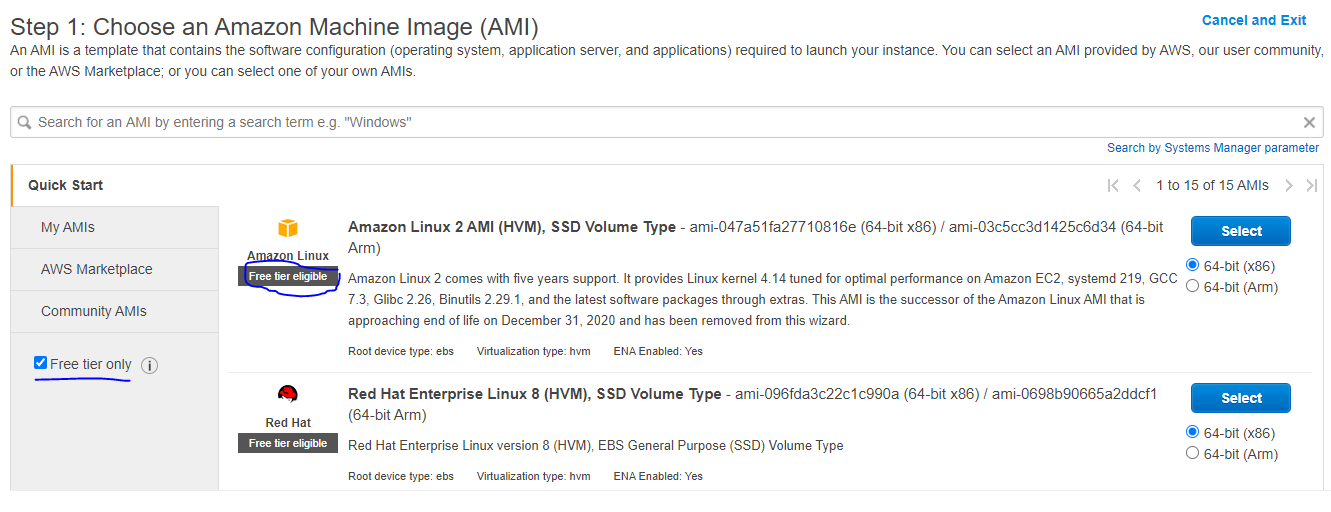
* Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones. Each Region is a separate geographic area.
  + Availability Zones are multiple, isolated locations within each Region.
  + Local Zones provide you the ability to place resources, such as compute and storage, in multiple locations closer to your end users.
  + AWS Outposts brings native AWS services, infrastructure, and operating models to virtually any data center, co-location space, or on-premises facility.
  + Wavelength Zones allow developers to build applications that deliver ultra-low latencies to 5G devices and end users. Wavelength deploys standard AWS compute and storage services to the edge of telecommunication carriers' 5G networks.

**To Launch an Instance:**

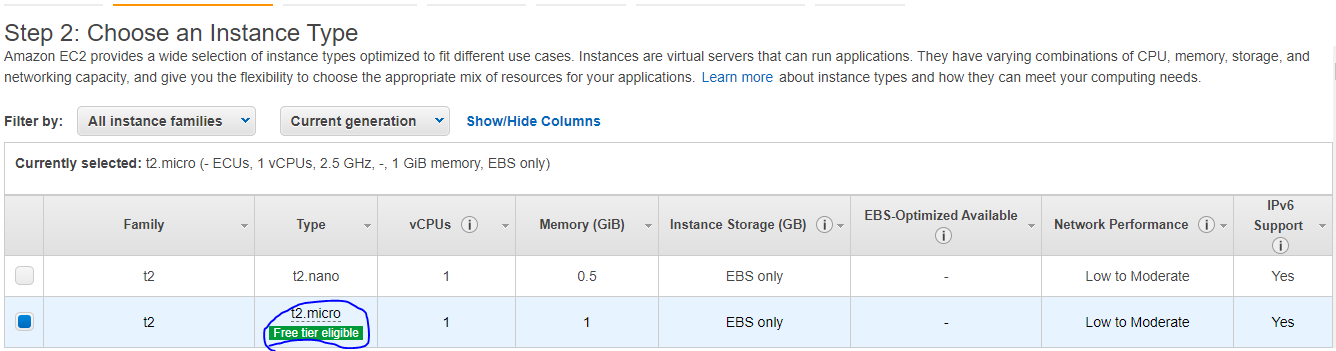
* **Login to AWS console 🡪Under Services go to Compute and select EC2 🡪 EC2 Dashboard will open**

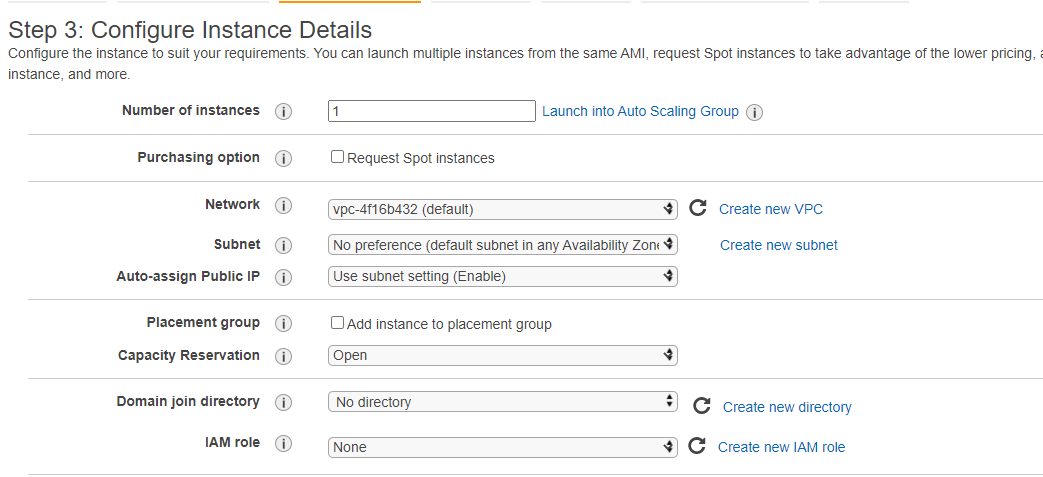


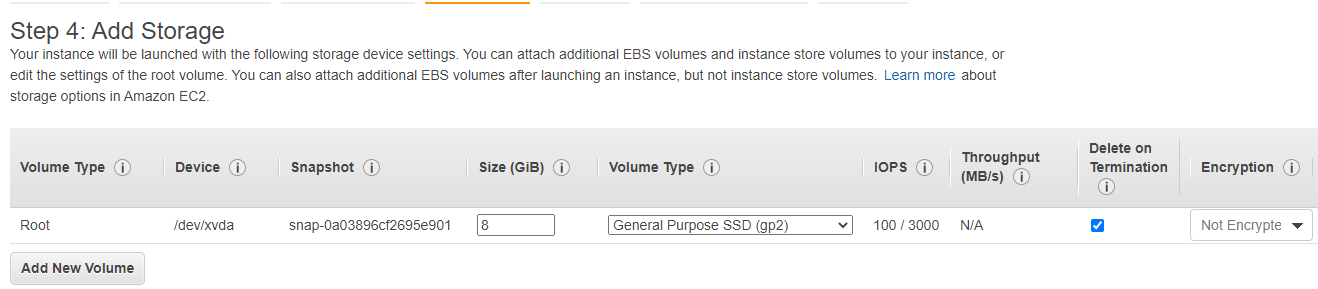
* **Select Instance 🡪Click on Launch Instance 🡪Choose AMI 🡪**



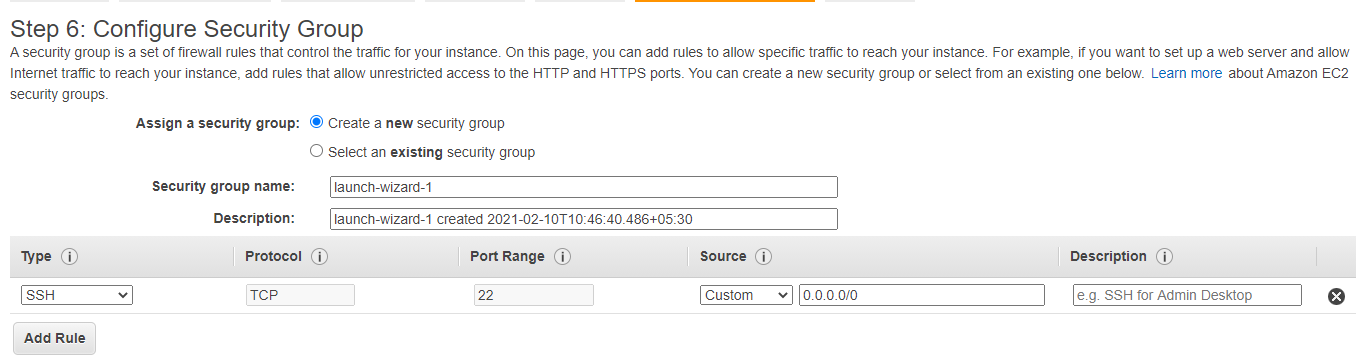
* **Choose an Instance Type (Check free tier while configuring)**



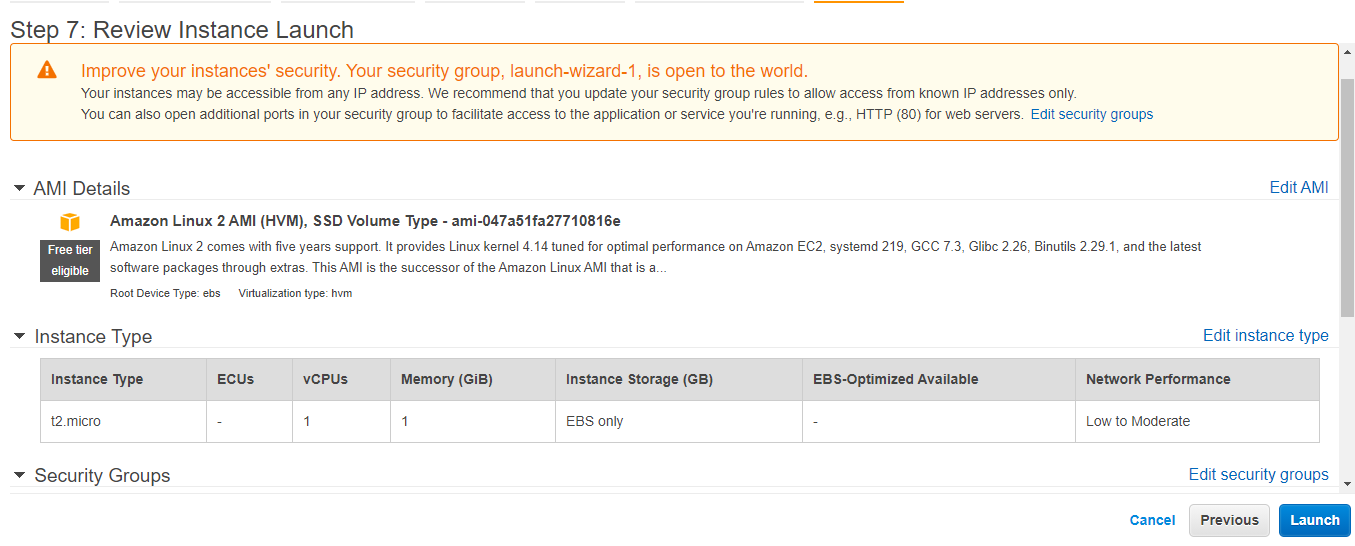
* **Configure Instance Details**
* **Add Storage –**



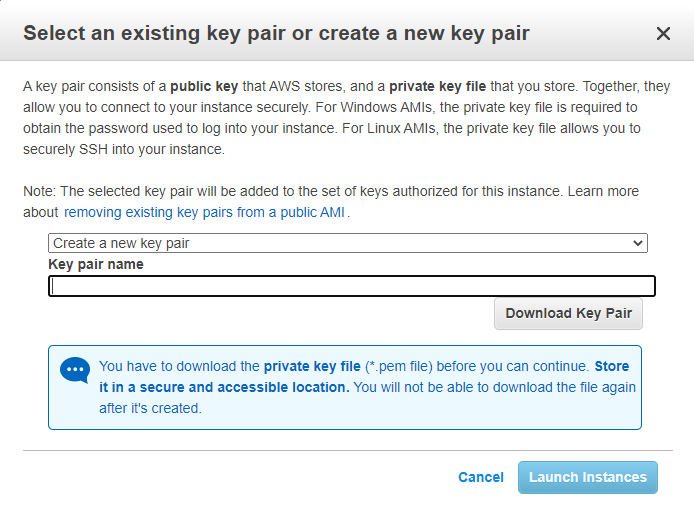
* **Configure Security Group – This acts as a firewall – (Either create new security group or use existing security group.)**



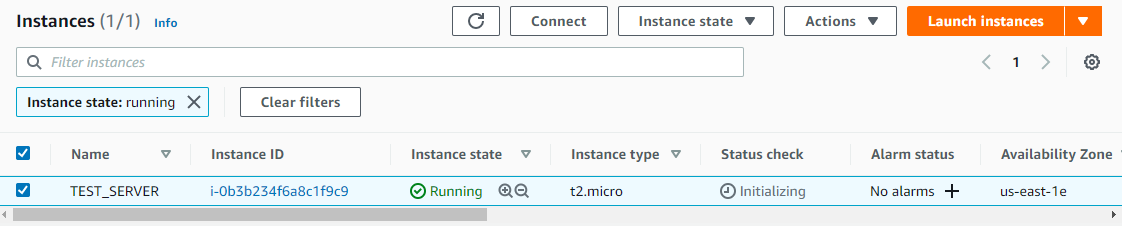
* **Review Instance Launch 🡪 Launch**

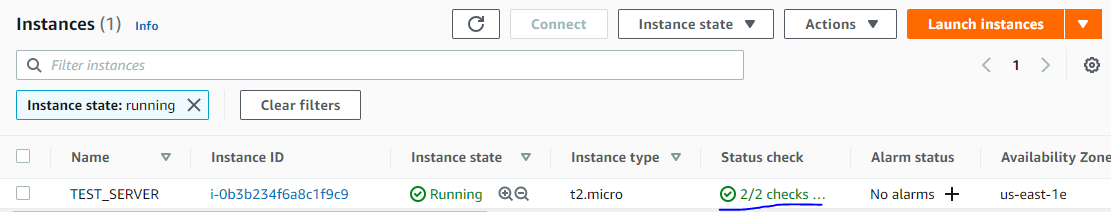


* **Before Launching an instance, AWS will instruct to download Key Pair, which will be used while launching actual virtual server (Linux or Windows). Enter Key Pair Name 🡪 Download key Pair (keep it safe) 🡪Launch Instance**



* **Go to EC2 Dashboard 🡪 Select Instances (Running) 🡪 Wait till System Check Status is Green –**





**EBS Storage**

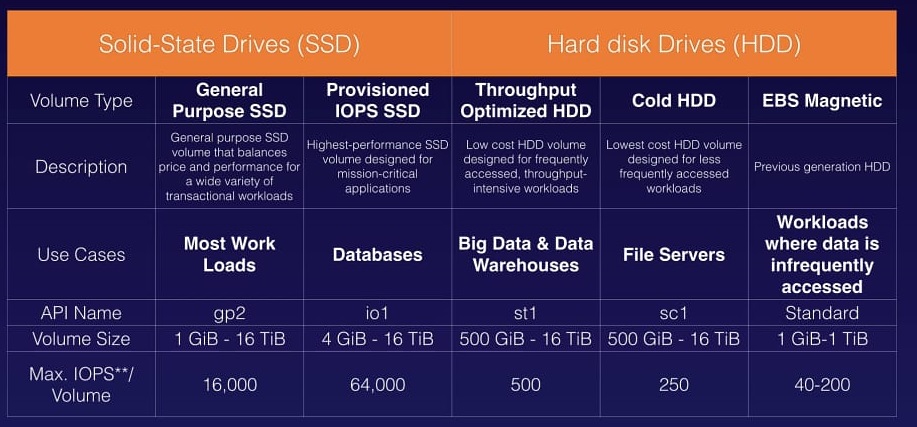
* Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance.
* You can create a file system on top of these volumes or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

**Features of EBS**

* You create an EBS volume in a specific Availability Zone, and then attach it to an instance in that same Availability Zone. To make a volume available outside of the Availability Zone, you can create a snapshot and restore that snapshot to a new volume anywhere in that Region.
* You can create your EBS volumes as encrypted volumes, to meet a wide range of data-at-rest encryption requirements for regulated/audited data and applications.

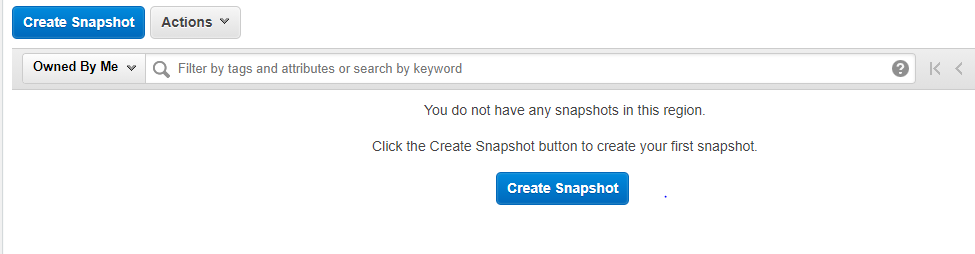
**EBS Volumes**

* General Purpose SSD (gp2 and gp3)
* Provisioned IOPS SSD (io1 and io2)
* Throughput Optimized HDD (st1)
* Cold HDD (sc1)
* Magnetic (standard)

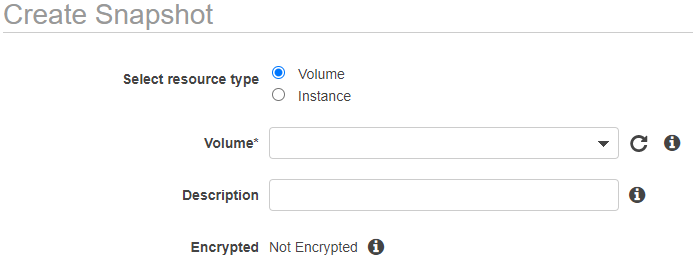


**EBS Snapshots**

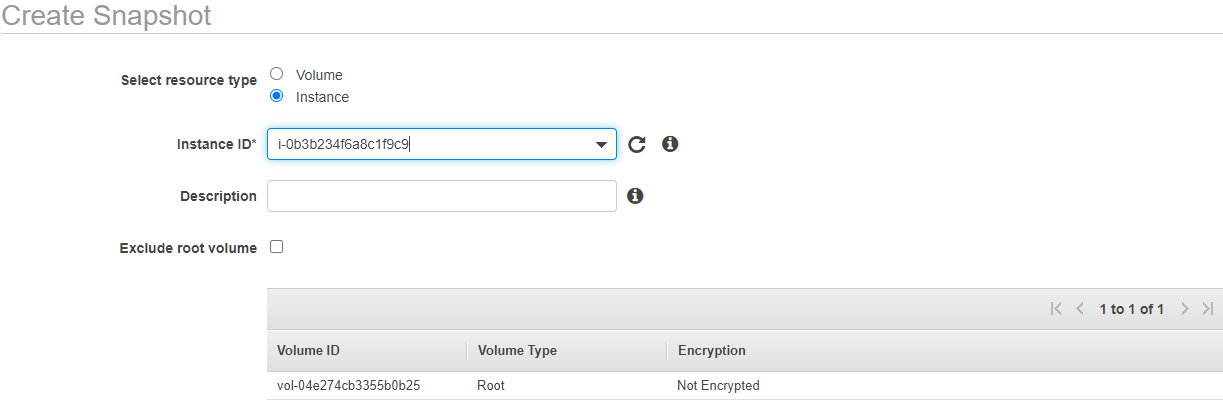
* You can back up the data on your Amazon EBS volumes to Amazon S3 by taking point-in-time snapshots. Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved.
* This minimizes the time required to create the snapshot and saves on storage costs by not duplicating data. Each snapshot contains all the information that is needed to restore your data (from the moment when the snapshot was taken) to a new EBS volume.
* When you create an EBS volume based on a snapshot, the new volume begins as an exact replica of the original volume that was used to create the snapshot.
* Steps to create Snapshot –
  + Select Snapshot from Left Panel 🡪



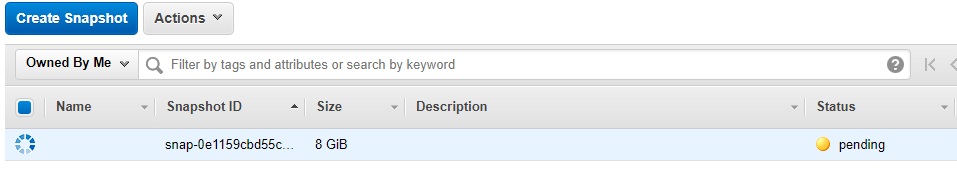
* + Click Create Snapshot🡪 Select the type of Snapshot to be taken (Volume or Instance)🡪 Enter Volume Details & Description.

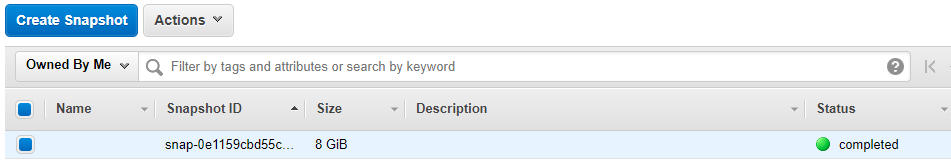


* + Select Instance 🡪Enter Instance Details & Description 🡪 Click Create Snapshot

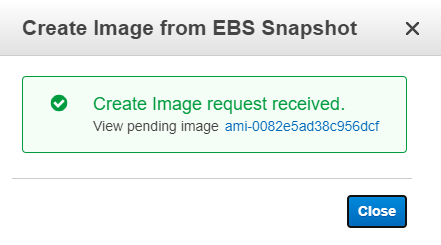


* + Go to Snapshot Dashboard 🡪 Check the Status of Snapshot 🡪 Wait till the Status turns from Pending to Completed

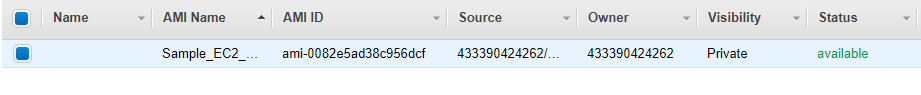




* + If an Image need to be created from the snapshot 🡪 Select a Snapshot 🡪 Select Actions 🡪 Click Create Image



* + Then check the created image in AMI Dashboard



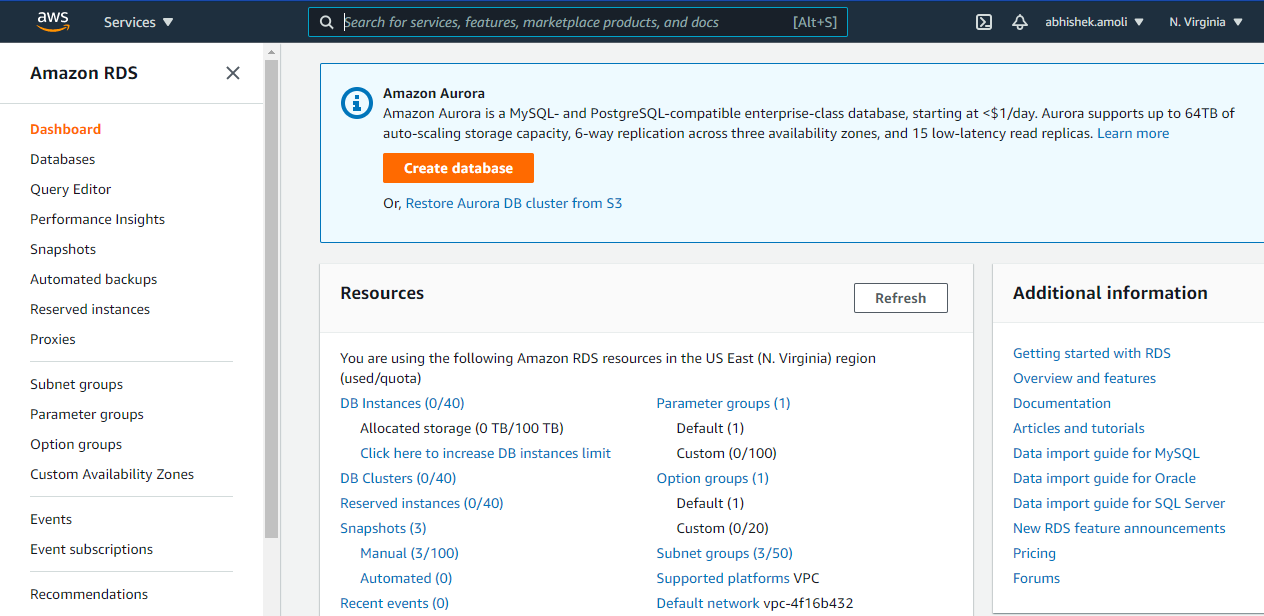
* + Now using the created image, you can replicate the instance to another Availability Zone in same region. And same can be done with the volume as well. These changes can be done under Configure Instance details.

**AWS Relational Database Service (RDS)**

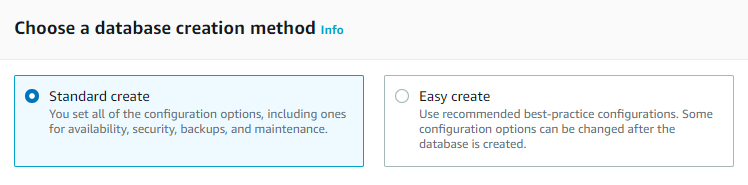
* It makes it easy to set-up and operate a relational database in the cloud.
* It provides a very cost-effective way to use industry’s leading RDBMS software as a managed service.
* You must subscribe AWS RDS web service and start using the RDBMS features after some initial configuration involving memory and CPU capacity allocation etc.
* As RDS is a managed service provided by AWS, we can expect that like other AWS services it will provide scalability, security, and cost effectiveness to the various RDBMS it provides. The database products available through AWS RDS are as listed below.
  + **MySQL** - Support versions for MySQL 5.5 to 5.7. Minor upgrades happen automatically without needing any involvement from the user.
  + **MariaDB** – Support versions for MariaDB from 10.0 to 10.2.
  + **Oracle** – Supports version 11g and 12c. You can use the oracle license provided by aws or bring your own license. The costing for these two are different.
  + **Microsoft SQL Server** – Supports version 200t to 2017. Also AWS supports the various editions like – Enterprise, Standard, Web and Express.
  + **PostgreSQL** – Supports version 9 to 11. Can be configured as a multi A-Z deployment with read replicas.
  + **Amazon Aurora** – This is Amazon’s own RDBMS. We will be covering it in a separate tutorial.
* Each of this Database software is offered as Software as a Service (saas) by providing following features.
  + Customization of CPU capacity, Memory allocation and IOPS (Input Output per second) for a database instance.
  + Manage software patching, failure, and recovery of the RDBMS software without any user intervention.
  + Allow manual or automated backup of the database using snapshots. Restore the database from these snapshots.
  + Provide high availability by creating a primary and secondary instance which are synchronous. In case of a failure of primary AWS RDS automatically fails over to secondary.
  + Put the databases in a virtual private cloud (VPC) and use AWS IAM (Identity and Access management) service to control access to the databases.
  + There are two purchase options for AWS RDS service. On-Demand Instances and Reserved Instances. For on-Demand instance you pay for every hour of usage while for Reserved instance you make a upfront payment for one year to three period time frame.

**Using RDS Service**

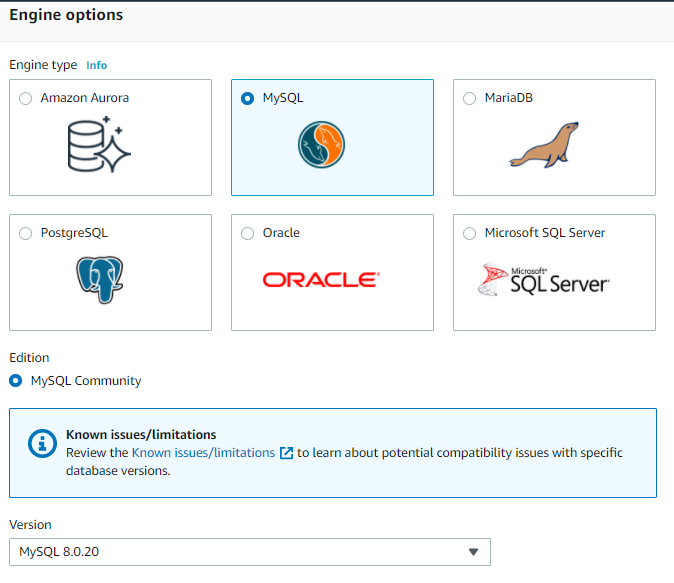
* Log in to AWS Console 🡪 Go to Services 🡪Database 🡪Select RDS
* Below Dashboard will display



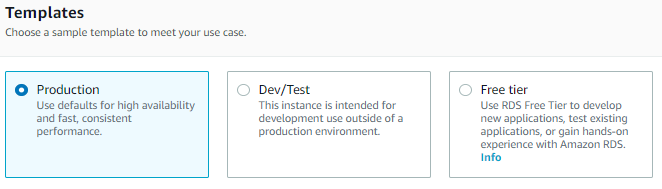
* Click on Create database –
  + Select Database creation method (Standard Create / Easy Create) 🡪



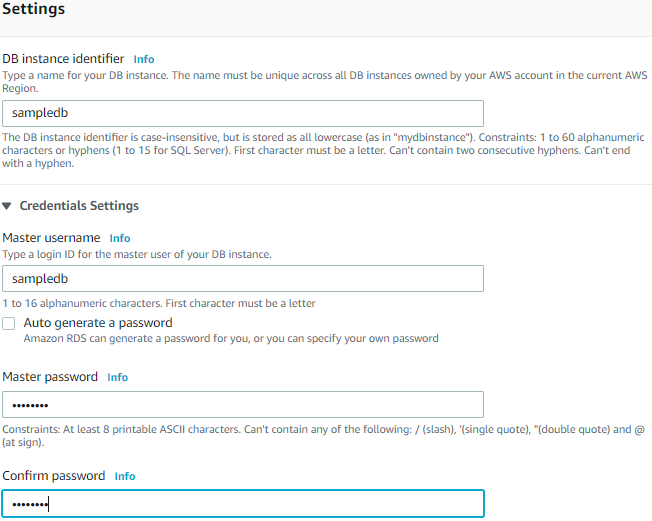
* + Select Engine Type (Database Engine), its edition and version 🡪



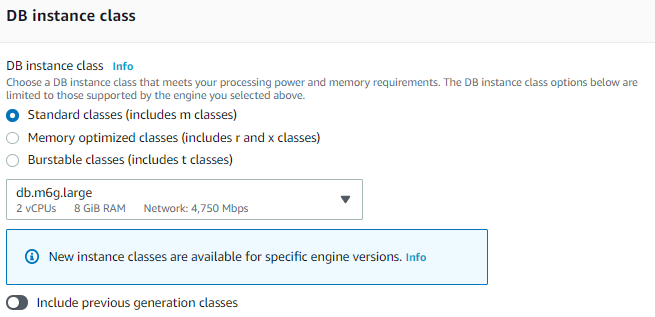
* + Select Templates (Production / Dev-Test / Free Tier) 🡪



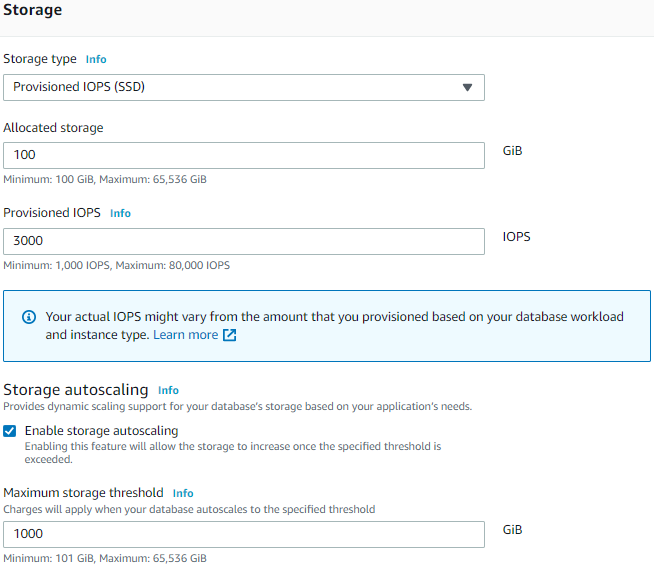
* + Enter Database Settings (db name, db username, db password) 🡪



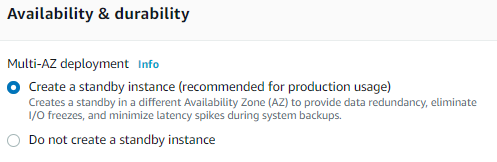
* + Select DB Instance Classes this will help in meeting performance expectations 🡪



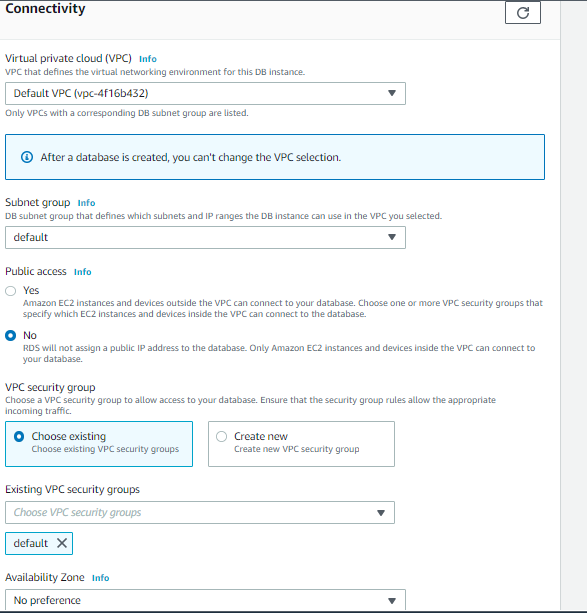
* + Select the Storage types, total storage, auto-scaling storage 🡪



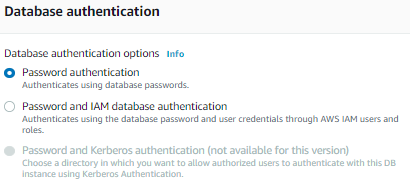
* + Select DB Availability & Durability 🡪



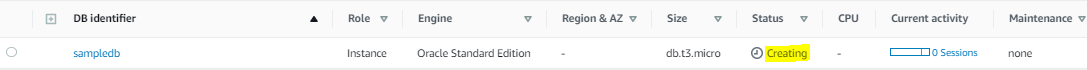
* + Select DB Connectivity (create new VPC / create new VPC security Group) 🡪



* + Select Database Authentication type 🡪



* + Then Select Create Database (Database creation will take 5-15 mins approx.)



**AWS RDS Components**

* DB Instances
  + They are the building blocks of RDS. It is an isolated database environment in the cloud, which can contain multiple user-created databases, and can be accessed using the same tools and applications that one uses with a stand-alone database instance.
  + The computation and memory capacity of a DB Instance depends on the DB Instance class. For each DB Instance you can select from 5GB to 6TB of associated storage capacity.
  + The DB Instances are of the following types:
    - Standard Instances (m4, m3)
    - Memory Optimised (r3)
    - Micro Instances (t2)
* Regions & Availability Zones:
  + The AWS resources are housed in highly available data centers, which are in different areas of the world. This “area” is called a region.
  + Each region has multiple Availability Zones (AZ), they are distinct locations which are engineered to be isolated from the failure of other AZs.
  + You can deploy your DB Instance in multiple AZ, this ensures a failover i.e. in case one AZ goes down, there is a second to switch over to. The failover instance is called a standby, and the original instance is called the primary instance.
* Security Groups
  + A security group controls the access to a DB Instance. It does so by specifying a range of IP addresses or the EC2 instances that you want to give access.
  + Amazon RDS uses 3 types of Security Groups:
    - VPC Security Group - It controls the DB Instance that is inside a VPC.
    - EC2 Security Group - It controls access to an EC2 Instance and can be used with a DB Instance.
    - DB Security Group - It controls the DB Instance that is not in a VPC.
* DB Parameter groups
  + It contains the engine configuration values that can be applied to one or more DB Instances of the same instance type.
  + If you don’t apply a DB Parameter group to your instance, you are assigned a default Parameter group which has the default values.
* DB Option groups
  + Some DB engines offer tools that simplify managing your databases.
  + RDS makes these tools available with the use of Option groups.

**AWS RDS Components**

* The backups can be taken in two ways
  + The automated backups where in you set a time for your backup to be done.
  + DB Snapshots, where in you manually take a backup of your DB, you can take snapshots as frequently as you want.
* It automatically creates a secondary instance for a failover, therefore provides high availability.
* RDS AWS supports read replicas i.e. snapshots are created from a source DB and all the read traffic to the source database is distributed among the read replicas, this reduces the overall overhead on the source DB.
* RDS AWS can be integrated with IAM, for giving customized access to your users who will be working on that database.
* The updates to your database in RDS AWS are applied in a maintenance window. This maintenance window is defined during the creation of your DB Instance, the way it functions is like this:
  + When an update is available for your DB you get a notification in your RDS Console you can take one of the following actions
    - Defer the maintenance items.
    - Apply maintenance items immediately.
    - Schedule a time for those maintenance items.
  + Once maintenance starts, your instance has to be taken offline for updating it, if your instance is running in Multi-AZ, in that case the standby instance is updated first, it is then promoted to be a primary instance, and the primary instance is then taken offline for updating, this way your application does not experience a downtime.
  + If you want to scale your DB instance, the changes that make to your DB instance also happen during the maintenance window, you can also apply them immediately, but then your application will experience a downtime if its in a Single-AZ.

**AWS RDS Pricing**

* **Instance Class** i.e. the type of instance that you are choosing.
* **Running Time** i.e. the amount of time an instance is running, partial hours are billed as full hours.
* **Storage** i.e. the amount of storage that you have provisioned to your DB Instance
* **I/O Requests per Month** i.e. the I/O requests that are made to your DB Instance per month
* **Data Transfer**: Data transfer in and out of your DB Instance.
* **Reserved Instance** is also a way of using AWS RDS, in this you reserve an RDS Instance for a term, which can be for one or three years by making a onetime payment, it is a less expensive way compared to the monthly bill that one pays.
* **Free Tier**
  + AWS has an amazing free tier usage for most of its services, so that the customer can first use the service and then do the needful.
  + Similarly, it offers free tier usage for RDS AWS, which includes the following benefits:
    - 750 hours of Amazon RDS usage in single-AZ for db.t2.micro instance, every month for one year from signup.
    - 20 GB of DataBase Storage: any combination of General Purpose (SSD) or Magnetic storage.
    - 10 million IOs
    - 20GB of backup storage

**AWS Route 53**

* It is a scalable (DNS) service that provides a reliable way to redirect traffic to applications. To achieve this domain names are translated to IP addresses to help computers connect better.
* Route 53 is mainly used for 3 purpose:-
  + DNS Registration
  + Route Internet Traffic
  + Check Health Status

**AWS Route 53 Benefits**

* Highly Available and Reliable
  + It is built using AWS’s highly available and reliable infrastructure. DNS servers are distributed across many availability zones, which helps in routing end users to your website consistently.
  + Its Traffic Flow service helps improve reliability with easy re-route configuration when the system fails.
* Flexible
  + Its Traffic Flow provides users flexibility in choosing traffic policies based on multiple criteria, such as endpoint health, geographic location, and latency.
* Simple
  + Your DNS queries are answered by Route 53 in AWS within minutes of your setup, and it is a self-service sign up.
  + Also, you can use the simple AWS Route 53 API and embed it in your web application too.
* Fast
  + Distributed Route 53 DNS servers around the world make a low-latency service. Because they route users to the nearest DNS server available.
* Cost-effective
  + You only pay for what you use, for example, the hosted zones managing your domains, the number of queries which are answered per domain, etc.
  + Also, optional features like traffic policies and health checks are available at very low cost.
* Designed to Integrate with Other AWS Services
  + Amazon Route 53 works very well with other services like Amazon EC2 and Amazon S3.
  + For example, you can use Route 53 to map your domain names or IP addresses to your EC2 instances and Amazon S3 buckets.
* Secure
  + You can create and grant unique credentials and permissions to each user with your AWS account, while you must mention who have access to which parts of the service.
* Scalable
  + Amazon Route 53 is designed to automatically scale up or down when the query volume size varies.

**DNS routing (Route 53)**

The route 53 helps connect the browser to the user's website or application. When a user sends a request to a website (domain or subdomain), this request needs to be routed to the right address to get the relevant response. This is taken care of by route 53.

**Configuring route 53 to route requests:**

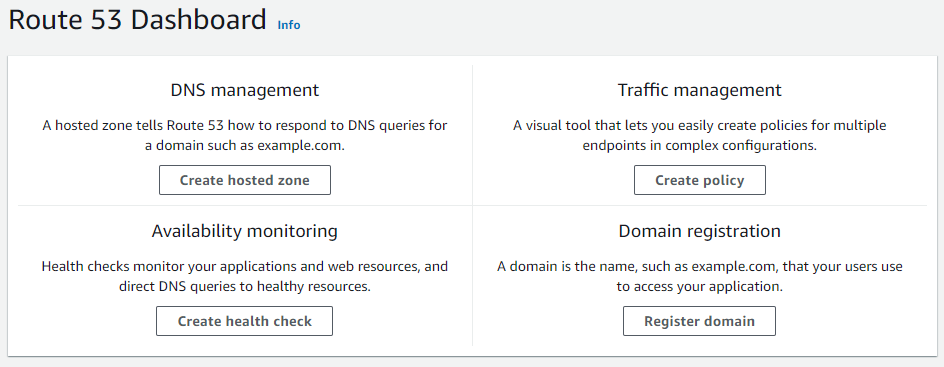
* The domain name is registered and once this is done, route 53 created a publicly hosted zone automatically. This zone has the same name as that of the domain name.
* The user creates ‘records’, which helps route traffic/requests from users to the user’s resources. These records are also known as ‘resource record sets’, which are present in the publicly hosted zone.
* Every record’s name present in the hosted zone should end with the hosted zone’s name, which is done automatically for the user by route 53.
* Every record has the below information, that helps route traffic to the specific domain:
  + Name: This corresponds to the domain or subdomain name; whose traffic needs to be routed with the help of route 53.
  + Type: It helps determine the type of resource which has to be used to route the traffic to. For example: Routing a traffic to a web server which has an IPV4 address, requires an A type, whereas routing traffic to an email server requires an MX Type.
  + Value: This is similar to Type, which specifies the name of the resource to which the traffic is routed to.

**Routing traffic to a user domain:**

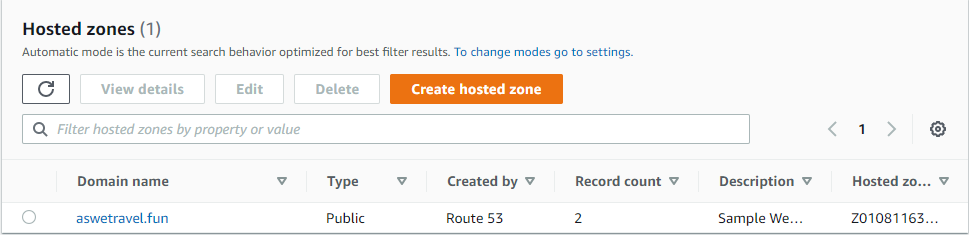
* When the user enters a domain name, and clicks on enter, this request is routed to a DNS resolver, which is managed by the user’s ISP (Internet Service Provider).
* This DNS resolver forwards the request by the user for the specific domain name to a DNS root name server.
* It also forwards the user request to TLD name services (a .com domain).
* The name server for the domain responds to this request by providing 4 route 53 name servers, which are associated with that specific domain.
* The DNS resolver stores these four routes 53 name servers and behaves like a cache, so that if the same user or any other user requests for the same domain, it can be easily fetched from the cache, instead of resolving the domain name.
* This cache stores information for only 2 days, post which data is refreshed to store more recent name servers.
* The DNS resolver chooses one of the 4 name servers and this is forwarded to that name server (to which the user originally sent a request).
* This name server looks for the record in the hosted zone of the domain name, and fetches the value associated with it (IP address).
* This address is returned to the DNS resolver, which will have the IP address required by the user.
* It returns this to the web browser, which sends a request for the name associated with that IP address.
* This is the location where the content is stored, that the user needs.
* The resource (usually a web server) returns the web page associated with that IP address to the web browser to be shown to the user.

**Hosting Domain on Route 53:**

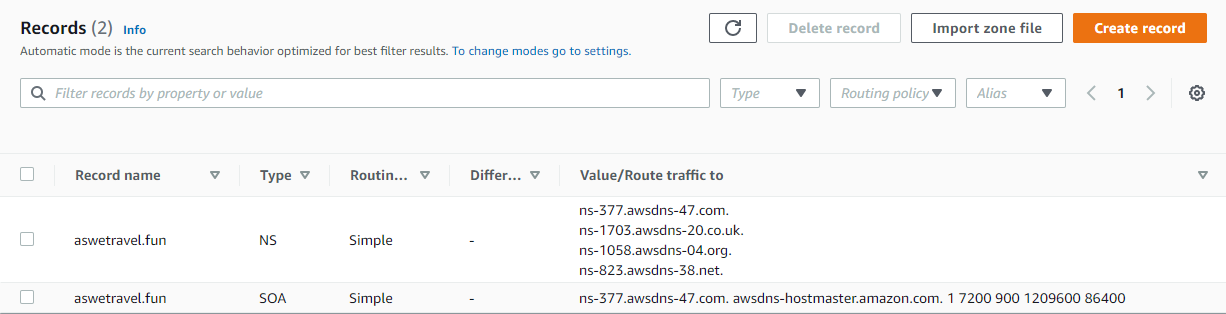
* Login to AWS Console 🡪 Go to Services 🡪 Drag down to Networking & Content Delivery and Select Route 53 🡪 Below Dashboard will open



* To create a hosted zone 🡪 Click on **Create hosted zone 🡪** Do the necessary hosted zone configuration.
  + Domain Name – **Eg. aswetravel.fun**
  + Description is optional.
  + Select the Type (**Public hosted zone** or **Private hosted zone**).
* Then click **Create hosted zone.** Below hosted zone is created



* NS & SOA records are created by default.



* After the zone is hosted, update the name server at the resolver end from where Domain name is bought.

**TTL (Time to Live)**

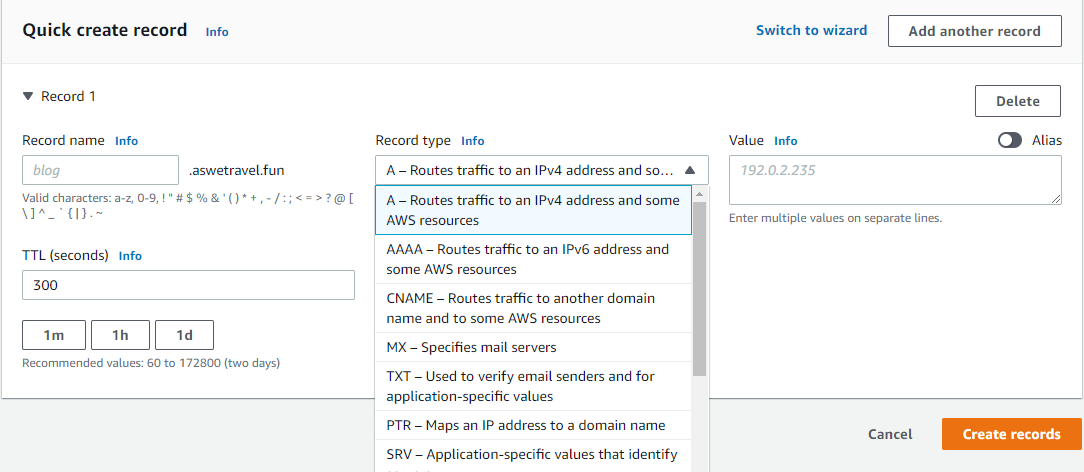
TTL is mandatory for each DNS record. So TTL is length that a DNS records is cached on either the resolving server or user own Laptop. The Lower the TTL, the faster changes to DNS records. Whenever you created record set, you need to define TTL for it.

**Working on records on Route 53:**

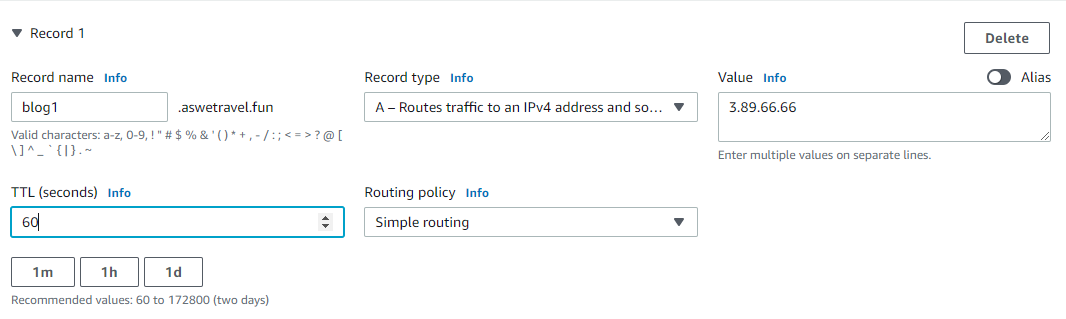
* **SOA (Start of Authority Records) -** Name of Server that supplied the data for zone. The administrator of that zone & current version of data file.
* **NS Record (Name Server Records)** - NS records is basically your name server records which are used by top level domain servers to direct traffic to content DNS server which contains the authoritative records.
* **A Record (URL to IPv4)** - The “A” record stands for Address record. The A record is used by computer to translate the name of the domain to an IP address.
* **CNAME (Canonical Records- URL to URL)** - CNAME Points a URL to any other URL. (gaurav.gupta.com => gkg.example.com), We use it only for Non-Root Domain (aka. something.mydomain.com)
* **Alias Record** - Alias record points a URL to an AWS Resource, Alias record are used to map resource record sets in your hosted zone to Elastic Load Balancer, CloudFront or S3 Buckets websites.
* **AAAA: (URL to IPv6)** - An AAAA record maps a domain name to the IP address (Version 6) of the computer hosting the domain. An AAAA record is used to find the IP address of a computer connected to the internet from a name.
* **MX Record (Mail Exchange Record)** - A mail Exchanger record (MX record) specifies the mail server responsible for accepting email messages on behalf of a domain name. It is a resource record in the Domain Name System (DNS). It is possible to configure several MX records, typically pointing to an array of mail servers for load balancing and redundancy.

**Creating records on Route 53:**

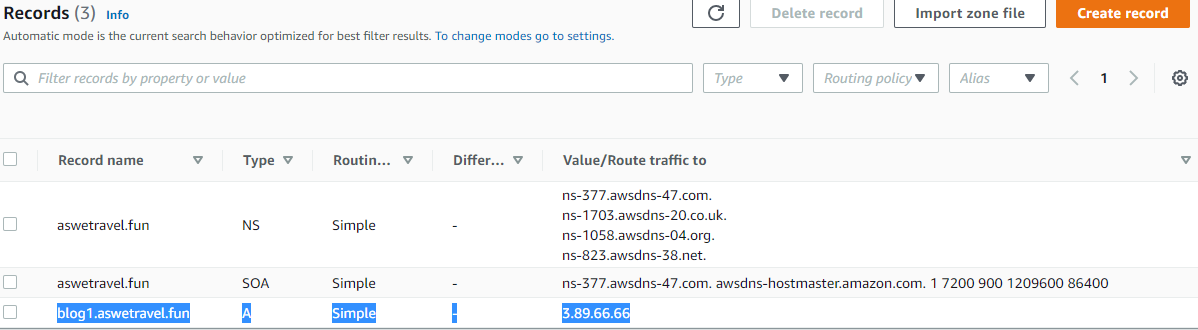
* Navigate to your hosted zone 🡪 Click on create record 🡪 Below Dashboard will open



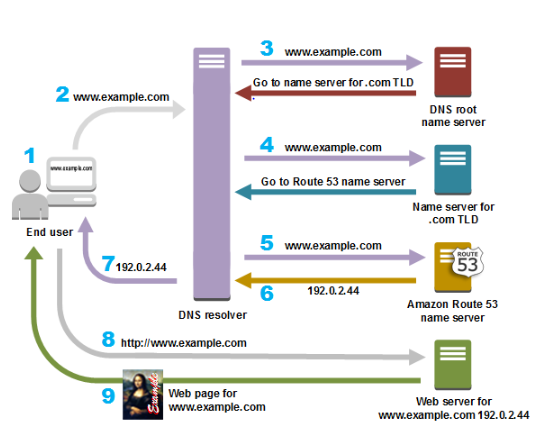
* Enter the Details like **Record Name, Record Type, Value** (IP Address in case of A records), **TTL, Routing Policy**. Below is the example of A record. After entering all details, click **Create records**.



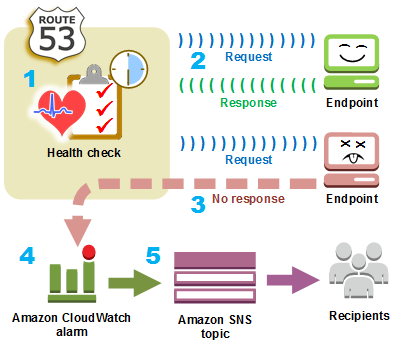
* Below Record is created. Similarly while creating certain records we can assign routing policies explained in next section.



**How AWS DNS (Route 53) Works?**



**How Amazon Route 53 Checks the Health of Your Resources**



**AWS Route 53 Routing Policies**

* **Simple Routing**
  + Simple routing policy is a simple round robin policy and can be applied when there is a single resource doing the function for the domain for e.g. web server that serves content for the website.
  + AWS Route 53 responds to the DNS queries based on the values in the resource record set for e.g. ip address in an A record.
* **Weighted routing policy**
  + Weighted routing policy enables Route 53 to route traffic to different resources in specified proportions (weights) for e.g., 75% one server and 25% to the other.
  + Weights can be assigned any number from 0 to 255.
  + Weighted routing policy can be applied when there are multiple resources that perform the same function for e.g., webservers serving the same site.
  + Weighted resource record sets let you associate multiple resources with a single DNS name.
  + To create a group of weighted resource record sets, two or more resource record sets can be created that have the same combination of DNS name and type, and each resource record set is assigned a unique identifier and a relative weight.
* **Latency routing policy**
  + Latency-based Routing Policy enables Route 53 to respond to the DNS query based on which data center gives the user the lowest network latency.
  + Latency-based routing policy can be used when there are multiple resources performing the same function and Route 53 needs to be configured to respond to the DNS queries with the resources that provide the fastest response with lowest latency.
  + Latency resource record set can be created for the EC2 resource in each region that hosts the application. When Route 53 receives a query for the corresponding domain, it selects the latency resource record set for the EC2 region that gives the user the lowest latency.
  + Route 53 then responds with the value associated with that resource record set for e.g., you might have web servers *for example.com in the EC2 data centers in Ireland and in Tokyo. When a user browse to example.com from Singapore, Route 53 will pick up the data center (Tokyo) which has the lowest latency from the user’s location.*
  + Latency based routing cannot guarantee users from the same geographic will be served from the same location for any compliance reason.
  + Latency resource record sets can be created using any record type that Route 53 supports except NS or SOA.
* **Failover routing policy**
  + Failover routing policy allows active-passive failover configuration, in which one resource takes all traffic when it’s available and the other resource takes all traffic when the first resource isn’t available.
  + Route 53 health checking agents will monitor each location/endpoint of the application to determine the availability.
  + Failover routing policy is applicable for Public hosted zones only.
* **Geolocation routing policy**
  + Geolocation routing policy enables Route 53 to respond to DNS queries based on the geographic location of the users i.e. location from which the DNS queries originate.
  + Geolocation routing policy allows geographic locations to be specified by continent, country, or by state (only in US).
  + Geolocation works by mapping IP addresses to locations, which might not map to an exact geographic location.
  + Two geolocation resource record sets that specify the same geographic location cannot be created.
* **Multivalue answer routing policy** — Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.
  + Multivalue answer Routing Policy is like Simple Routing Policy but it can return multiple IP addresses associated with an FQDN. Here there is more than one resource record for the same FQDN, pointing to different IP addresses, load balancer etc.
  + DNS queries return the result in random order, which means that when you query next time, the IP address corresponds to the FQDN is in different order or sequence. This concept was traditionally used for load-balancing.

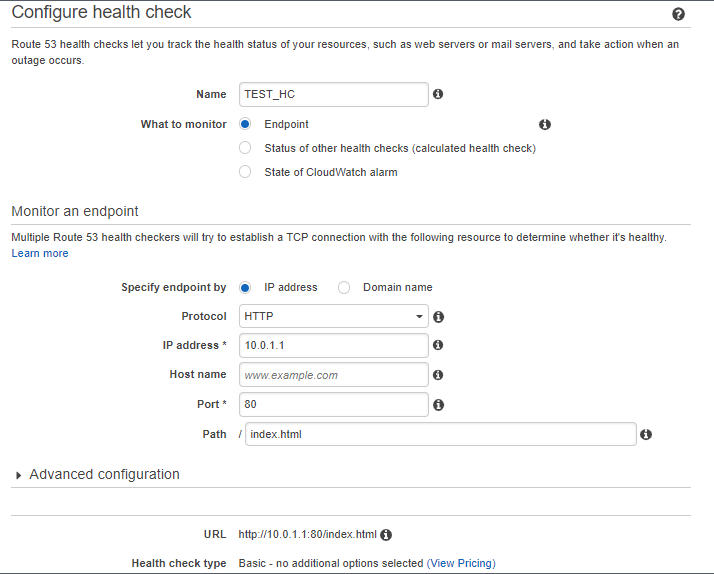
**Health Checking:**

Route 53 is also responsible in sending automated requests to a resource through the internet. This is done by route 53 to make sure that the web server which is serving user requests is available, functioning properly and reachable. There is a facility in route 53 which allows enabling the notifications which inform the user when a specific resource fails or becomes unavailable. This way, the user can route the requests to other available resources:

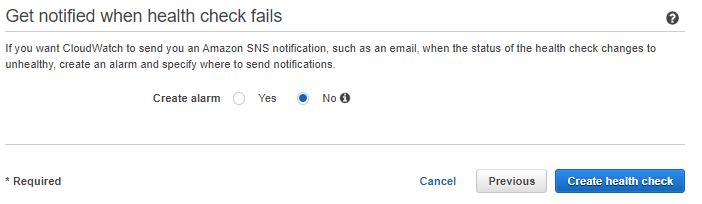
* It can be created by specifying values which allow different kinds of health checks to be done. Below are a few of the values which could be specified:
  + **IP address** or **domain name** of the endpoint (the web server which must be monitored by the Route 53).
  + **The request interval**: This tells about the frequency of route 53 sending a request to the endpoint.
  + **Protocol**: The set of rules which the user wants route 53 to use to perform the health check, whether it should be HTTP, HTTPS, or TCP.
  + **Failure threshold**: The number of times that the endpoint fails to respond to the request continuously before it can be deemed unhealthy or unavailable by route 53.
* When an endpoint is unavailable or unhealthy, it is detected by route 53. Notifications can be enabled to notify the same. This is done by setting a CloudWatch alarm (which is done by route 53 automatically). Amazon SNS is used by CloudWatch to notify users about an endpoint’s health.
* A health check interval is provided so that at this interval of time, route 53 sends request to the endpoint to perform a health check.
* When the endpoint responds to the request sent by route 53, it is considered as a healthy endpoint, and no further action is taken.
* If the endpoint doesn’t respond to a request, route 53 starts sending consecutive requests and keeps a count of the number of requests which are continuously sent to the endpoint:
  + A failure threshold is pre-defined and when the count reaches the failure threshold, route 53 considers such an endpoint to be unhealthy.
  + If the endpoint responds before the count reaches the failure threshold, the count is reset to 0 by route 53.
* If the endpoint is deemed unhealthy by route 53, this is notified to CloudWatch by route 53 (if it was configured to provide a notification when the endpoint is unhealthy).
* If the configuration to provide a notification wasn’t done, the status of the route 53 can be still be checked in the route 53’s console.
* If the notification is configured for a health check, CloudWatch triggers an alarm to indicate about an endpoint not being healthy, with the help of Amazon SNS, and sends it to the respective user.

**AWS Route 53 Creating Health Checks –**

* Open Route 53 Dashboard 🡪 From Left Panel Select Health Checks 🡪 Then Click Create Health Check 🡪 Below Dashboard will open.



* Enter the health Name 🡪 Select What to Monitor 🡪 Specify Endpoint by (IP Address or Domain Name) 🡪 Enter Protocol 🡪 Enter IP Address (of instance) 🡪 Enter Hostname (optional) 🡪Enter Port No
* Click Next 🡪 Below Dashboard will open 🡪 Select Create Alarm (optional) 🡪 Click Create Health Check.



* Health Check will be created as below, and status must be Healthy



**AWS VPC (Virtual Private Cloud)**

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network

that you've defined. This virtual network closely resembles a traditional network that you'd operate in

your own data center, with the benefits of using the scalable infrastructure of AWS.

**Amazon VPC Concept:**

* **Virtual Private Cloud -** A virtual network dedicated to your AWS account.
* **Subnet -** A range of IP addresses in your VPC.
* **Route table -** A set of rules, called routes, that are used to determine where network traffic is directed.
* **Internet gateway -** A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.
* **VPC endpoint -** Enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.
* **CIDR block -** Classless Inter-Domain Routing. An internet protocol address allocation and route aggregation methodology.

**VPCs & Subnet**

* A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC. You can specify an IP address range for the VPC, add subnets, associate security groups, and configure route tables.
* A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.
* To protect the AWS resources in each subnet, you can use multiple layers of security, including security groups and network access control lists (ACL).
* You can optionally associate an IPv6 CIDR block with your VPC and assign IPv6 addresses to the instances in your VPC.

**Default and nondefault VPCs**

* A default VPC has the benefits of the advanced features provided by EC2-VPC and is ready for you to use. If you have a default VPC and don't specify a subnet when you launch an instance, the instance is launched into your default VPC. You can launch instances into your default VPC without needing to know anything about Amazon VPC.
* You can also create your own VPC and configure it as you need. This is known as a nondefault VPC. Subnets that you create in your nondefault VPC and additional subnets that you create in your default VPC are called nondefault subnets.

**Route Tables**

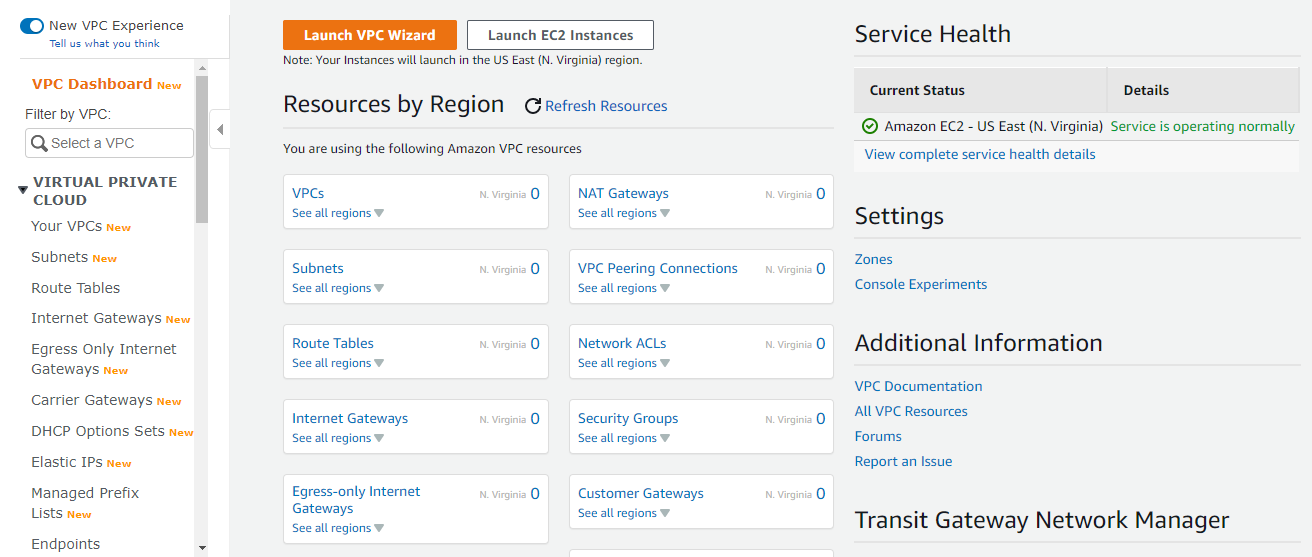
* A route table contains a set of rules, called routes, that are used to determine where network traffic from your VPC is directed. You can explicitly associate a subnet with a particular route table. Otherwise, the subnet is implicitly associated with the main route table.
* Each route in a route table specifies the range of IP addresses where you want the traffic to go (the destination) and the gateway, network interface, or connection through which to send the traffic (the target).

**Accessing the Internet**

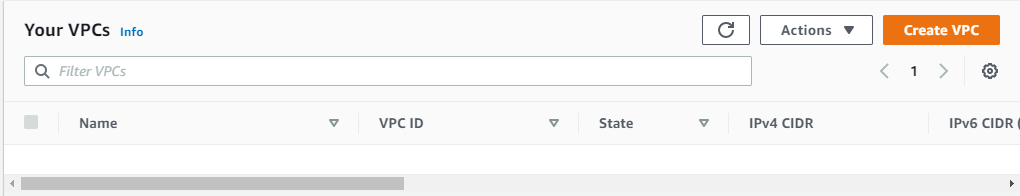
* You control how the instances that you launch into a VPC access resources outside the VPC.
* Your default VPC includes an internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IPv4 address and a public IPv4 address. These instances can communicate with the internet through the internet gateway. An internet gateway enables your instances to connect to the internet through the Amazon EC2 network edge.

**Working on VPC Console:**

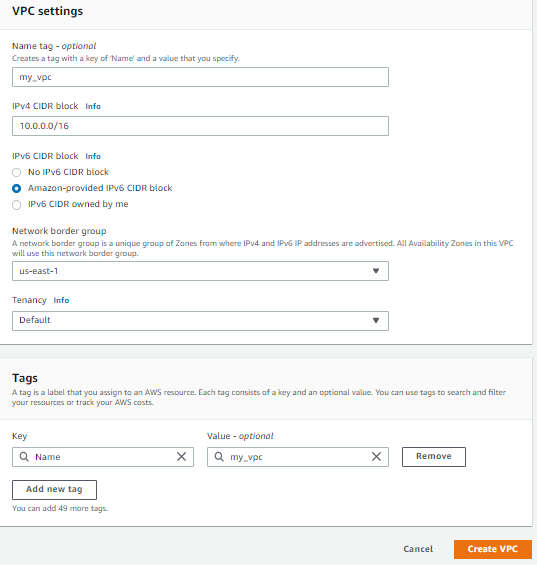
* **Login to AWS Console 🡪 Got to Services and Drag down to Networking & Content Delivery & select VPC.** Below Dashboard will open.



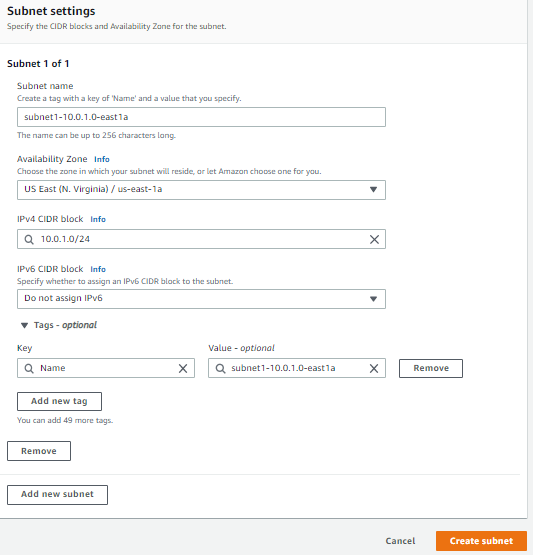
* **For new VPC Creation: Click on Your VPC from Left Panel 🡪 Below Console will open**



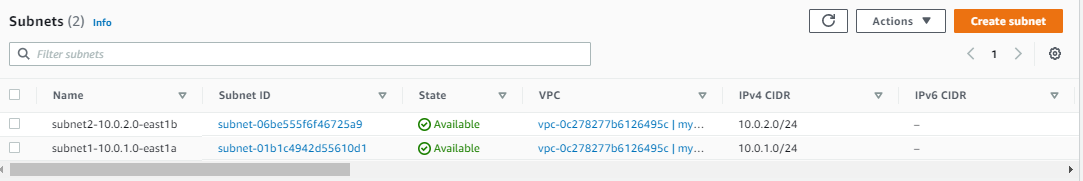
* **Click on Create VPC 🡪 Enter Name of the VPC (my\_vpc) 🡪 Enter the CIDR Block (IP Segment) 🡪 Click Create VPC**



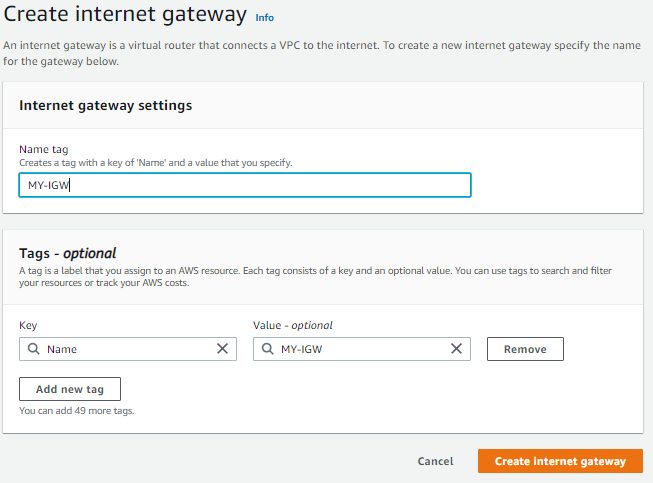
* **After VPC is created, we need to create Subnet: Go to Subnets from Left Panel 🡪 Enter Subnet Name 🡪 Select Availability zone as per your choice 🡪 Create new IPv4 CIDR Block (10.0.1.0/24) 🡪 IPv6 is optional 🡪 Click Create Subnet.**



* **This case we have create 2 subnets as below. To enable Auto Assign Public IP to any created subnet 🡪 Select the subnet 🡪 Go to Actions 🡪 Click Modify Auto Assign IP Settings.**



* **Now we will create a new Internet Gateway** 🡪 **Select Internet Gateway from Left panel 🡪 Then select create Internet Gateway 🡪 Enter the Internet gateway name 🡪 then click create internet gateway**



* After Internet Gateway is created 🡪 Go to actions and click on Attach a VPC 🡪 Then Click Attach Internet gateway

