AWS STUDY GUIDE

INTRODUCTION TO CLOUD COMPUTING

Cloud computing is the on demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. The term is generally used to describe data centres available to many users over the Internet.

ADVANTAGES OF CLOUD COMPUTING

- ➔ Cost saving
- ➔ Security
- ➔ Flexibility
- ➔ Mobility
- ➔ Disaster Recovery
- ➔ Automatic software updates
- ➔ Large number of services available for use.

NETWORKING CONCEPTS FOR CLOUD COMPUTING

LINUX NAMESPACES

A namespace is a way of scoping a particular set of identifiers. Using a namespace, you can use the same identifier multiple times in different namespaces. You can also restrict an identifier set visible to particular processes.

At a high level, Linux namespace allow for isolation of global system resources between independent processes. For example, the PID namespace isolates the process ID number space. This means that two processes running on the same host can have the same PID.

NETWORK NAMESPACES

In a network namespace, the scoped 'identifiers' are network devices; so a given network device, such as **eth0**, exists in a particular namespace. Linux starts up with a default network namespace, so if your operating system does not do anything special, that is where all the network devices will be located. However, it is also possible to create further non-default namespaces, and create new devices in those namespaces, or to move an existing device from one namespace to another.

OVERLAY NETWORKS

Overlay network is a network spread over another network. For example, any application that has or provides services and is in a client server architecture is an overlay network over the internet.

STORAGE CONCEPTS FOR CLOUD COMPUTING

STORAGE VIRTUALIZATION

In computer science, **storage virtualization** is "the process of presenting a logical view of the physical storage resources to a host computer system, "treating all storage media (hard disk, optical disk, tape, etc.) in the enterprise as a single pool of storage.

TYPES OF STORAGE

- ➔ SAN : Storage area Network (Fibre optics)
- ➔ DAS : Direct attached storage (Physical hard drive)
- ➔ NAS : Network attached storage (NFS)

CLOUD COMPUTING DEPLOYMENT MODELS

PUBLIC CLOUD: The public cloud is defined as computing services offered by third-party providers over the public Internet, making them available to anyone who wants to use or purchase them. They may be free or sold on-demand, allowing customers to pay only per usage for the CPU cycles, storage, or bandwidth they consume.

PRIVATE CLOUD: A private cloud is a particular model of cloud computing that involves a distinct and secure cloud based environment in which only the specified client can operate.

HYBRID CLOUD: Is a cloud-computing environment that uses a mix of on-premises, private cloud and third party, public cloud services with orchestration between the two platforms

COMMUNITY CLOUD: A community cloud in computing is a collaborative effort in which infrastructure is shared between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third party and hosted internally or externally. This is controlled and used by a group of organizations that have shared interest. The costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized.

CLOUD COMPUTING SERVICE DELIVERY MODEL

IAAS: Infrastructure as a service : IaaS is the lowest-level cloud service paradigm and arguably the most important. With IaaS, pre-configured hardware resources are provided to users through a virtual interface. Unlike PaaS and SaaS, IaaS doesn't include applications or even an operating system (implementing all of that is left up to the customer), it simply enables access to the infrastructure needed to power or support that software. IaaS can provide extra storage for corporate data backups, network bandwidth for a company website server, or it can even enable access to high power computing which was previously only accessible to those with supercomputers. Popular IaaS offerings like Amazon EC2

PAAS: a cloud service model where the cloud is used to deliver a platform to users from which they can develop, initialize and manage applications. PaaS offerings typically include a base operating system and a suite of applications and development tools. PaaS eliminates the need for organizations to build and maintain the infrastructure traditionally used to develop applications

SAAS: Sometimes referred to as 'on-demand software', SaaS is a software licensing and delivery model where a fully functional and complete software product is delivered to users over the web on a subscription basis. SaaS offerings are typically accessed by end users through a web browser (making the user's operating system largely irrelevant) and can be billed based on consumption or, more simply, with a flat monthly charge. Example ServiceNow.

INTRODUCTION TO AWS

- ➔ Stands for Amazon web services
- ➔ Fastest growing cloud computing platform.
- ➔ Was officially launched in 2006

AWS GLOBAL INFRASTRUCTURE

AWS infra consists of 3 components

1) **Regions**: A region is a geographical area on the earth.
2) **Availability zones**: Each region have multiple data centres and we call them as Availability zone.'
3) **Edge locations**: Are cloud delivery network endpoints for cloud front. These are used for fast delivery of huge amount of data in network.

SIGNING UP FOR A FREE TIER ACCOUNT

- ➔ Go to the link https://aws.amazon.com/free/
- ➔ Click on "Create a free account" and follow the steps.

Note : We will be asked for giving the credit card details and we need to provide it.
INR 2 will be deducted just for verification purpose and will be returned in 2 days max.

IDENTITY AND ACCESS MANAGEMENT (IAM)

IAM allows you to manage users and their level of access to the AWS console.
IAM is a GLOBAL service.

WHAT DOES IAM GIVES YOU?

- ➔ Centralised control of your AWS account.
- ➔ Shared access to your AWS account.
- ➔ Granular permissions
- ➔ Identity Federation (Including AD, Facebook etc)
- ➔ Multifactor authentication
- ➔ Set up our own password policy
- ➔ Consists of three major components

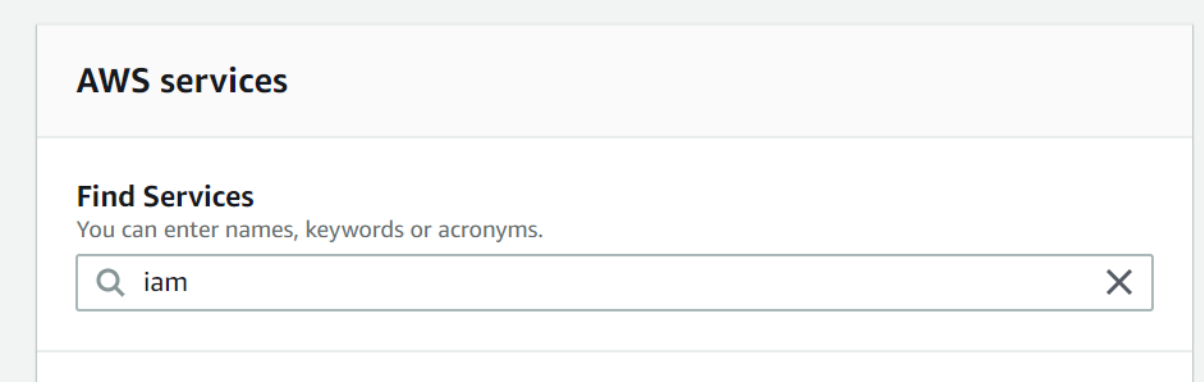
1) **Users**: End users that will use the AWS services.
2) **Groups**: Collection of users with similar interest.
3) **Roles**: We create roles (according to the desired permissions) and then attach to the AWS resources. By default maximum 500 roles can be created.
4) **Policies**: Document that define one or more permissions. These can be attached to user, group or a role.

- ➔ There are two type of access that can be given to a user
    1) Console access: With which a user can work via the console. A username and password is given to each user.
    2) Programmatic access: Which is helpful in allowing to work with AWS cli, each user when created is given an “Access Key ID” and “secret access key” that should be kept secret.
- ➔ The root account is the email id with which we logged in and created the AWS account.
- ➔ If the credentials are lost, they cannot be recovered and needs to be recreated.

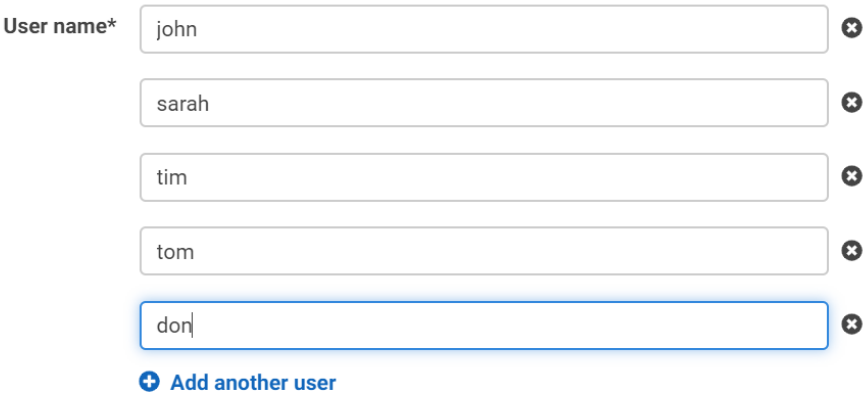
➔ Each user can have maximum of 2 key pairs

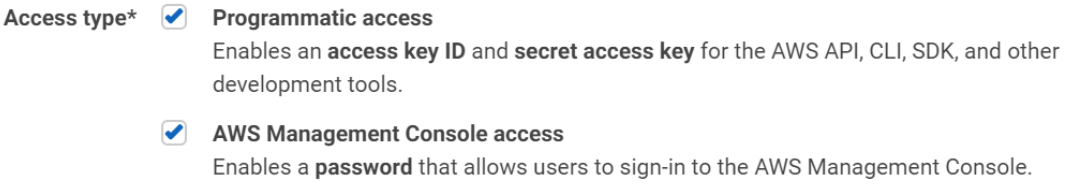
NOTE: There is a special type of access called the “Power user access” which has all the rights, same as admin but cannot manage users or groups, also don’t have the access to the billing console.

LAB: Create 5 IAM users John, Sarah,Tim,Tom,Don. Create two new groups “Developers” and “Admins”. Make John, sarah, tim members of “Developers” group and Tom, don , members of Admin group.

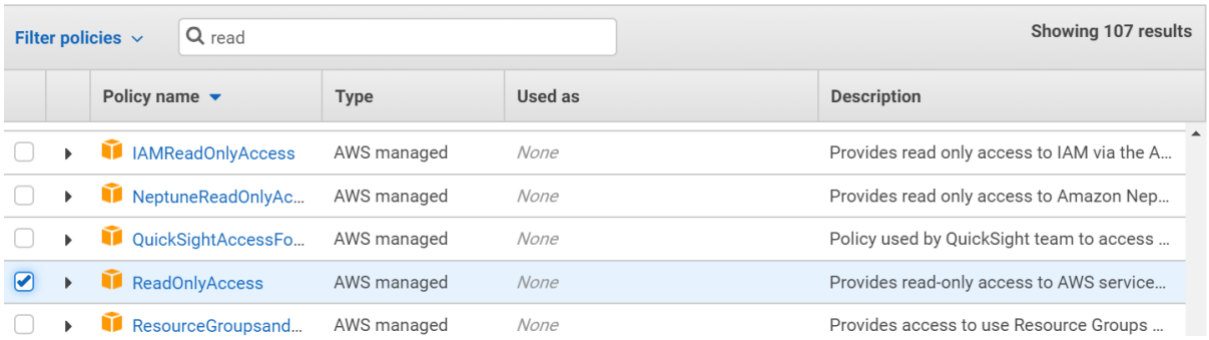


➔ Open the IAM service and click on users
➔ Click on “Add user”





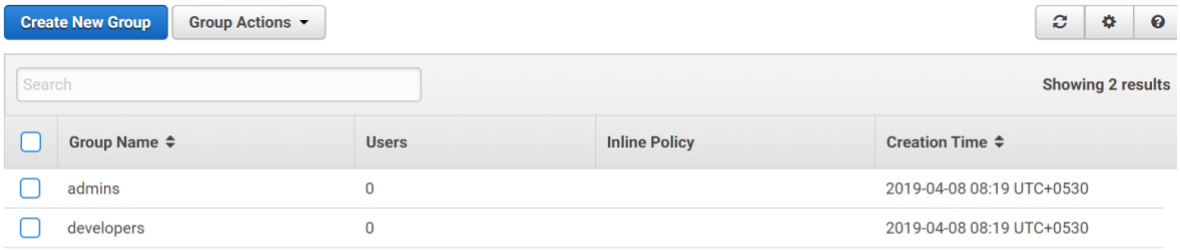
➔ Go to “Permissions” → Attach existing policies

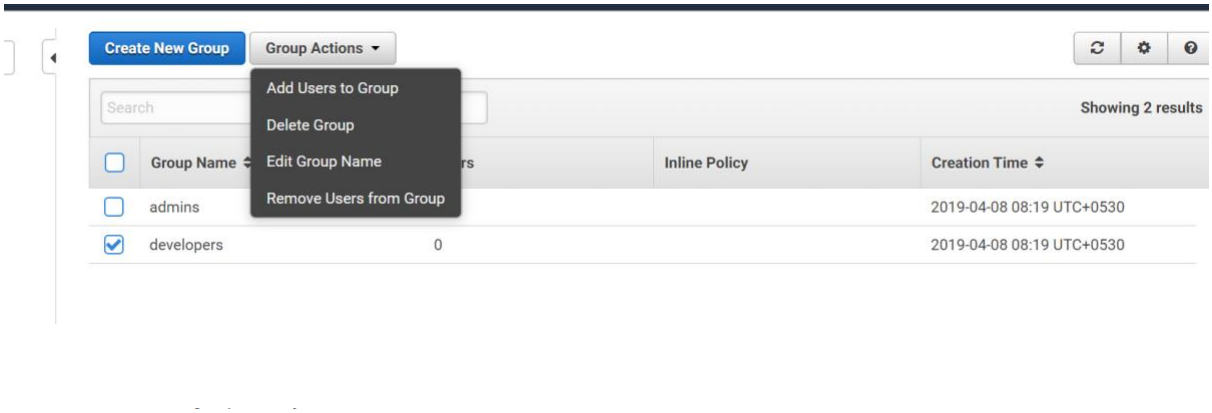


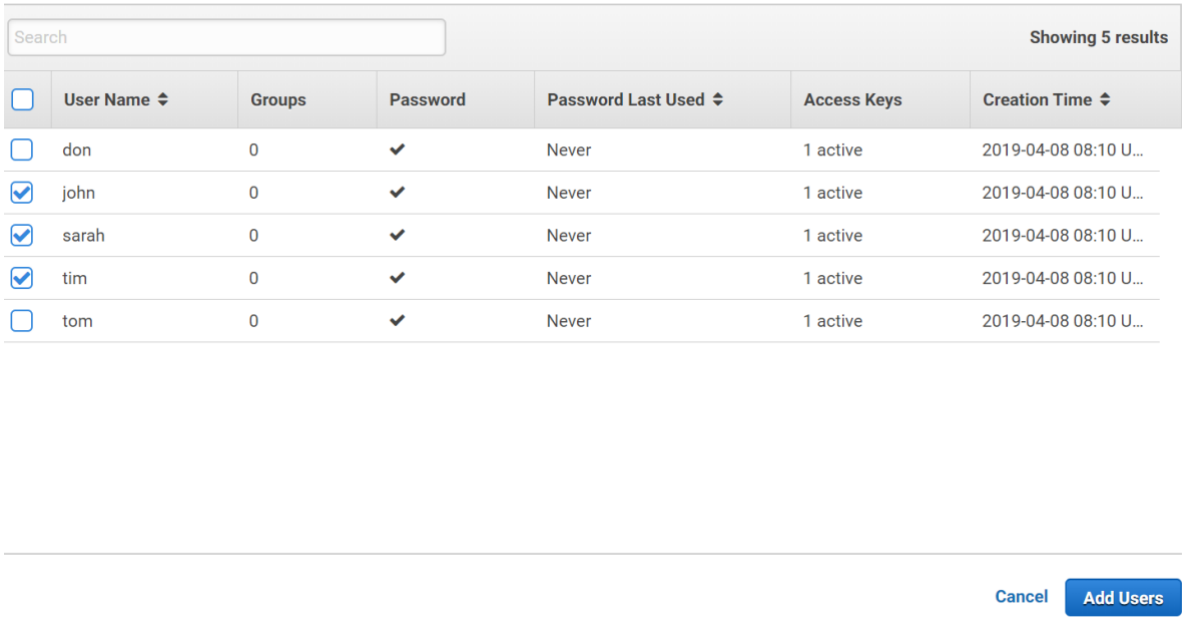
➔ Give tags → Click “Create Users”
➔ Download the keys for all users

➔ Create 2 new groups: Developers and Admins

■ Go to IAM -> Groups-> Create group-



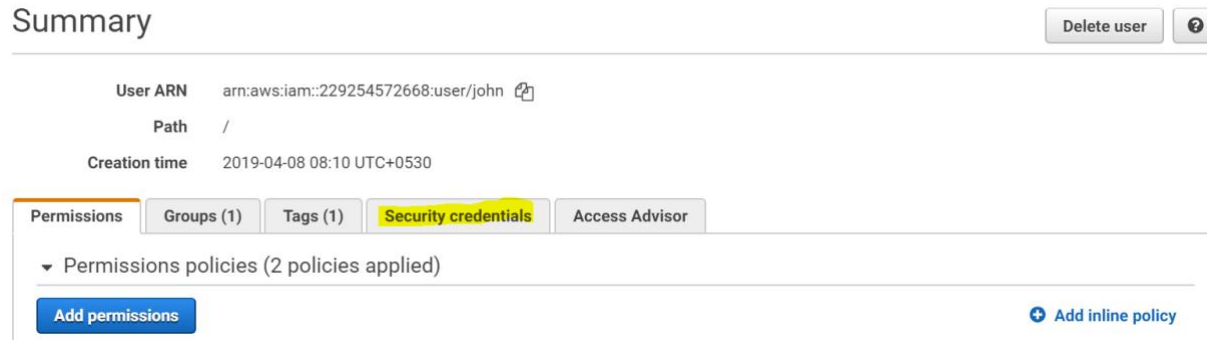


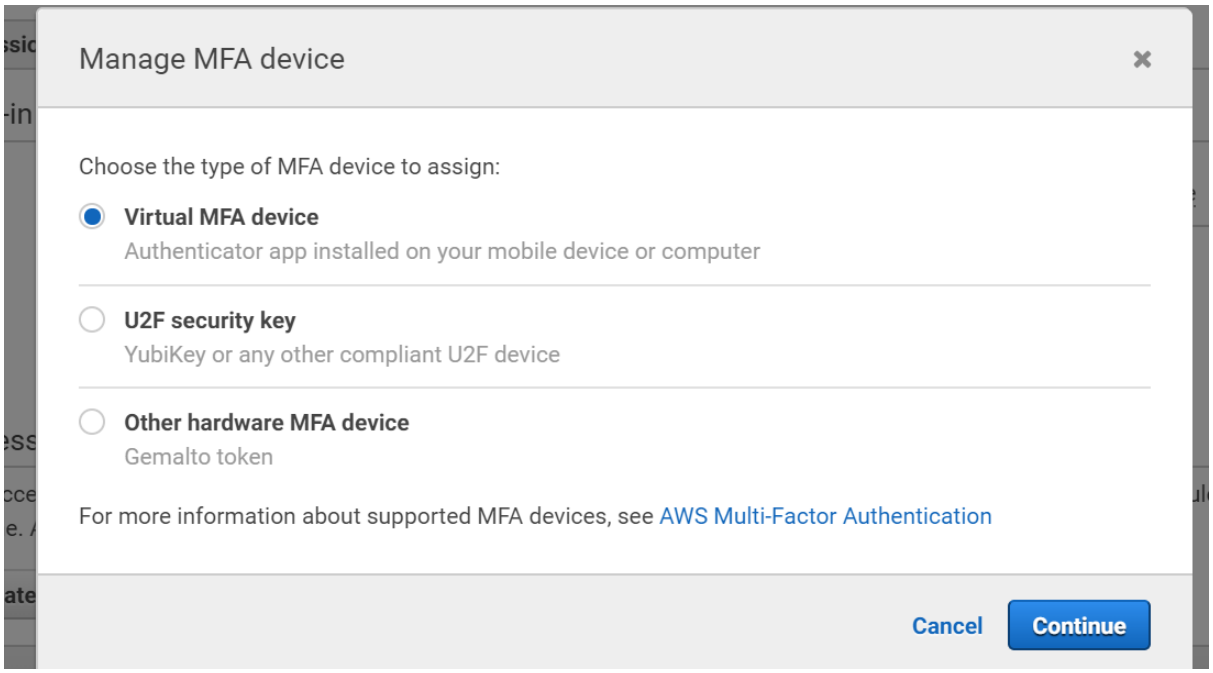


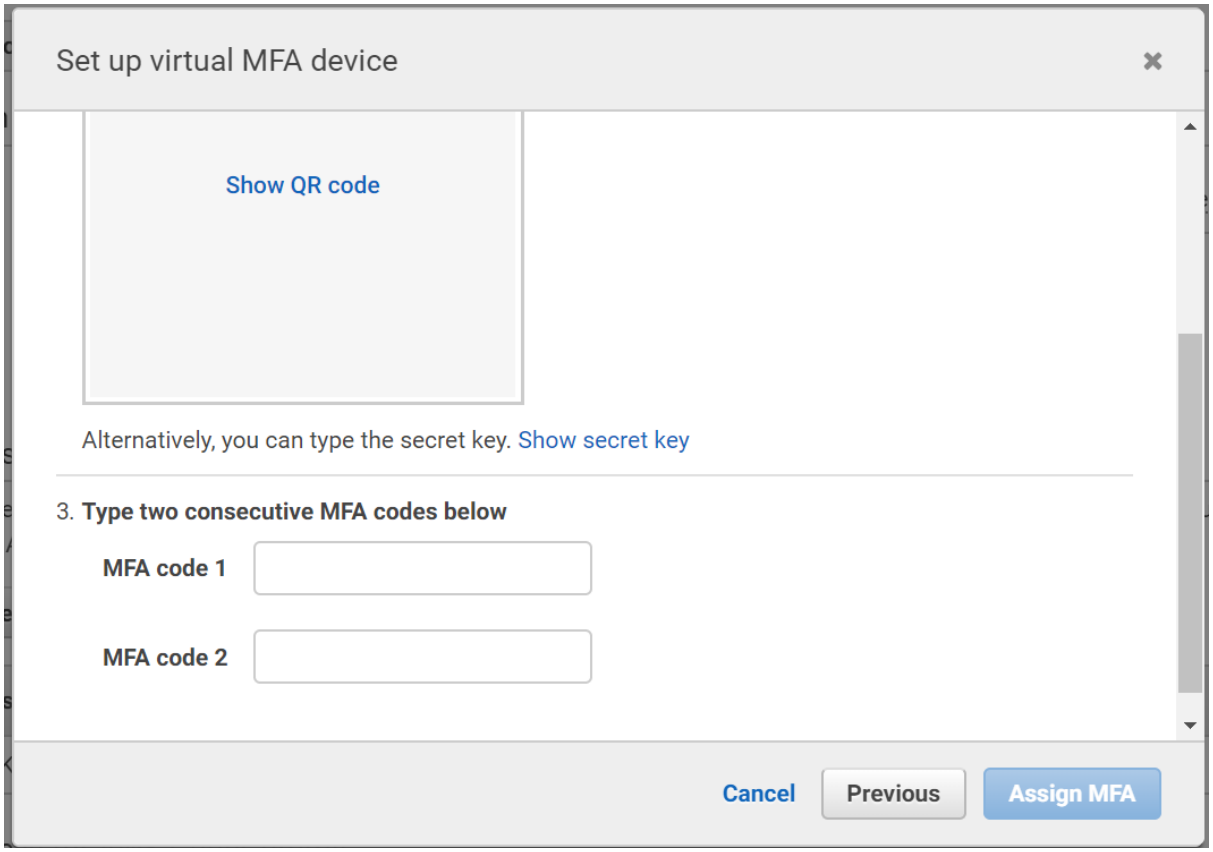
➔ Similarly proceed for Admins group.

■ Enable MFA for john

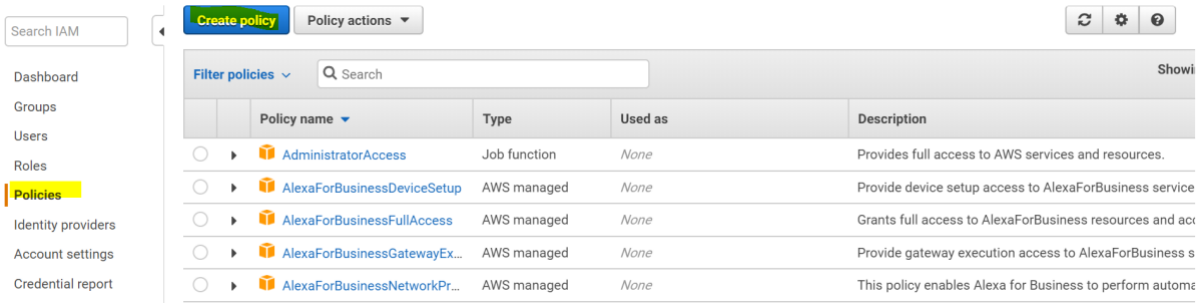
Download “Google Authenticator” from play store.


Summary
Delete user
User ARN
arn:aws:iam::229254572668:user/john
Path
/
Creation time
2019-04-08 08:10 UTC+0530
Permissions
Groups (1)
Tags (1)
Security credentials
Access Advisor
Permissions policies (2 policies applied)
Add permissions
Add inline policy

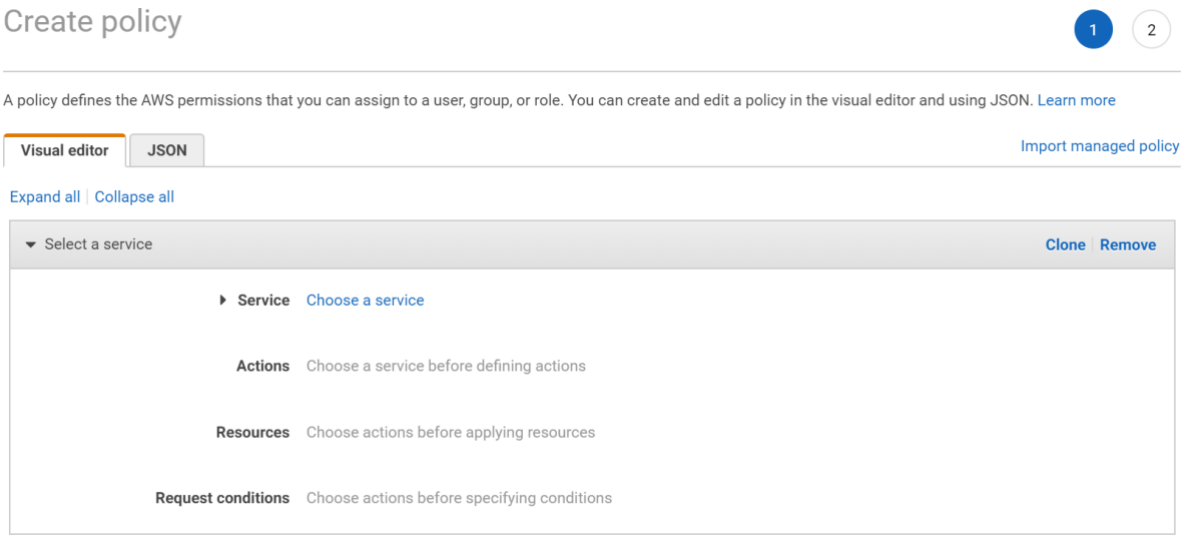

Manage MFA device
Choose the type of MFA device to assign:
Virtual MFA device
Authenticator app installed on your mobile device or computer
U2F security key
YubiKey or any other compliant U2F device
Other hardware MFA device
Gemalto token
For more information about supported MFA devices, see AWS Multi-Factor Authentication
Cancel
Continue



➔ Create your own custom IAM policy.



➔ Define policy rules



⇨ **IAM – POINTS TO REMEMBER**

- By default, 1000 IAM roles can be there per account.
- IAM role cannot be attached to an IAM group
- IAM role can be attached to multiple instances
- One role/instance is possible
- A role can be attached to a running instance.
- **IDENTITY FEDERATION**: AWS Identity and Access Management (IAM) supports identity federation for delegated access to the AWS Management Console or AWS APIs. With identity federation, external identities are granted secure access to resources in your AWS account without having to create IAM users. These external identities can come from your corporate identity provider (such as Microsoft Active Directory or from the AWS Directory Service) or from a web identity provider (such as Amazon Cognito, Login with Amazon, Facebook, Google, or any OpenID Connect-compatible provider).
- **CONSOLIDATED BILLING:** IAM and Consolidated Billing are complementary features. Consolidated Billing enables you to consolidate payment for multiple AWS accounts within your company by designating a single paying account.

S3 (SIMPLE STORAGE SERVICE)

➔ S3 stand for simple storage service.

➔ Provides developers with secure durable highly scalable object-oriented storage and is a safe place to store the files.

➔ it is object-based storage that is we can store PDF docs, videos files in S3 and cannot install OS or a database. For installation purpose we have block-based storage which is known as the EBS the elastic block storage.

➔ Data is spread across multiple devices and facilities so it is designed to withstand multiple failures.

➔ Maximum file chunk size can be 0 bytes to 5 GB in a single PUT or GET operation

- ➔ In case of S3 objects are stored in buckets which when we are working with Windows, we call them as folders.
- ➔ S3 is a Universal namespace that is names must be unique globally.
- ➔ A bucket is named as a DNS address of the format given below.
- ➔ A successful upload will get the http status as 200 which means the upload is successful

➔ **Data consistency model for S3**

- ➔ The first one is read after write consistency for puts of new objects.
- ➔ The second one is eventual consistency for overwrite puts and delete, it means that when we will put an object to S3 will be able to read them as soon as we uploaded, however we don't get that much speed when we are updating or deleting an object.
- ➔ Update in S3 are atomic that is be either are going to get the new data on the old data and we are not going to get the incomplete or corrupt data.
- ➔ Naming S3 buckets https://s3-ap-northeast-1.amazonaws.com/bucket_name/

<u>S3 STORAGE CLASSES</u>

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	millseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

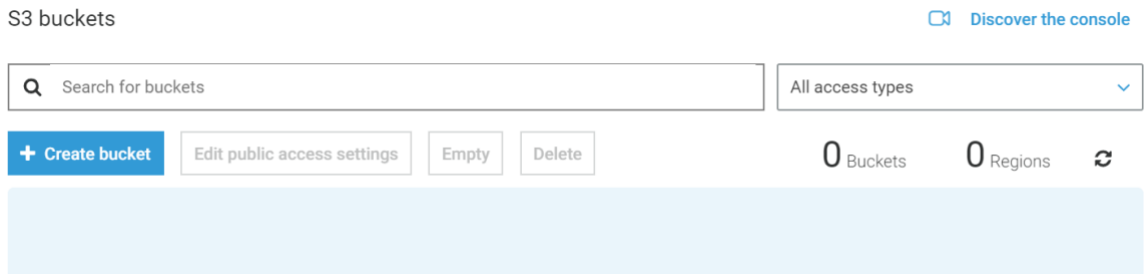
† Because S3 One Zone-IA stores data in a single AWS Availability Zone, data stored in this storage class will be lost in the event of Availability Zone destruction.

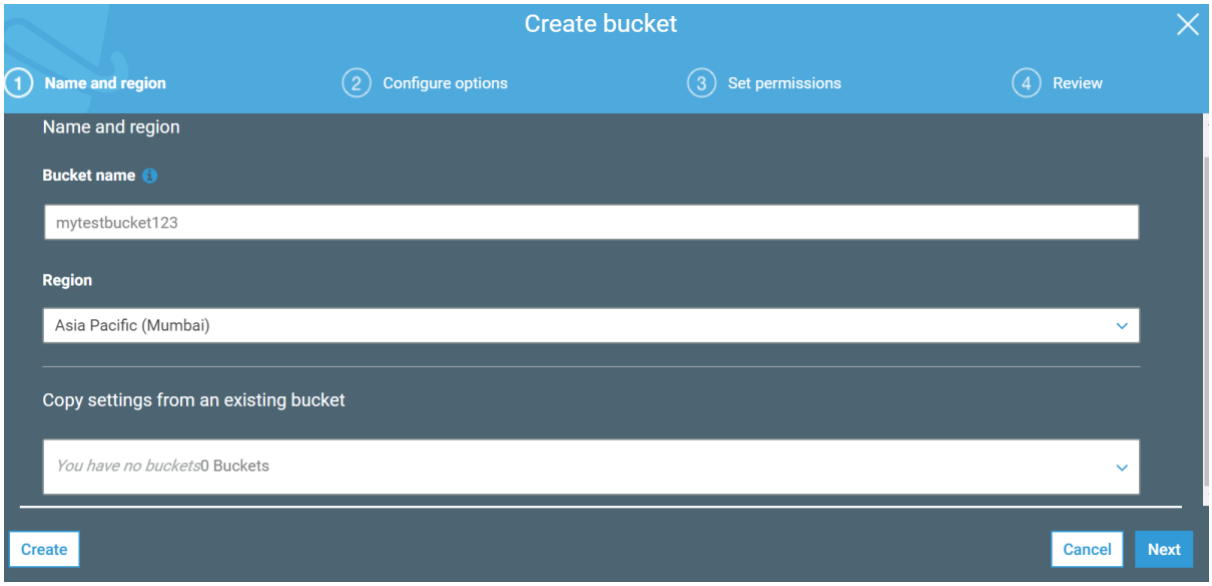
AMAZON GLACIER

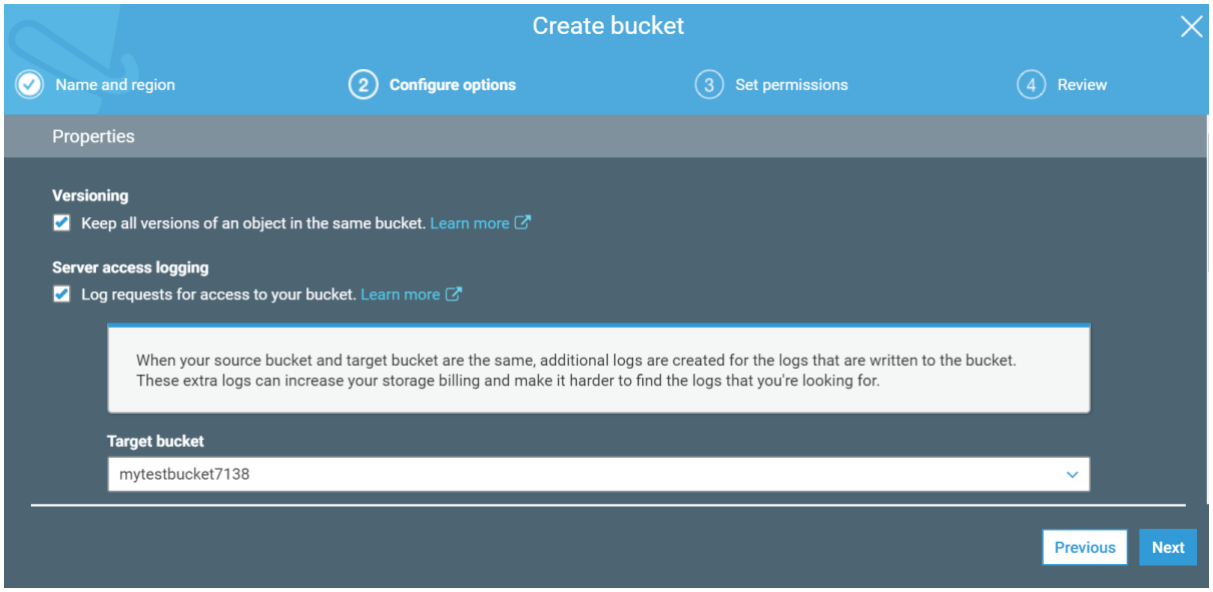
Glacier is an extremely low-cost storage service used for data archival and optimised for data that is infrequently accessed and takes 3 to 5 hours to retrieve data.

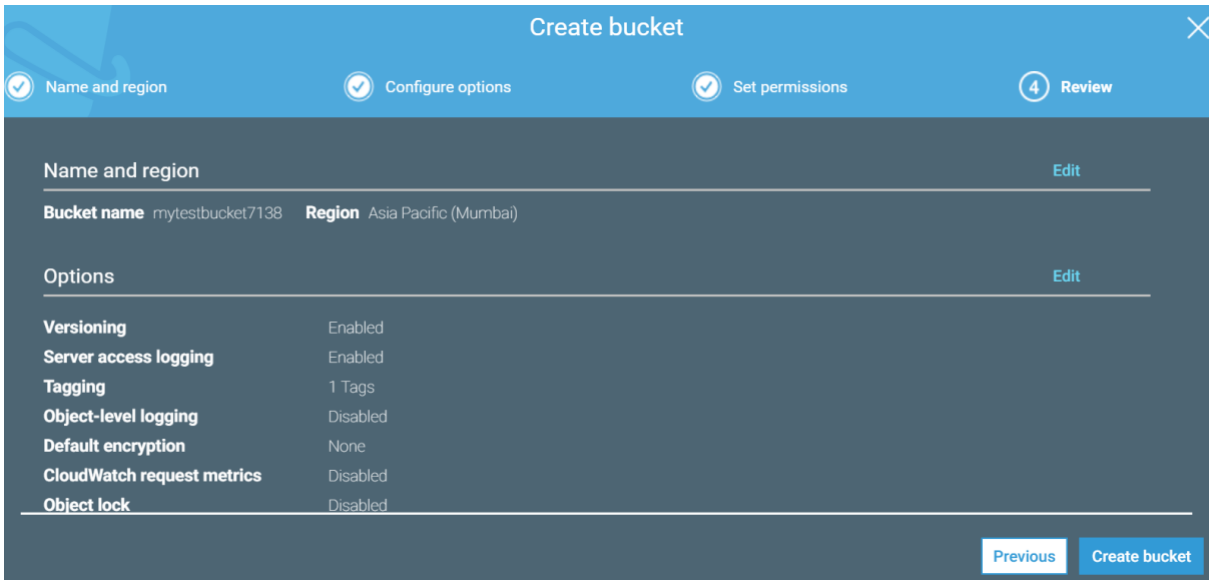
LAB 1: Create an S3 bucket, give necessary permissions and upload objects.

Services → S3 →









➔ Enable versioning on this bucket and no public access at all.

To modify the access permissions

- Select the bucket ->Go to permissions -> Modify the ACL or access policy
- Policy can be at Bucket level or at the object level.
- To sort the objects, we can add prefixes as well.
- We can enable S3 access logging as well.
- Always good to enable versioning for the files so that we have a backup.
- Once versioning is enabled, it can't be removed, only can be disabled.
- By default everything in a bucket is private, if in case a bucket is public, doesn't mean that objects are public as well.
- If multiple versions of a file exist, in the view it will appear as only one file but the storage space consumed will be the sum of sizes of all files.

- If a version is deleted, it cannot be recovered and it will be deleted forever
- If an object is deleted, we have an option to undo the delete and it can be recovered.
- It's a good thing to enable MFA on deletes.

  When versioning is enabled, a simple DELETE cannot permanently delete an object. Instead, Amazon S3 inserts a delete marker in the bucket, and that marker becomes the current version of the object with a new ID. When you try to GET an object whose current version is a delete marker, Amazon S3 behaves as though the object has been deleted (even though it has not been erased) and returns a 404 error.

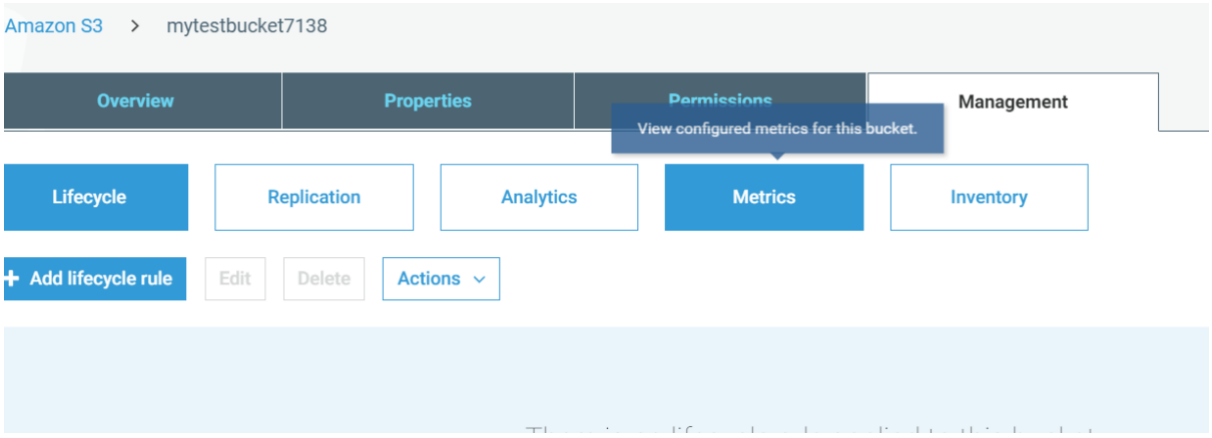
- Actually, when a version is deleted, a delete marker is set on that and if we want to restore the file, we need to just delete the delete marker.

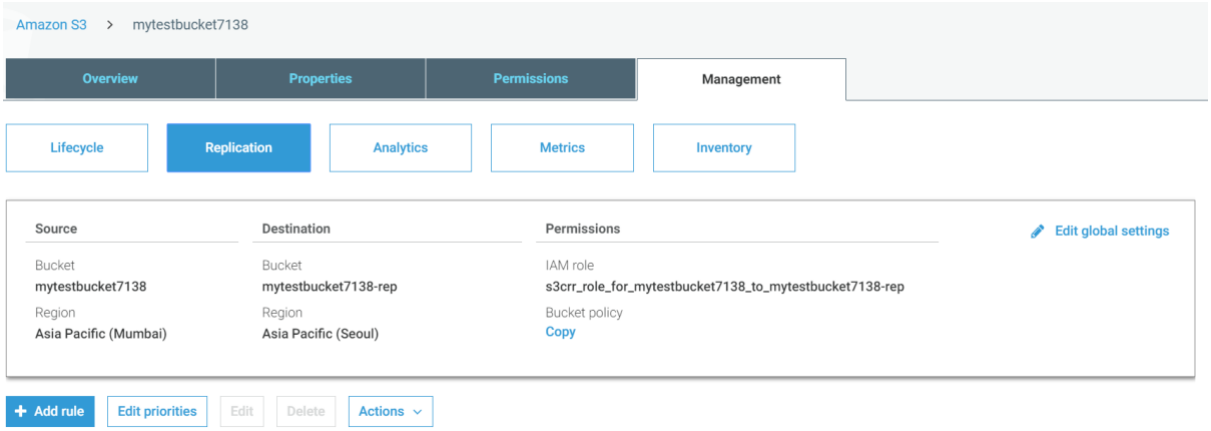
CROSS REGION REPLICATION OF S3 BUCKETS

- With this we can replicate a bucket in one region to another bucket in another region.
- Versioning should be enabled on both source and destination buckets.
- Already existing objects will not be replicated and only the new ones will be.
- All versions and object permissions are replicated.
- Deleted versions are not replicated.
- Delete markers are replicated.
- Deletion of delete markers is not replicated.

  LAB:

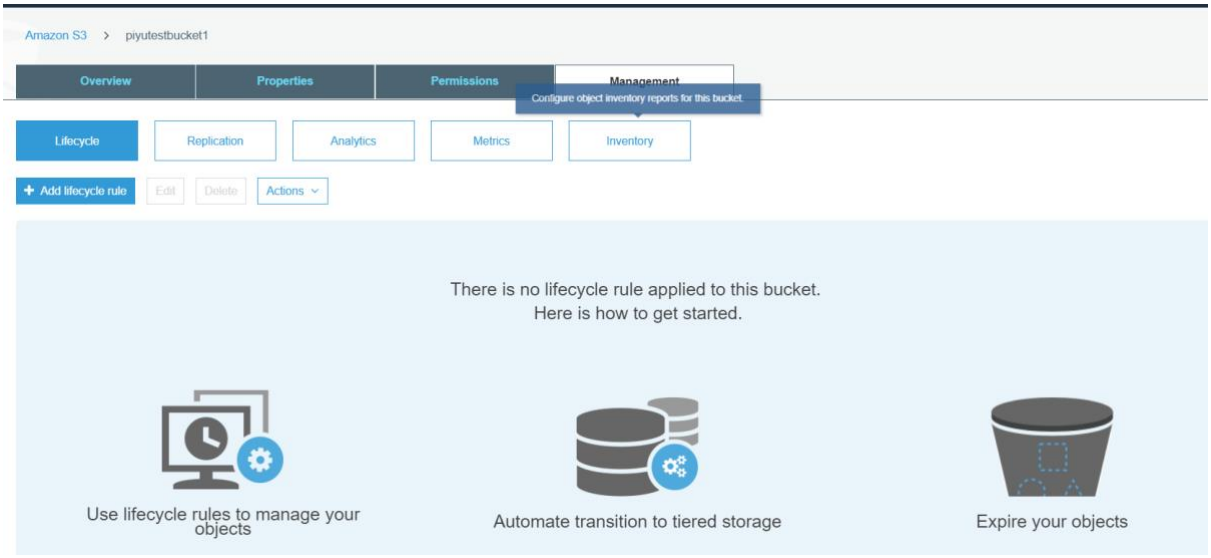
➔ Select the bucket → Management→ Replication→ Configure the settings.

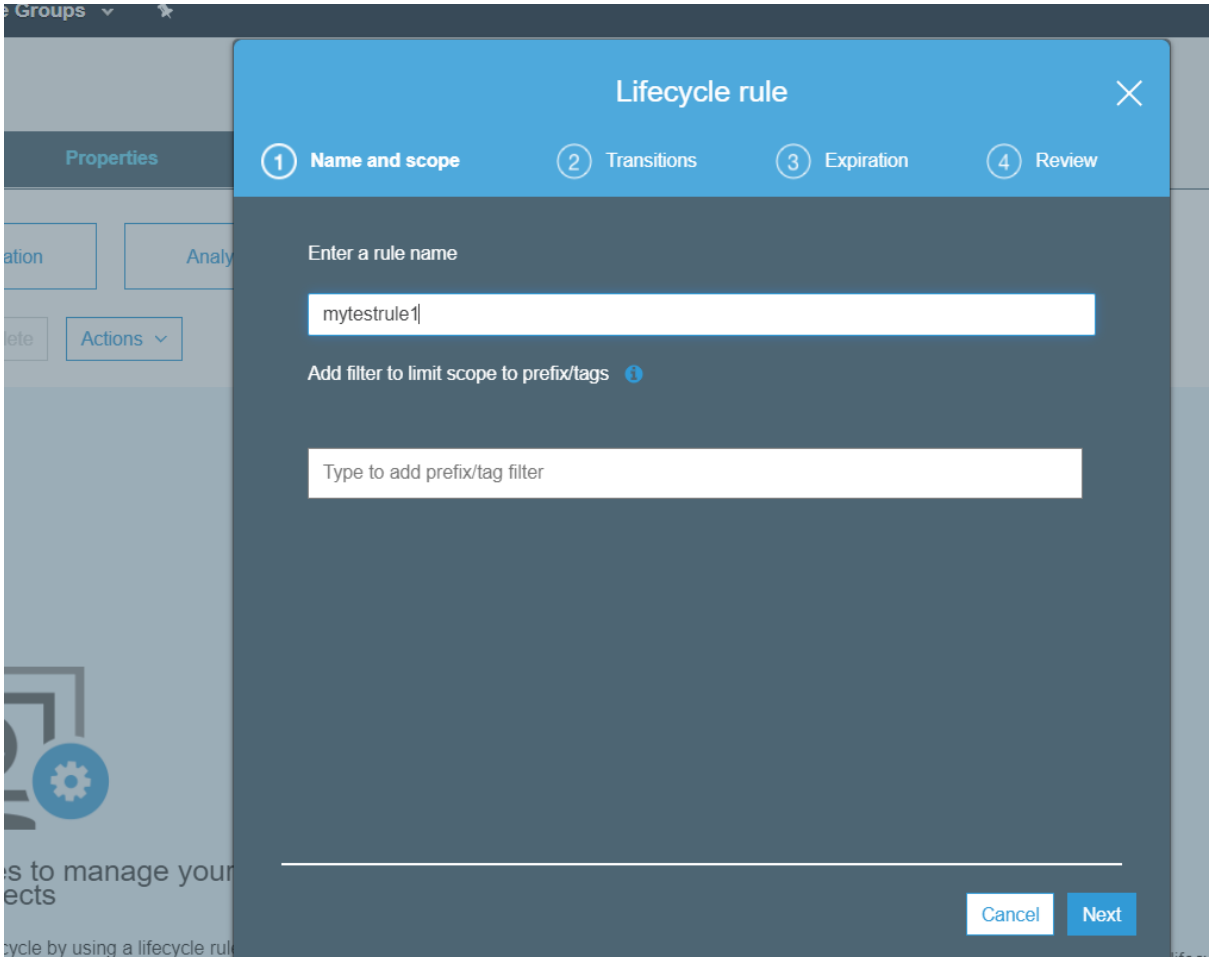


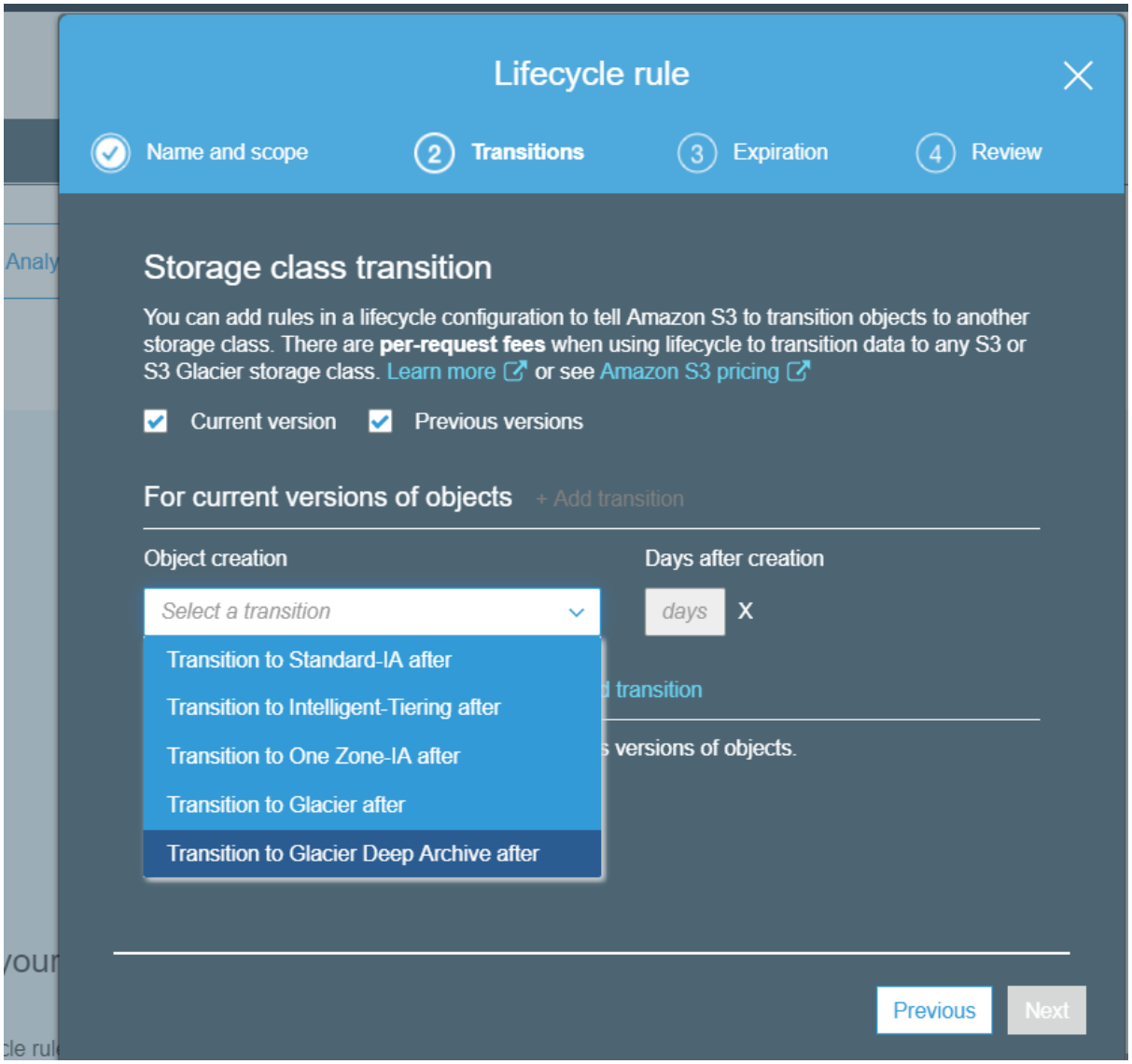


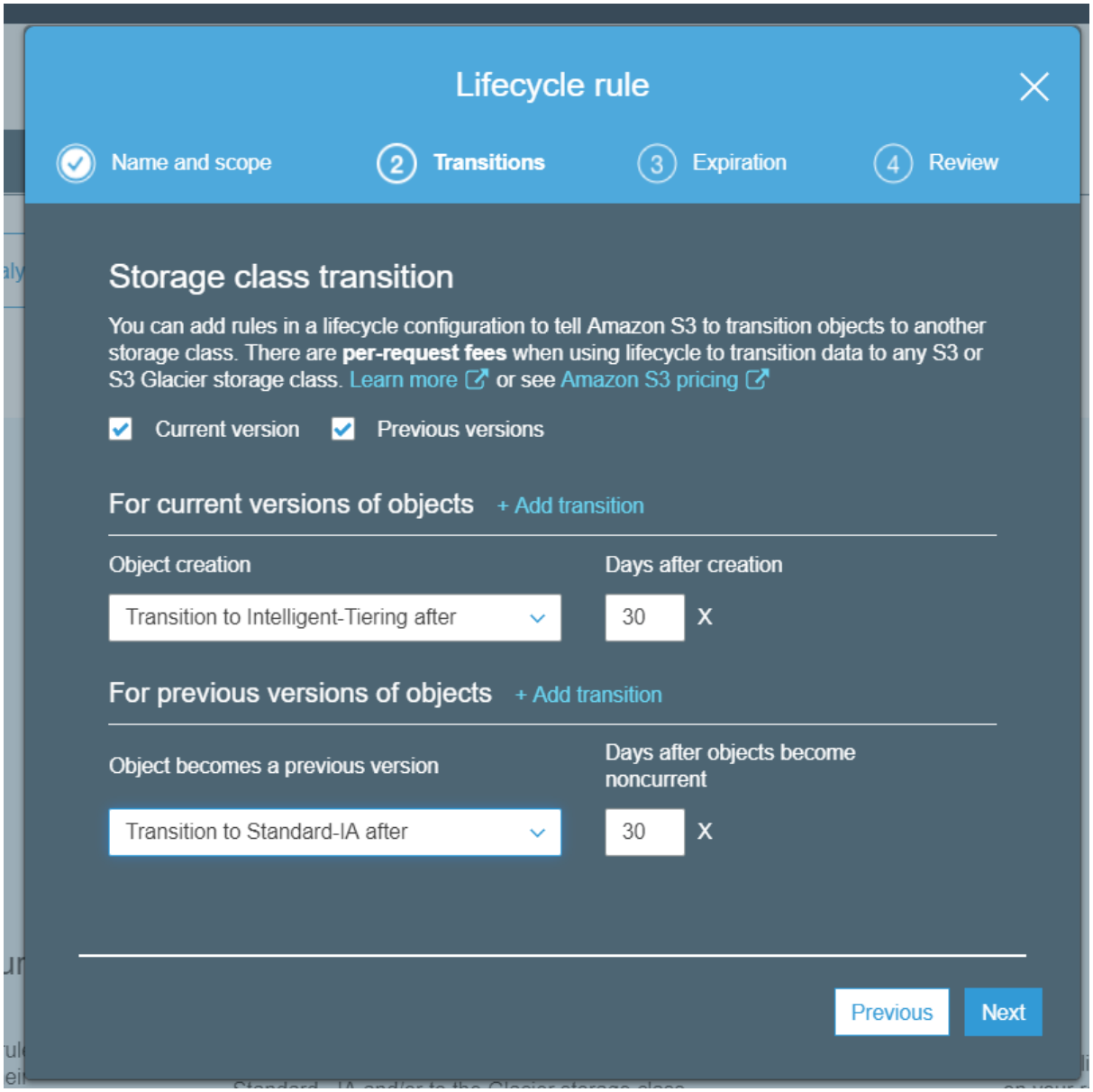
LIFE CYCLE MANAGEMENT

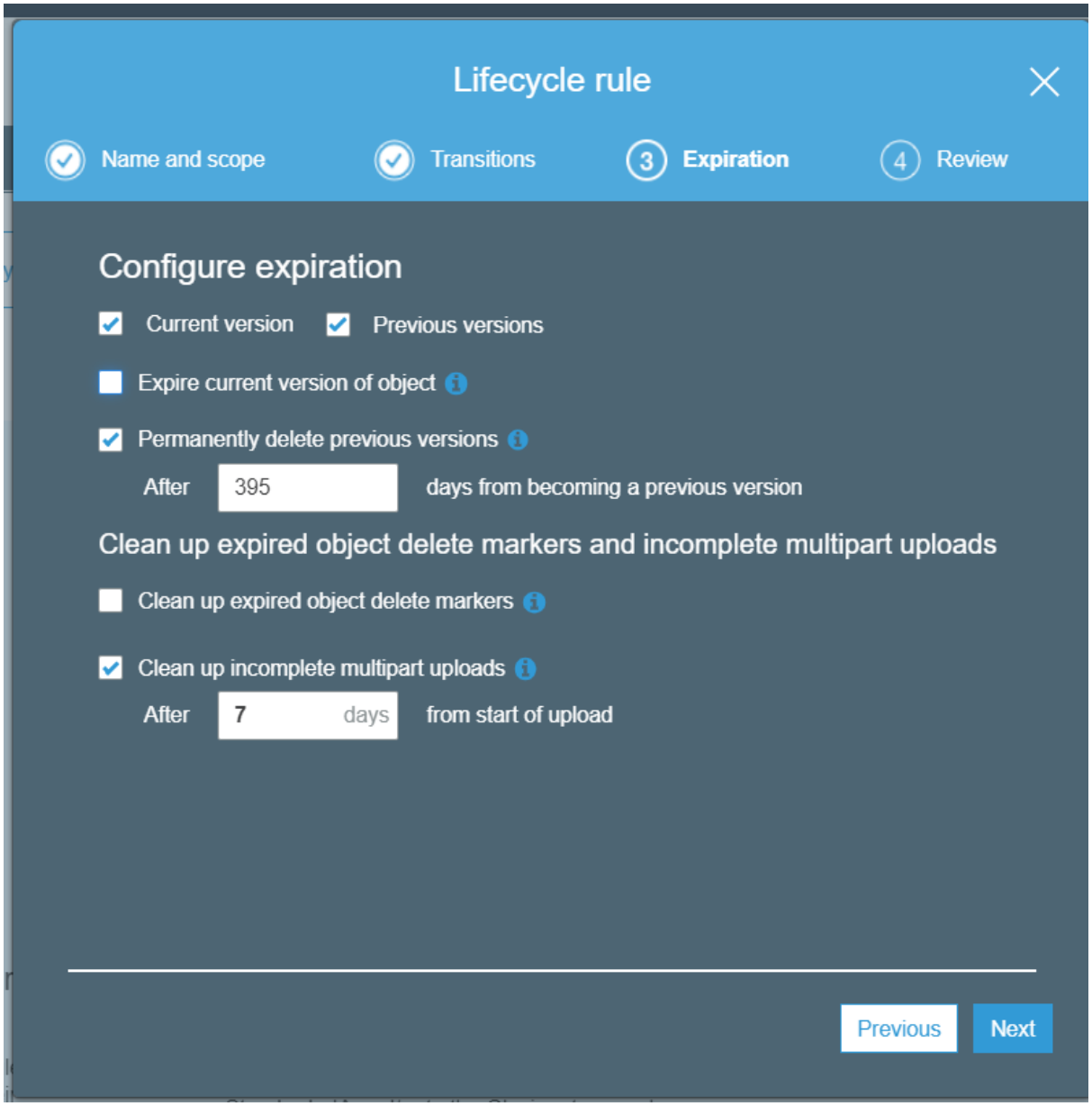
Life cycle management is a feature that helps us to keep data in S3 between different storage classes Depending upon how critical the data is, how frequently it is accessed, we can set time period when the data would be moved from one storage class to another.

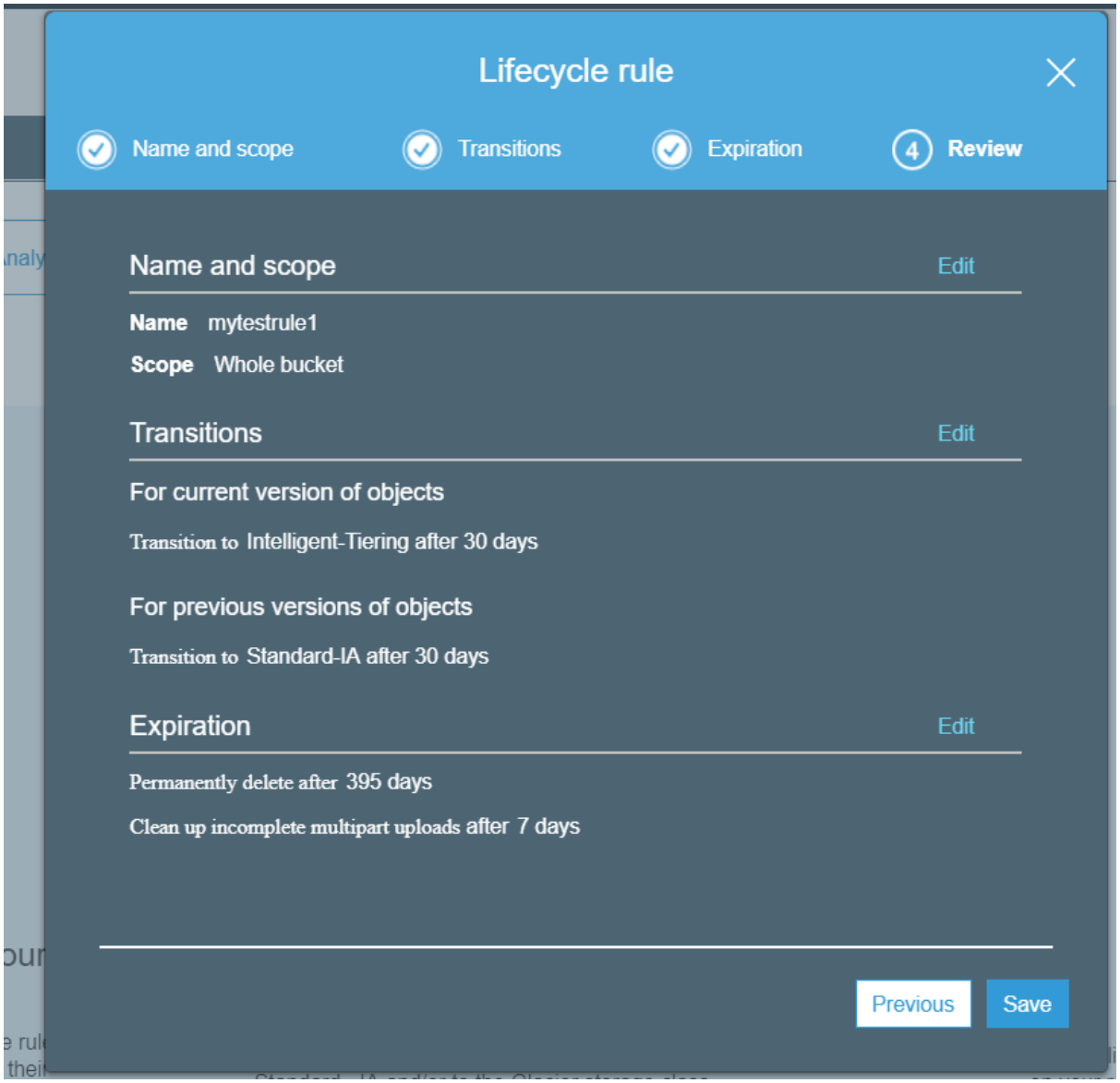



Groups
Properties
Actions
Lifecycle rule
1 Name and scope
2 Transitions
3 Expiration
4 Review
Enter a rule name
mytestrule1
Add filter to limit scope to prefix/tags
Type to add prefix/tag filter
Cancel
Next


Lifecycle rule
Name and scope
2 Transitions
3 Expiration
4 Review
Storage class transition
You can add rules in a lifecycle configuration to tell Amazon S3 to transition objects to another storage class. There are per-request fees when using lifecycle to transition data to any S3 or S3 Glacier storage class. Learn more or see Amazon S3 pricing
Current version
Previous versions
For current versions of objects
+ Add transition
Object creation
Days after creation
Select a transition
days
X
Transition to Standard-IA after
Transition to Intelligent-Tiering after
Transition to One Zone-IA after
Transition to Glacier after
Transition to Glacier Deep Archive after
transition
versions of objects.
Previous
Next


Lifecycle rule
Name and scope
2 Transitions
3 Expiration
4 Review
Storage class transition
You can add rules in a lifecycle configuration to tell Amazon S3 to transition objects to another storage class. There are per-request fees when using lifecycle to transition data to any S3 or S3 Glacier storage class. Learn more or see Amazon S3 pricing
Current version
Previous versions
For current versions of objects
+ Add transition
Object creation
Days after creation
Transition to Intelligent-Tiering after
30
X
For previous versions of objects
+ Add transition
Object becomes a previous version
Days after objects become noncurrent
Transition to Standard-IA after
30
X
Previous
Next

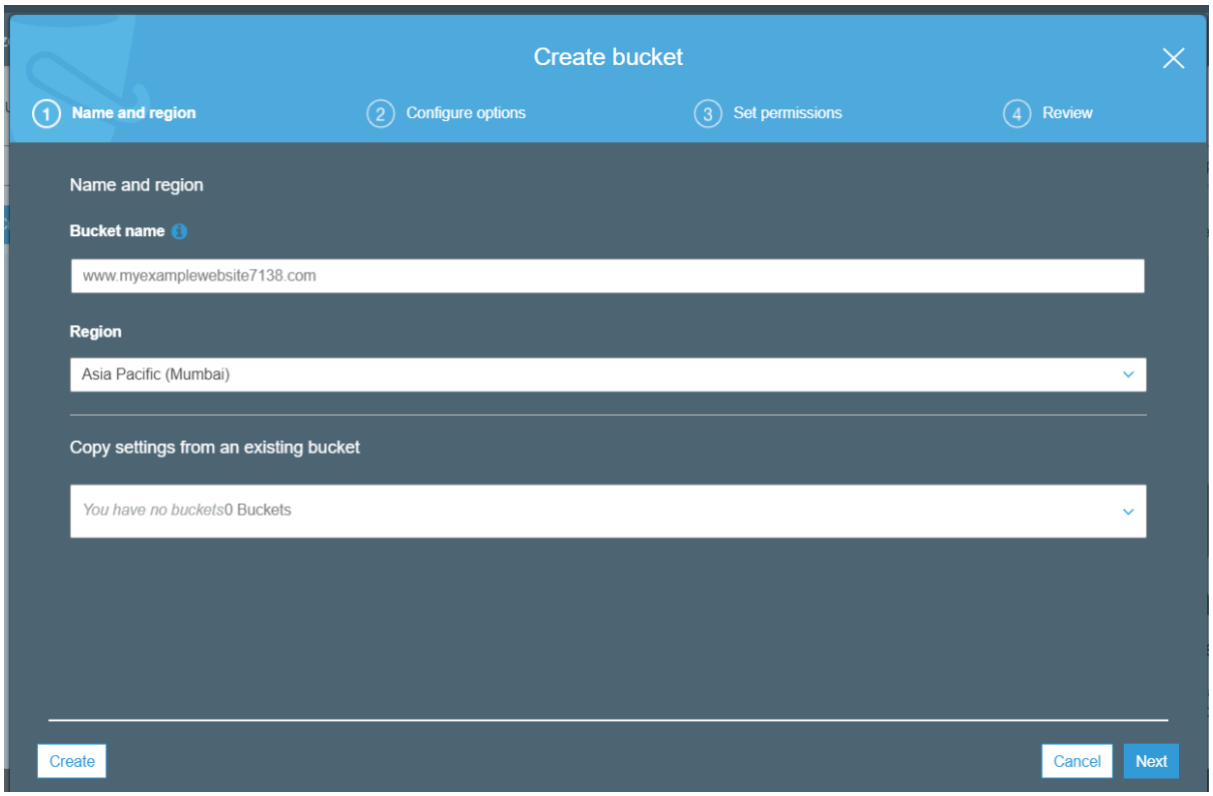

Lifecycle rule
Name and scope
Transitions
3
Expiration
4
Review
Configure expiration
Current version
Previous versions
Expire current version of object
Permanently delete previous versions
After
395
days from becoming a previous version
Clean up expired object delete markers and incomplete multipart uploads
Clean up expired object delete markers
Clean up incomplete multipart uploads
After
7
days
from start of upload
Previous
Next

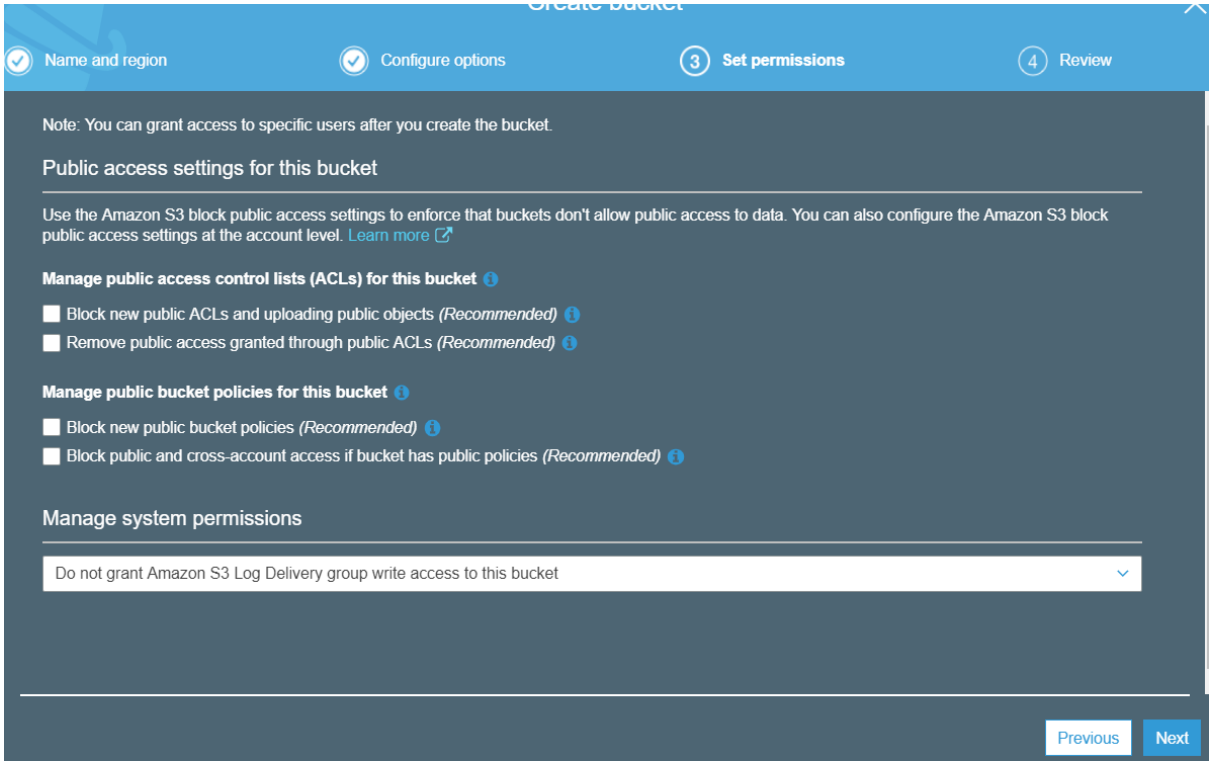

Lifecycle rule
Name and scope
Transitions
Expiration
4 Review
Name and scope
Edit
Name mytestrule1
Scope Whole bucket
Transitions
Edit
For current version of objects
Transition to Intelligent-Tiering after 30 days
For previous versions of objects
Transition to Standard-IA after 30 days
Expiration
Edit
Permanently delete after 395 days
Clean up incomplete multipart uploads after 7 days
Previous
Save

HOST A STATIC WEBSITE ON S3

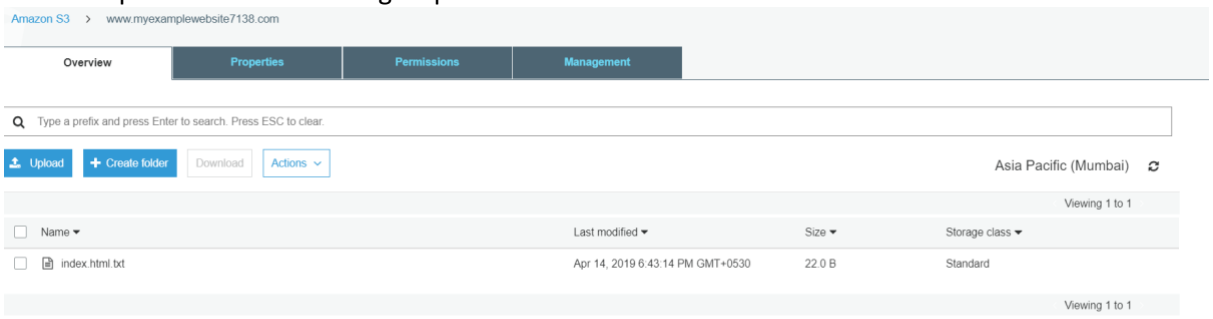
An S3 bucket can be used to host a static website. This is very economical way to do so .Lets see how we can do it.

- Create a new bucket with Public access.

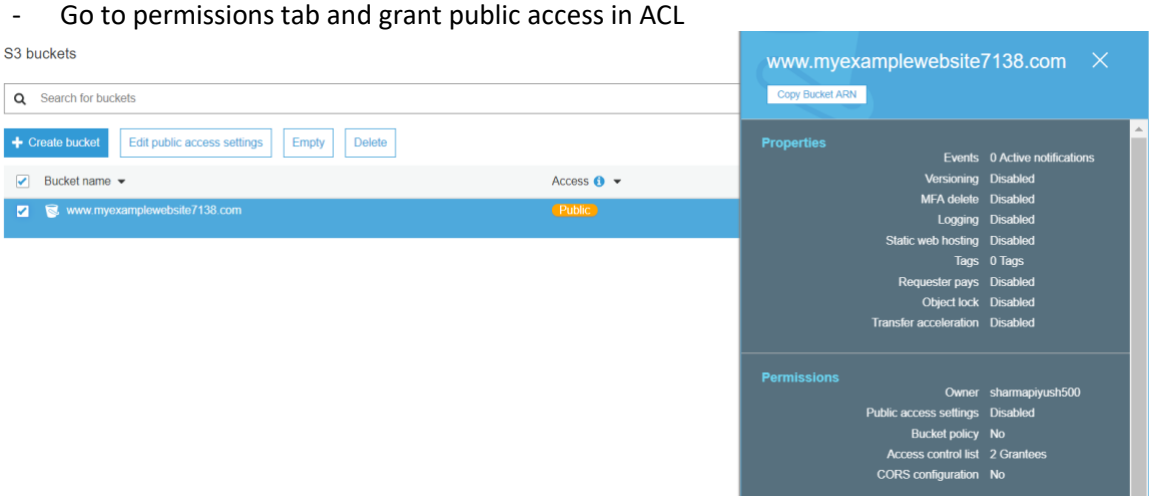


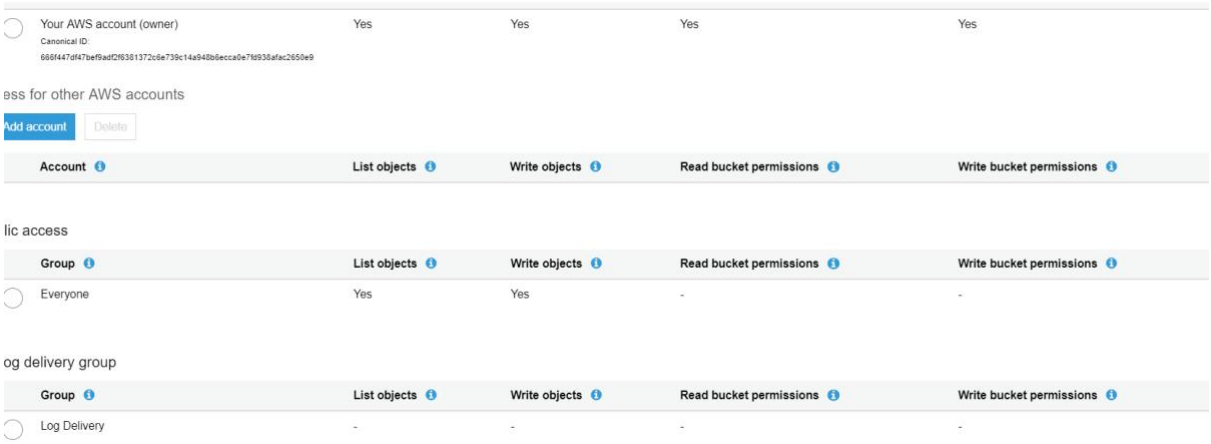


- Upload index.html and give public read access.

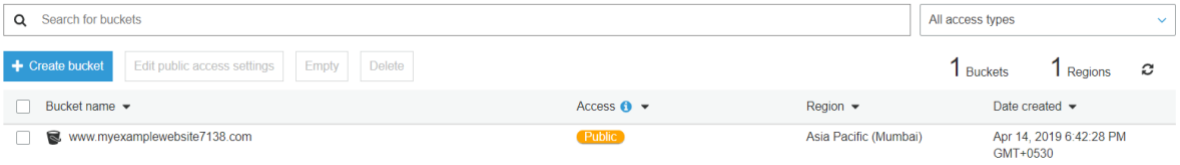


- Go to permissions tab and grant public access in ACL

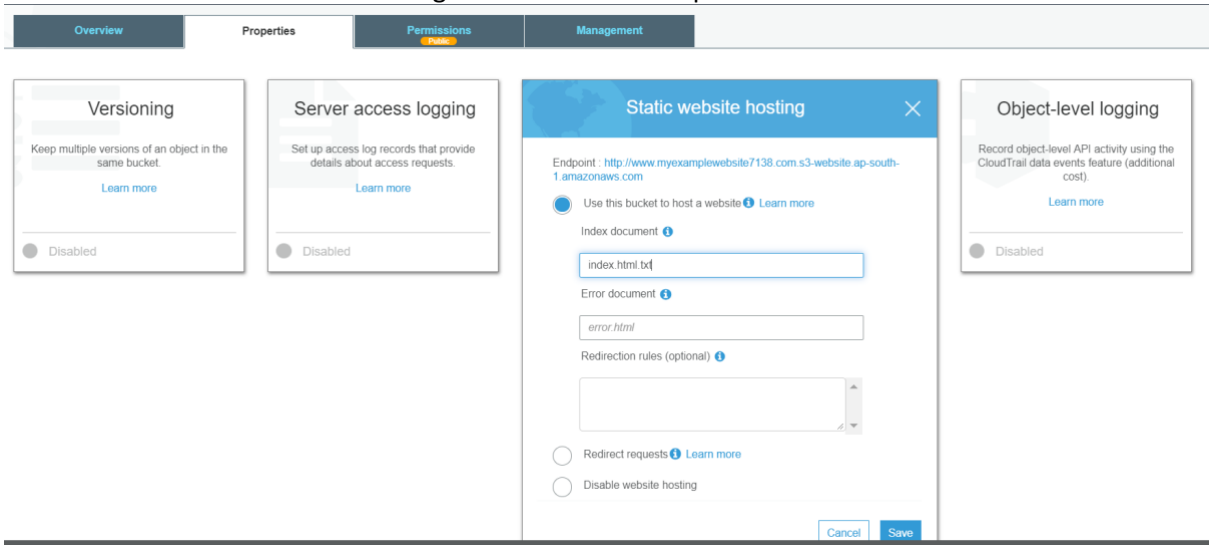




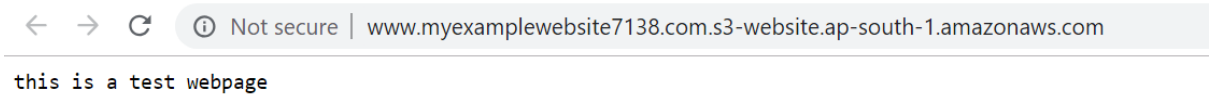
- Bucket should now look like this



- Enable static website hosting on the S3 bucket . Upload the file index.html.txt



- Clicking on the URL will open the webpage




```
this is a test webpage
```

S3 SECURITY

- By default all the newly created buckets are private
- For further security there are Bucket ACL and bucket policies

S3 ENCRYPTION

- **ENCRYPTION IN TRANSIT**

This relates to encryption of data in motion i.e. when it travels from one place to another. This is taken care by SSL/TLS

- **ENCRYPTION AT REST**

i) SERVER SIDE ENCRYPTION

A) SSE WITH AMAZON S3 MANAGED KEYS (SSE-S3)

Each object is encrypted with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

B) SSE WITH AWS KMS MANAGED KEYS (SSE-KMS)

Similar to SSE-S3, but with some additional benefits along with some additional charges for using this service. There are separate permissions for the use of an envelope key (that is, a key that protects your data's encryption key) that provides added protection against unauthorized access of your objects in S3. SSE-KMS also provides you with an audit trail of when your key was used and by whom. Additionally, you have the option to create and manage encryption keys yourself, or use a default key that is unique to you, the service you are using, and the region you are working in.

C) SSE WITH CUSTOMER PROVIDED KEYS (SSE-C)

You manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects.

ii) CLIENT SIDE ENCRYPTION

This is a process where the data is encrypted before being sent to S3.

S3 STORAGE GATEWAYS

AWS Storage Gateway connects an on-premises software appliance with cloud-based storage to provide seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure.

We have the below types of storage gateways

1) **FILE GATEWAY**

This is NFS based solution for data transfer between amazon s3 and on premises data centre.

We have a software appliance installed at our data centre that interacts with Amazon S3 and transfer data.

2) **VOLUME GATEWAY**

A volume gateway provides cloud-backed storage volumes that you can mount as Internet Small Computer System Interface (iSCSI) devices from your on-premises application servers. This is further divided in to two categories.

i) STORED VOLUMES

If you need low-latency access to your entire dataset, first configure your on-premises gateway to store all your data locally. Then asynchronously back up point-in-time snapshots of this data to Amazon S3. This configuration provides durable and inexpensive offsite backups that you can recover to your local data centre or Amazon EC2. For example, if you need replacement capacity for disaster recovery, you can recover the backups to Amazon EC2.

ii) CACHED VOLUMES

You store your data in Amazon Simple Storage Service (Amazon S3) and retain a copy of frequently accessed data subsets locally. Cached volumes offer a substantial cost savings on primary storage and minimize the need to scale your storage on-premises. You also retain low-latency access to your frequently accessed data.

3) **TAPE GATEWAY**

With a tape gateway, you can cost-effectively and durably archive backup data in Amazon S3 Glacier or S3 Glacier Deep Archive. A tape gateway provides a virtual tape infrastructure that scales seamlessly with your business needs and eliminates the operational burden of provisioning, scaling, and maintaining a physical tape infrastructure.

⇨ **POINTS TO REMEMBER –S3**

- Amazon S3 is a simple key-based object store. When you store data, you assign a unique object key that can later be used to retrieve the data. Keys can be any string, and they can be constructed to mimic hierarchical attributes. Alternatively, you can use S3 Object Tagging to organize your data across all of your S3 buckets and/or prefixes.
- **S3 INTELLIGENT TIERING:** Amazon S3 Intelligent-Tiering (S3 Intelligent-Tiering) is an S3 storage class for data with unknown access patterns or changing access patterns that are difficult to learn. It is the first cloud storage class that delivers automatic cost savings by

moving objects between two access tiers when access patterns change. One tier is optimized for frequent access and the other lower-cost tier is designed for infrequent access.

- **S3 STORAGE CLASS ANALYTICS:** With Storage Class Analysis, you can analyze storage access patterns and transition the right data to the right storage class. This new S3 feature automatically identifies infrequent access patterns to help you transition storage to S3 Standard-IA. You can configure a Storage Class Analysis policy to monitor an entire bucket, prefix, or object tag. Once an infrequent access pattern is observed, you can easily create a new S3 Lifecycle age policy based on the results.
- **S3 INVENTORY:** You can configure S3 Inventory to provide a CSV, ORC, or Parquet file output of your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or prefix. You can simplify and speed up business workflows and big data jobs with S3 Inventory. You can also use S3 inventory to verify encryption and replication status of your objects to meet business, compliance, and regulatory needs.

AMAZON EC2 (ELASTIC COMPUTE CLOUD)

INTRODUCTION

- ➔ EC2 instances are nothing but servers in AWS cloud ecosystem.
- ➔ Provides resizable capacity in cloud.
- ➔ EC2 service allows us to launch new servers in a matter of minutes and hence scale our infra quickly.

⇨ **EC2 PRICING OPTIONS**

1) **ON DEMAND**: We pay for what we use in this pay as you go model.
No long-term commitments or upfront payments are required.

2) **SPOT INSTANCES**: Here the billing is just like share market. There is an AWS market price and a price quoted by us, which will be called as spot price.
AWS keeps on changing the market price of the instances depending upon the demand.
Price is directly proportional to the demand.

- Example: Let's say AWS market price for an instance is 2$ and user made a bid of 3$. Now as long as the AWS market price is < 3$, instance will be up and running otherwise it will get terminated. User will be charged as per AWS market price and not what he bid for.
- We can also request for spot fleet instances, here we can request for "n" number of instances at once.
- Request for duration/Blocked spot instances: Here we can block spot instances to run for a particular duration ranging from 1 to 6 hours. Here whatever happens to the price , our instance will not be terminated before 6 hours.
- In case the market price becomes more than the bid , instance will be terminated with a 2 min warning.
- If instance is terminated by AWS, we are not charged for the partial hour, if done by us, we are charged for that partial hour.

3) **RESERVED INSTANCES**: In scenarios where we are sure about our capacity requirements , we can reserve a an instance for 1 and 3 yrs, We get discount as compared to on demand instances.

4) **DEDICATED INSTANCES:** Dedicated Instances are Amazon EC2 instances that run in a VPC on hardware that is dedicated to a single customer. Your Dedicated instances are physically isolated at the host hardware level from instances that belong to other AWS accounts. Dedicated instances may share hardware with other instances from the same AWS account that are not dedicated instances.

5) **DEDICATED HOSTS:** These are hosts where we can launch Amazon EC2 instances on physical servers that are dedicated for your use. Dedicated Hosts give you additional visibility and control over how instances are placed on a physical server, and you can reliably use the same physical server over time. As a result, Dedicated Hosts enable you to use your existing server-bound software licenses like Windows Server and address corporate compliance and regulatory requirements.

⇨ **EBS VOLUMES**

- EBS volumes are ISCSI drives that are used to store data.
- It stands for Elastic block storage and as the name says, it's a block-based storage used to install operating systems or databases.
- Advantages are data availability, data persistence (Volumes persists irrespective of the instance lifecycle, snapshots, flexibility.

➔ **EBS VOLUME TYPES**

1) GENERAL PURPOSE SSD (GP2)

- Balances both price and performance
- 1 GiB to 16 Tib size range.
- Solid state drives
- 3 IOPS /GB

2) PROVISIONED IOPS SSD (IO1)

- Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads.
- Recommended for high workload databases such as Mongo DB , Cassandra, Mysql etc.

3) THROUGHPUT OPTIMISED HDD (ST1)

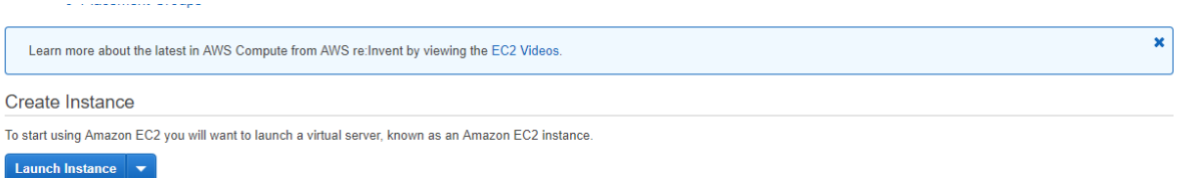
- Low-cost HDD volume designed for frequently accessed, throughput-intensive workloads
- Cannot be boot volumes.
- Can be used for big data and data warehousing.

4) **COLD HDD(SC1)**

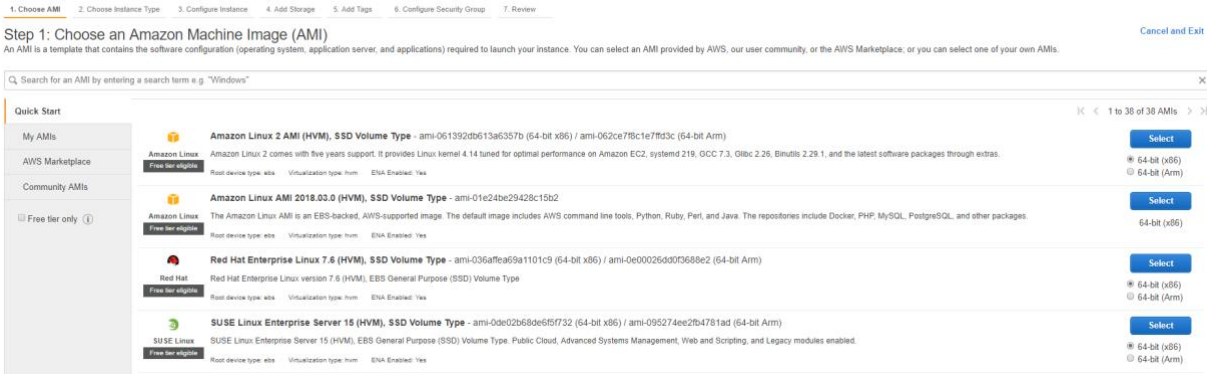
- Lowest cost HDD volume designed for less frequently accessed workloads
- Useful where lowest storage cost is important.
- Cannot be a boot volume.

⇨ **LAUNCHING AN EC2 INSTANCE**

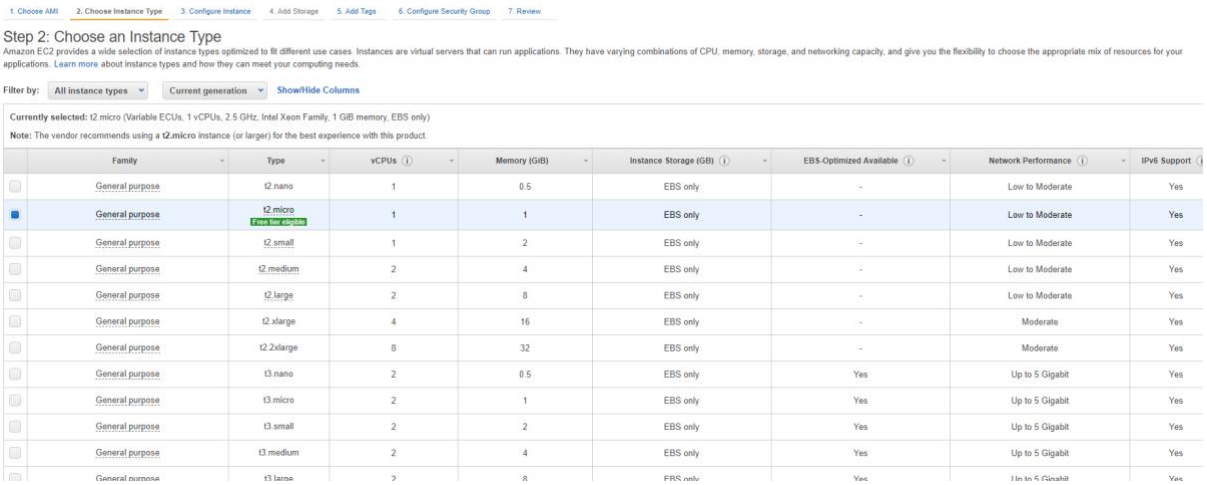
1) Go to Services -> EC2-> Launch Instance



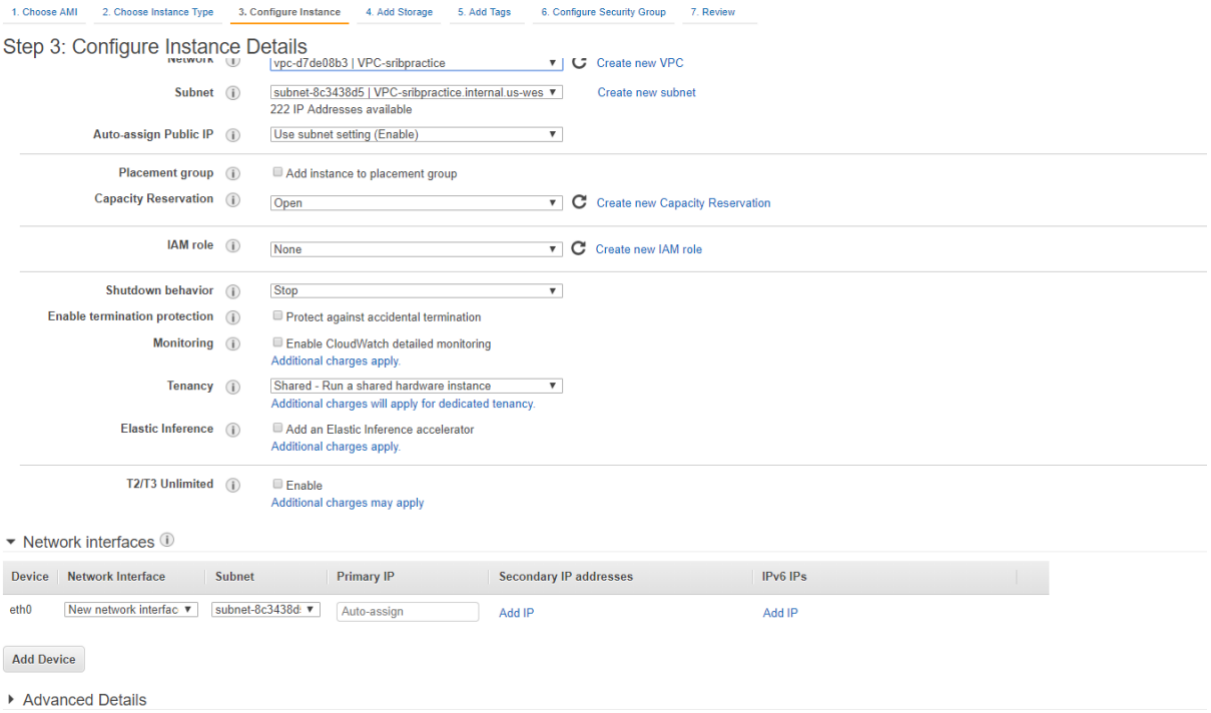
2) Select the AMI (Amazon Machine Image) from which the server needs to be launched. We have large number of AMI to choose from with different operating systems. For testing purpose select an AMI, which is free, tier eligible.



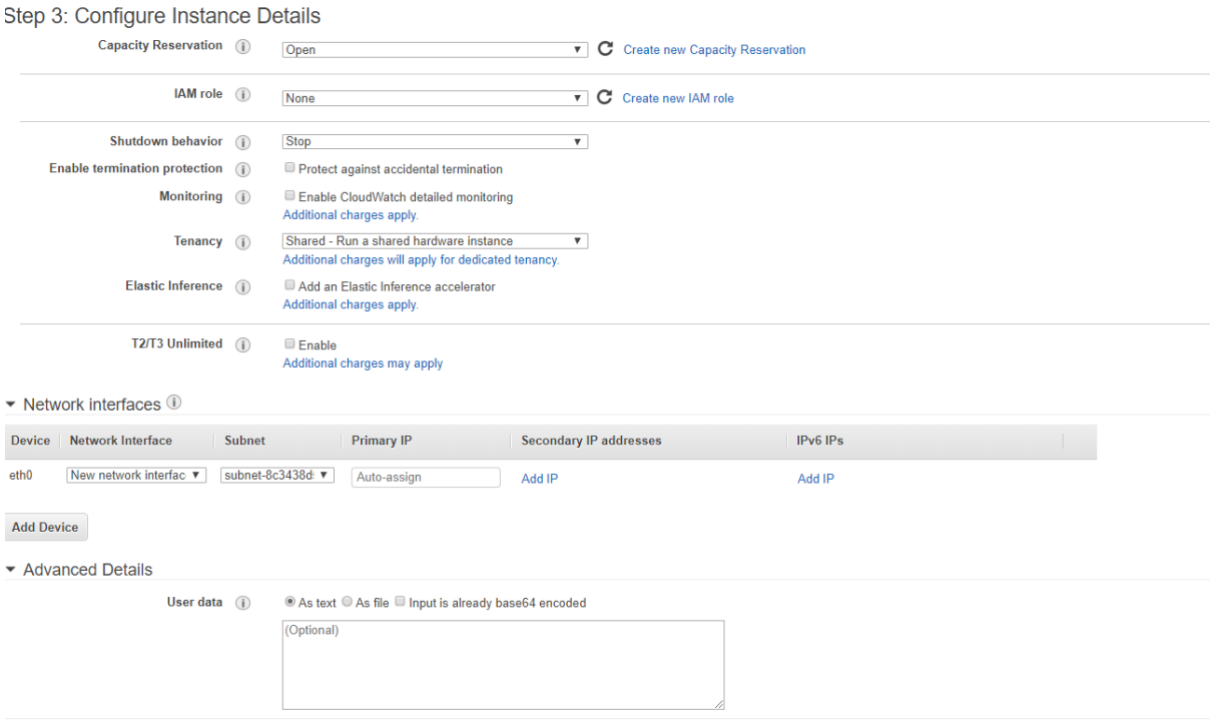
3) Select the instance type . We have large number of instance types to choose from with different configurations (RAM , CPU etc) and we can choose according to our requirement.For testing purpose we will select “T2 micro” which is free tier eligible.



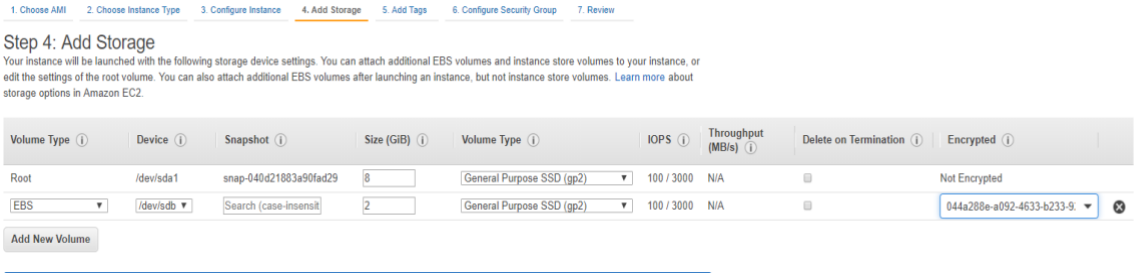
4) Configure the instance details . Choose the number of instances to launch, VPC , subnet, security group , role etc.



4.1) There is a section with the name “Advanced details”. Here we can put any configuration settings or the bash scripts that we want to execute as the system boots up.



5) Add the storage, a disk on which the operating system will be installed. We can attach additional disks as well as per our requirement.

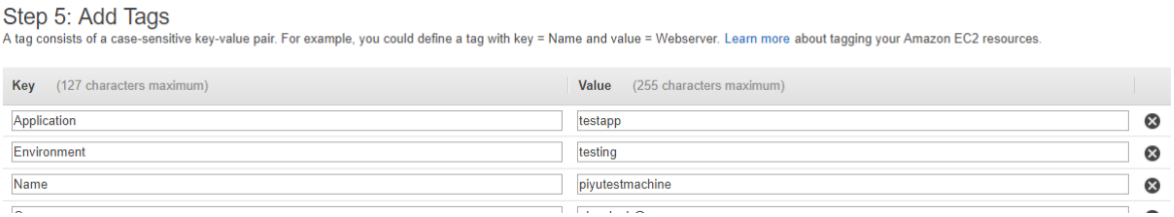


Here we have added one additional disk of 2 GB for storing additional data.

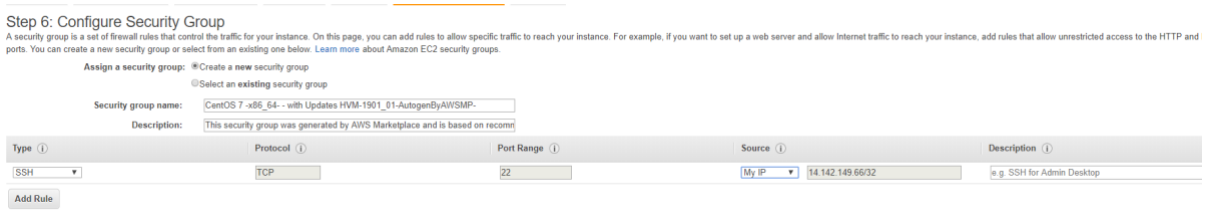
NOTE : On the root disk , there is an option that we can check .”**Delete on Termination**”. If we check this , the root volume will be deleted when the server will be terminated, by default all ebs volumes persists even after the system is terminated.

NOTE: Root disk is not encrypted by default; also, there is no way to do it in the start. Whereas other Ebs volumes that we attach, we can encrypt them.

6) Add Tags



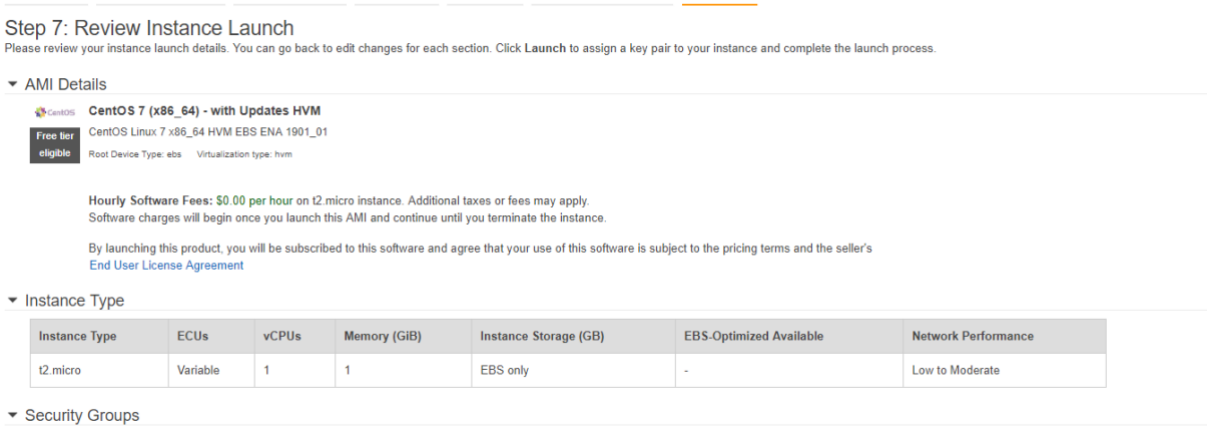
7) Configure security groups



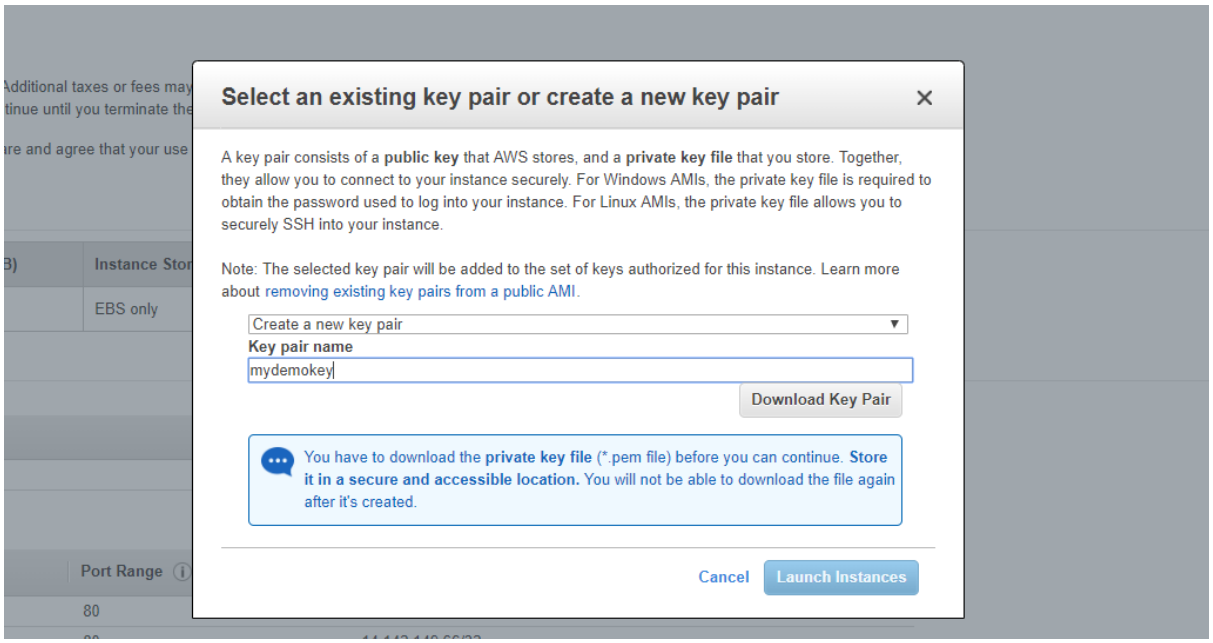
Security groups are a way to allow or disallow traffic. We can add rules as per our requirements.

It is never recommended to make a sever public facing or exposed to the internet.

8) Review and launch



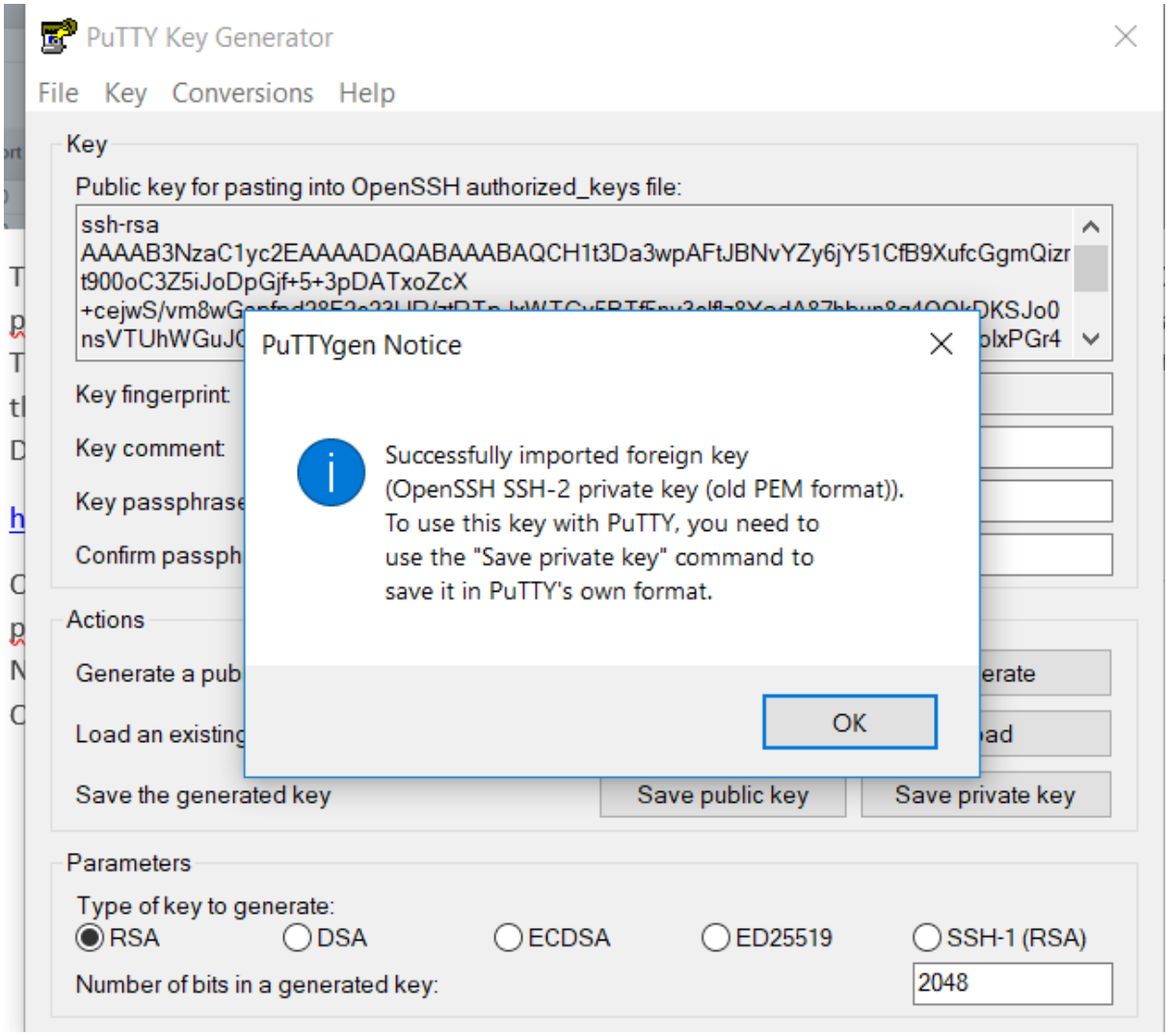
9) Click on “launch” and create a new key pair, download the key.



10) The key that we downloaded is in the “pem” format . In case we are working on MAC OS, the pem format will work , on Windows server while using putty we need to use .ppk format .
11) To do the same we will download puttygen.exe file, we will upload the pem file and get back the ppk format key file.
12) Download puttygen from here

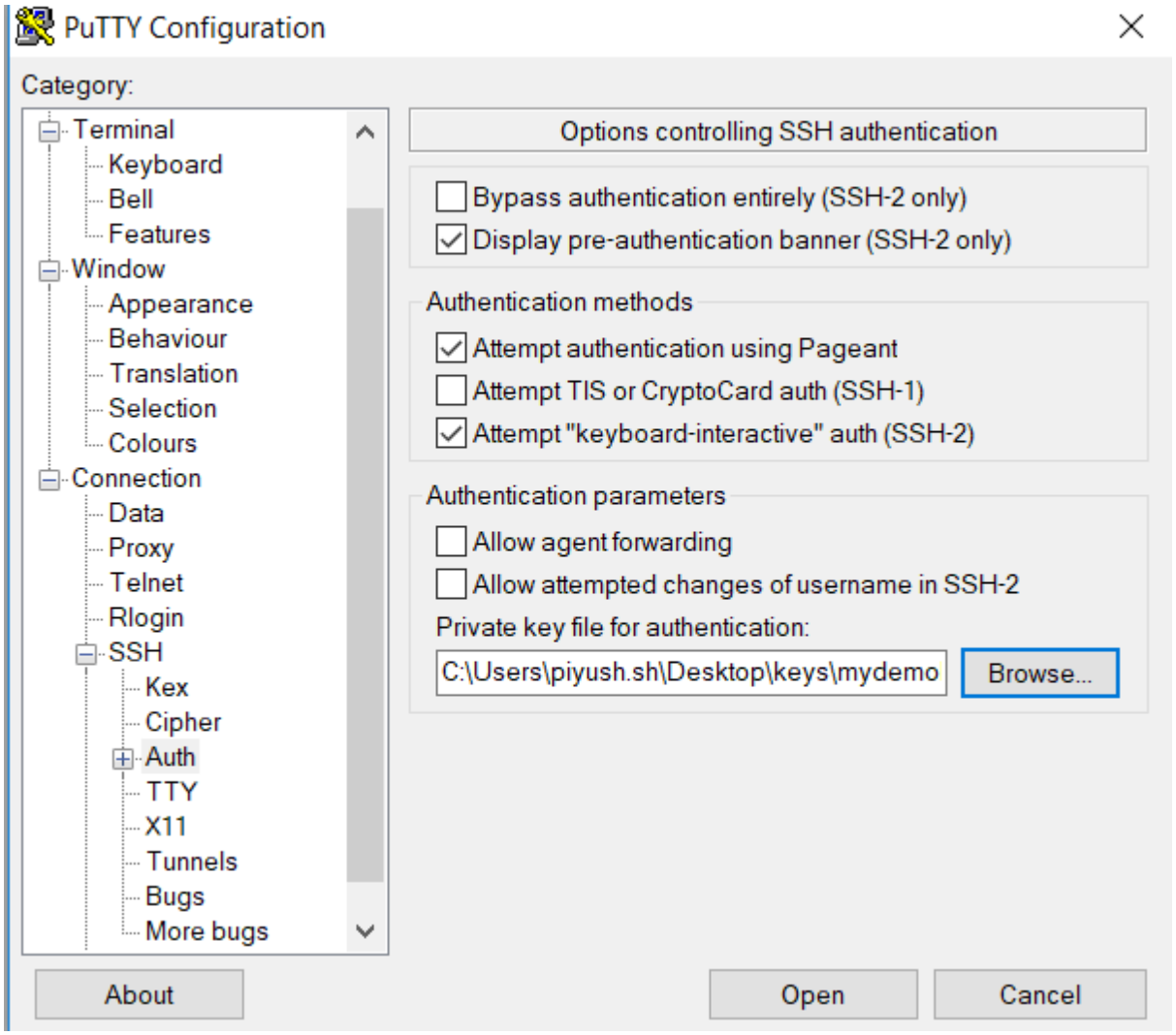
    https://puttygen.com/download.php?val=49

13) Open puttygen and upload the pem keys , convert to ppk and download the ppk format privaye keys. Save it with and extension of .ppk



14) Now open putty and give the public IP of our instance.

15) On the left side of putty , select ssh-> Auth and provide the key in ppk format.



16) The login prompt will appear and here type username as " centos" and login should be successful.

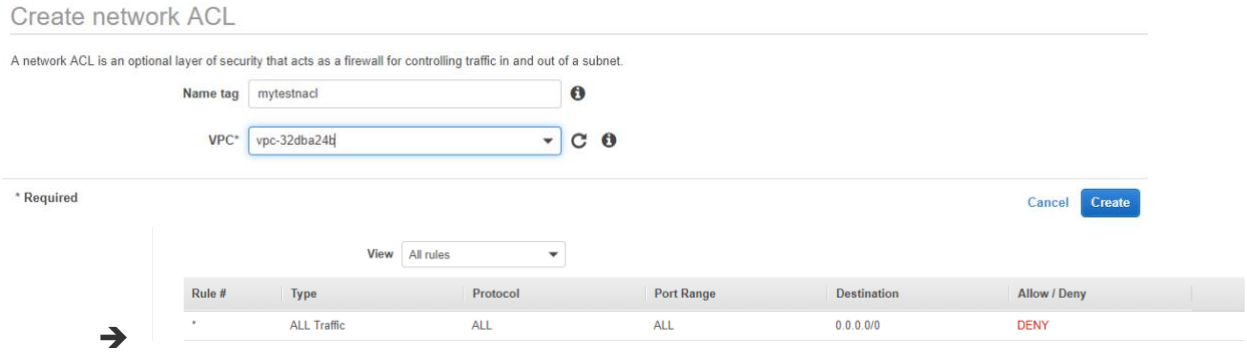
➔ **MORE ABOUT SECURITY GROUPS AND NETWORK ACL (NACL)**

These services allow us to set firewall rules, it lets us decide what traffic we should allow to our server and what to negate.It is the level of granularity at which you want to restrict access to your instances.

There are a couple of points to note here :

1. Network Access control lists are applicable at the subnet level, so any instance in the subnet with an associated NACL will follow rules of NACL. That's not the case with security groups, security groups has to be assigned explicitly to the instance.
2. By default your default vpc, will have a default Network Access Control List which would allow all traffic , both inbound and outbound. If you want to restrict access at subnet level it's a good practice to create a custom NACL and edit it with the rules of your choice and while editing you could also edit subnet associations , where you associate a subnet with this custom NACL, now any instance in this subnet will start following NACL rules
3. NACLs are stateless unlike security groups. Security groups are statefull ,if you add an inbound rule say for port 80, it is automatically allowed out, meaning outbound rule for that particular port need not be explicitly added. But in NACLs you need to provide explicit inbound and outbound rules
4. In security groups you cannot deny traffic from a particular instance, by default everything is denied. You can set rules only to allow. Where as in NACLs you can set rules to both deny and allow. There by denying and allowing only the instances of your choice. You get to take decisions that are more granular.
5. Security groups evaluate all the rules in them before allowing a traffic . NACLs do it in the number order, from top to bottom like, if your rule #0 says allow all HTTP traffic and your rule #100 says don't allow HTTP traffic from ip address 10.0.2.156 , it's will not be able to deny the traffic, because rule #0 has already allowed traffic. So it's good practice to have your deny rules first in NACL and followed by allow rules. AWS best practice is to number your rules in increment of 100s in NACL. By deny rules first, I mean, specifying narrow deny rules, like for specific ports only. And then write your allow rules.

➔ Network ACLs are applicable on subnet level , to create a new NACL , go to VPC service and select NACL



➔

➔ Edit the rules as per your requirement.

NOTE: One subnet corresponds to 1 AZ
One instance can be associated with multiple instances
Security groups are effective as soon as they are attached.

⇨ **SYSTEM STATUS CHECK VS INSTANCE STATUS CHECK**

System status check: Checks the underlying hypervisor, verifies that your instance is reachable, it is checked that network packets can be sent to the instance. The failure signifies the failure of underlying hardware infra.
To rectify it , we power off and power on the instance and then instance is started on a different host.

Instance status check verifies that OS and application are accepting traffic.

NOTE: Termination protection is turned off by default, it should be turned on .

➔ **CONFIGURE AND INSTALL AWSCLI ON A FRESH LINUX INSTANCE**

Login with root and execute below command
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum intall python2-pip
yum install python2-pip
pip install awscli

➔ If we attach a role to an instance with all the required permissions, we need not provide the keys, otherwise
aws configure
This command is used to configure the awscli
We have an instance with the s3-admin-role attached to it so here we will see that without providing the keys , we will be able to execute commands

```
[root@ip-172-31-18-226 opt]# aws s3 ls
2019-04-14 13:12:28 www.myexamplewebsite7138.com
[root@ip-172-31-18-226 opt]# aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: ap-south-1
Default output format [None]: json
[root@ip-172-31-18-226 opt]# aws s3 ls
2019-04-14 13:16:43 www.myexamplewebsite7138.com
[root@ip-172-31-18-226 opt]# aws s3 mb s3://mybengalurubucket7138
make_bucket: mybengalurubucket7138
[root@ip-172-31-18-226 opt]# aws s3 ls
2019-04-18 11:45:54 mybengalurubucket7138
2019-04-14 13:16:43 www.myexamplewebsite7138.com
```

➔ We get an error when we try to execute something related to ec2 because the role attached don't allow us to do that.

```
[root@ip-172-31-18-226 opt]# aws ec2 describe-instances

An error occurred (UnauthorizedOperation) when calling the DescribeInstances
operation: You are not authorized to perform this operation.
```

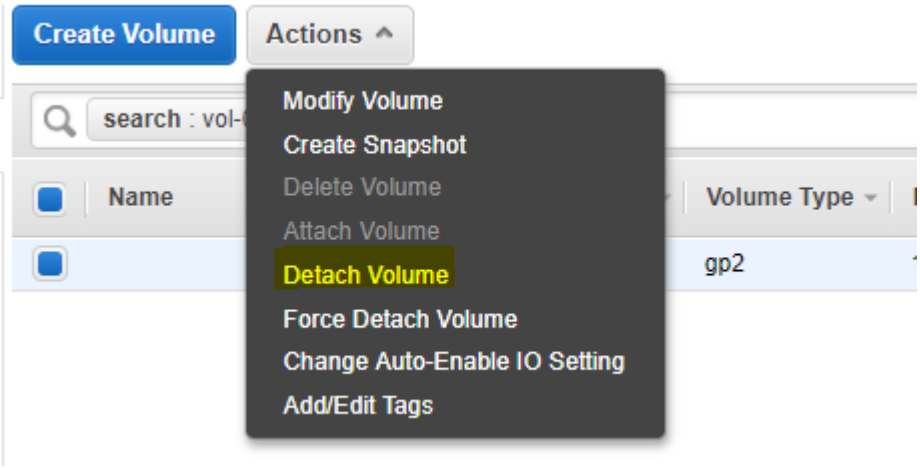
⇨ **CAPACITY RESERVATIONS**

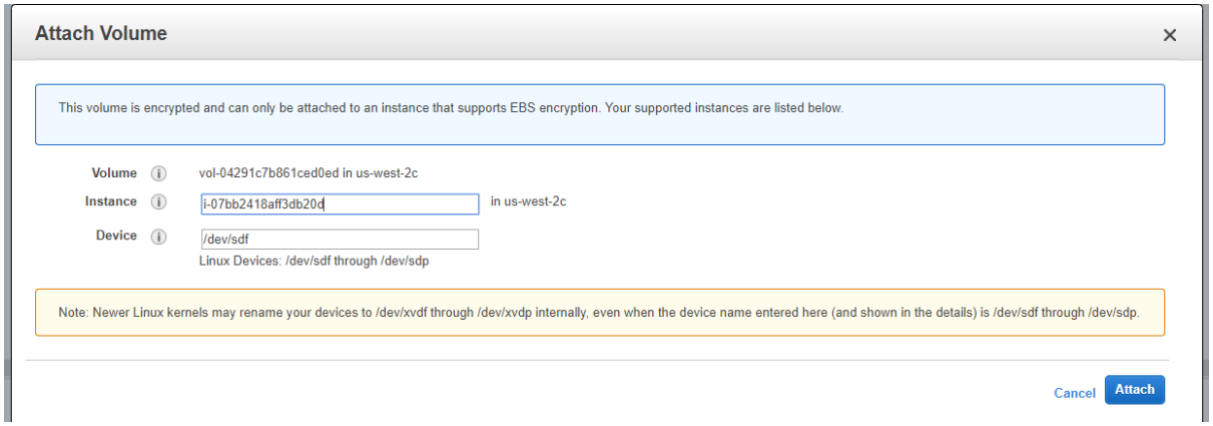
- Creating a Capacity Reservation in your account creates a capacity reservation in a specific Availability Zone. After it is created, you can launch instances into the reserved capacity as needed.
- The request can fail if AWS is short of capacity
- Request can fail if the number of instances we launch exceeds the number of on demand instances we can launch.
- Capacity reservation can be ended manually or after a specific time that we specify.
- Instance eligibility can be
- **Open**: The Capacity Reservation matches any instance that has matching attributes (instance type, platform, and Availability Zone). If you launch an instance with matching attributes, it is placed into the reserved capacity automatically.
- **Targeted**—The Capacity Reservation only accepts instances that have matching attributes (instance type, platform, and Availability Zone), and explicitly target the reservation.

➔ **ATTACHING AND DETACHING A VOLUME**

Here we will discuss, how we can convert ebs volume of one type to another type. For demo purpose, we have created a /data1 file system on our instance and created a text file. We have used the second disk of 2 GB that we attached earlier.

1) We will work on the /dev/vxdb volume , from the instance details , fetch the volume ID which is vol-04291c7b861ced0ed in our case .
2) Select the volume and detach volume

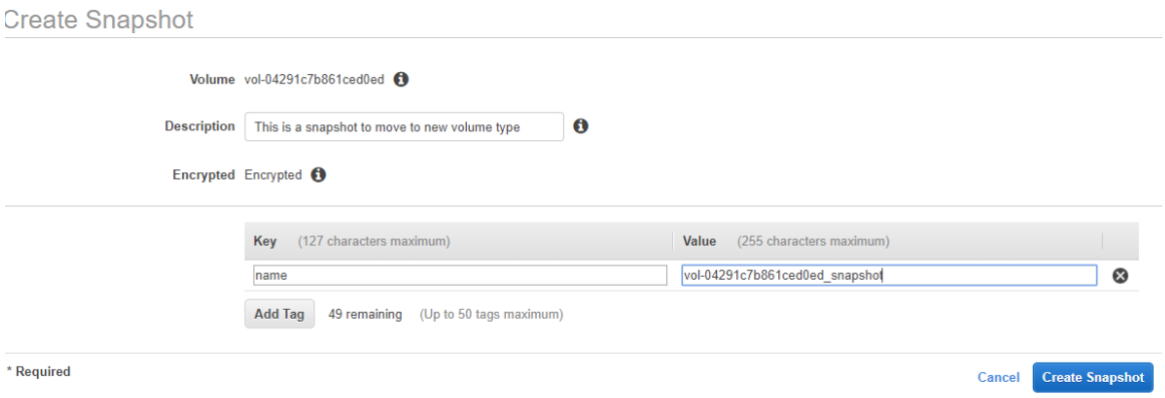




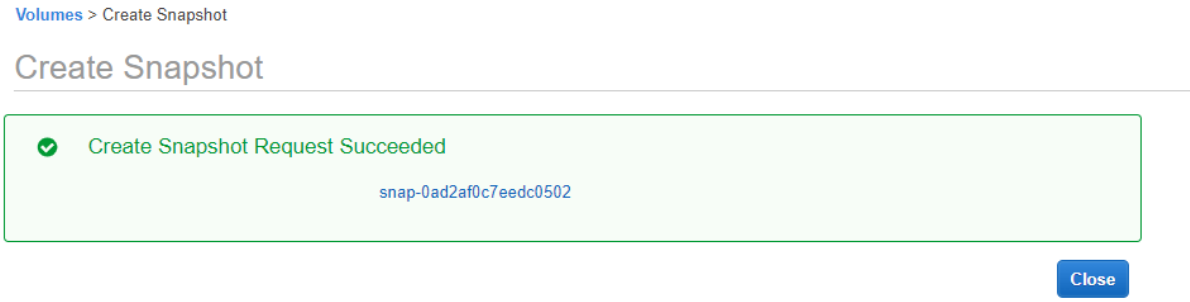
3) Now login to the server, execute fdisk –l and mount the disk again.
4) Attached volumes have “in-use” status, detached volumes are in “available” state.
   NOTE: EC2 instance and EBS volume should be in the same AZ

➔ **CONVERT ONE TYPE OF EBS VOLUME TO ANOTHER**
Here as well, we will use the second disk attached and change its type.

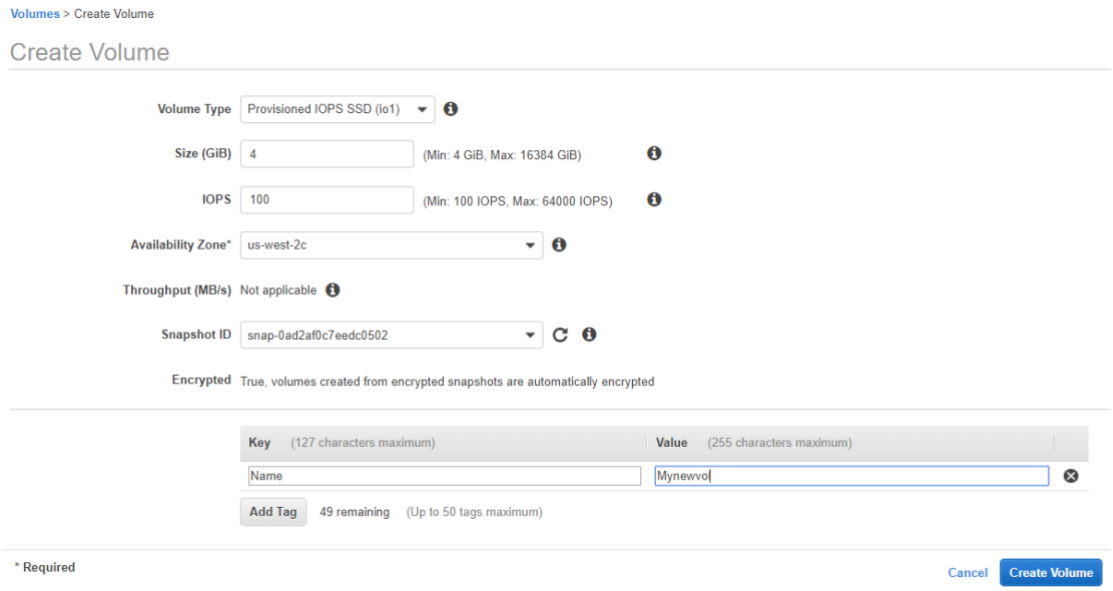
1) Unmount the disk /dev/svdf
2) Take a snapshot of that volume .



3) Note the snapshot ID



4) Now create a new EBS volume of a different type using the snapshot that we just created.
   Go To EC2-> Elastic Block Storage -> Volumes -> Create volume



Choose the snapshot that we created
Also, do not forget to select the same AZ as that of the EC2

5) Now detach the existing volume, and attach the new volume.

NOTE: Volumes created from snapshots are encrypted by default.
EBS volumes can only be scaled up in size and not down.
Snapshots from encrypted volumes are encrypted
Volumes restored from encrypted volumes are encrypted.

6) After attaching the disk, do mount –a and file system will be mounted.

⇨ **ENCRYPTING ROOT VOLUME DISKS**

As we discussed earlier, root volumes are not encrypted by default, so we will now see how we can do it.
This cannot be done online so the server needs to be powered off.

1) Power off the server.
2) Create snapshot of the root volume and note that this will be an unencrypted snap.
3) Create a copy of this unencrypted snapshot and then choose to encrypt it.
4) So now we will have an encrypted snapshot.
5) Create a new volume from the encrypted snapshot and that should be in the same AZ as the server.
6) Detach the old volume from the instance and attach the encrypted volume
7) While attaching name of the volume should be /dev/sda1
8) Power on the server

AMAZON MACHINE IMAGES (AMI)

We can anytime create an image from our existing instances. Lets assume that we have an instance running apache server hosting our website, we can do all the configuration and create an AMI out of it. In case we need to launch more servers , we can do it with AMI .

➔ We can select an AMI based on various choices such as Operating system, architecture and the type of root device.

➔ There are two types of AMI available, one with EBS backed volumes and other with instance store backed volumes.

➔ Instance store volumes are far more less flexible than ebs . Instance store volumes have the below shortcomings.

- Additional volumes can only be attached at the time of launching the instance and not after that.
- Instance store volume instances cannot be stopped, they can only be rebooted or terminated. This is a negative point because in case where system status check fails, we have no option to stop and start the instance.
- These cannot be encrypted.
- Snapshots cannot be taken
- These types of instances take longer to get provisioned.
- They are also called as ephemeral storage because they persists as long as server is alive.
- Instance store backed volumes cannot be attached from one to other EC2 instance.

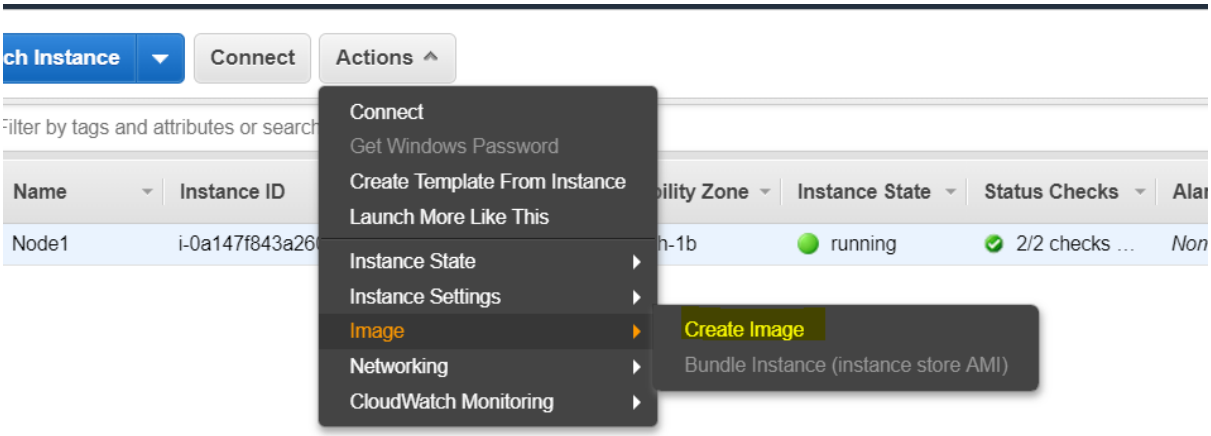
➔ **<u>CREATING AN AMI</u>**

1) Just for demo purpose, we have created one instance , install httpd rpm and deployed a simple webpage

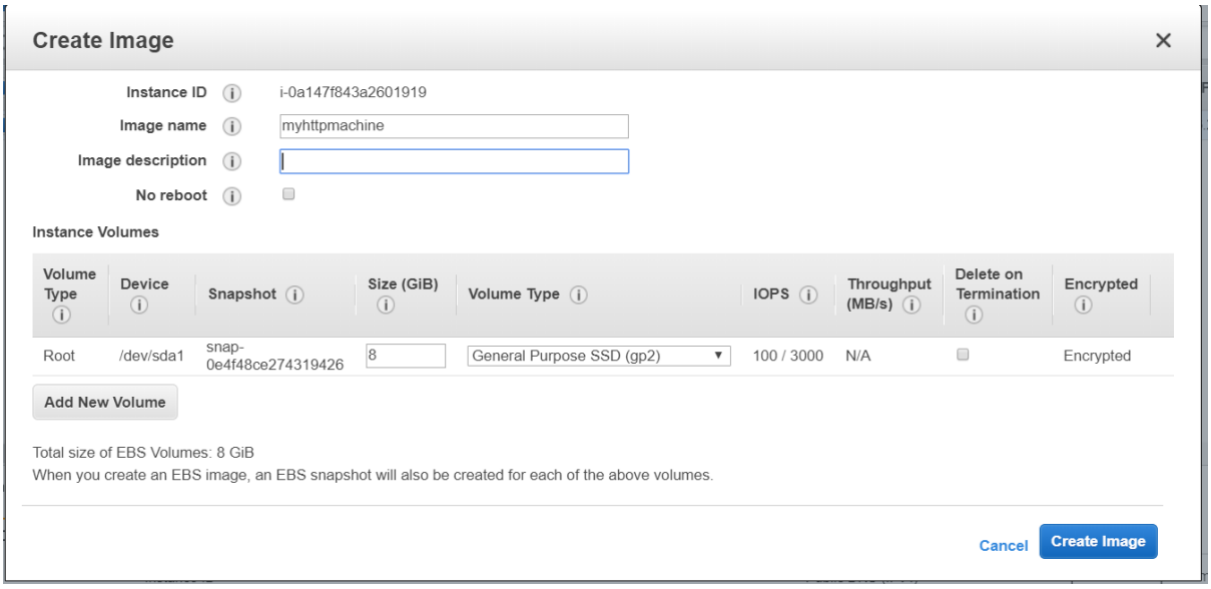
[root@ip-172-31-12-167 html]# curl http://ip-172-31-12-167

ThIs is a test page

2) To create an AMI always wherever possible, stop the instance and then create an image for consistency of data, though it is possible to create an image from a running machine.
3) Select the instance, Actions-> Image->Create Image

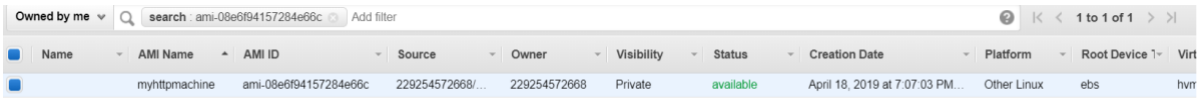


4) Fill the details

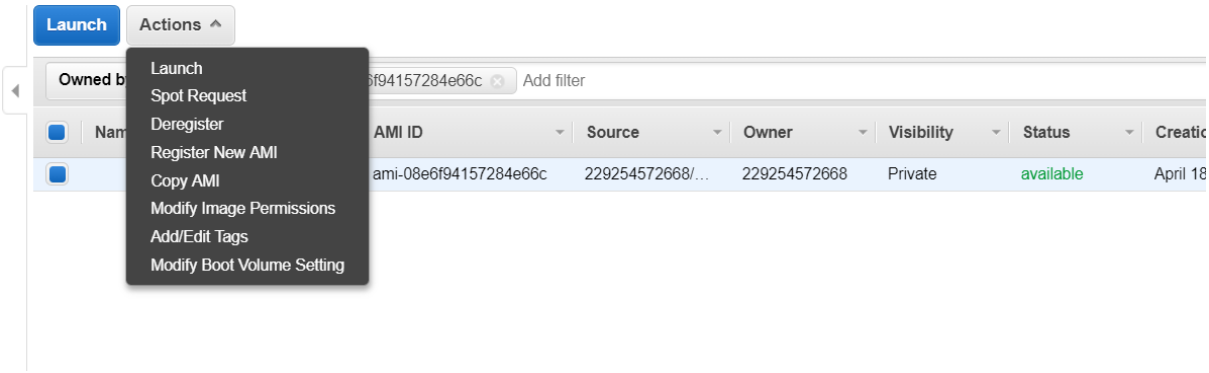


- No reboot option when checked , the instance is not shut down before taking a snapshot

5) AMI will be created. Check in “Owned by me”



6) These are the various options that we can have with our AMI



7) Launch a new instance from the AMI
8) Login to it using putty and we will see that it has the same web server as our source instance.

- .

ELASTIC LOAD BALANCERS

- ELB are a significant managed service provided by AWS to distribute traffic across multiple instances.
- Autoscaling is a feature which helps us to scale up our infra depending upon the load. We can configure thresholds to add up more ec2 instances in case the CPU utilization reaches a specific value.
- ELB is a logical thing and not a physical device as we used to have in traditional data centres.
- ELB is public facing and represent a single point of contact for multiple EC2 instances launched in the backend.
- ELB don't have an IP address, it just have an FQDN (Fully qualified domain name)
- Instances launched are spread over multiple AZ, hence providing high availability.

➔ AWS gives us three different ELBs to choose from.

Classic ELB:

1) This is the simplest one and just distributes traffic evenly between different instances
2) There are no granular routing rules to route traffic depending upon the contents. It doesn't inspect the nature of the request, just forwards it to EC2 instances.
3) Supports TCP, SSL, HTTP, HTTPS
4) Works on the transport layer of OSI (The layer 4)

Application load Balancer (ALB):

1) ALB is an advancement to Classic ELB
2) Works on Layer 7
3) ALB balance the traffic across instances using content-based rules.

- **Host based Rules:** Route traffic based on the host field of the HTTP header.
- **Path based Rules:** Route traffic based on the URL path field of the HTTP header

4) Can balance traffic to multiple ports.
5) An ALB supports ECS, EKS, HTTPS, HTTP/2, sticky sessions and Web application firewalls.

NOTE: STICKY SESSIONS: By default, a Classic Load Balancer routes each request independently to the registered instance with the smallest load. However, you can use the sticky session feature (also known as session affinity), which enables the load balancer to bind a user's session to a specific instance. This ensures that all requests from the user during the session are sent to the same instance.

The key to managing sticky sessions is to determine how long your load balancer should consistently route the user's request to the same instance. If your application has its own session cookie, then you can configure Elastic Load Balancing so that the session cookie follows the duration specified by the application's session cookie. If your application does not have its own session cookie, then you can configure Elastic Load Balancing to create a session cookie by specifying your own stickiness duration.

Network Load Balancer:

1) Simple layer 4 ELB
2) It's a plain TCP pass through
3) They are useful when there are sudden fluctuations in the load i.e. load goes up and down in matter of seconds eg. In case of Amazon big billion-day sale. In this scenario the Classic ELB and ALB are not efficient as it takes number of seconds or even minutes to launch new instance. In this case NLB is helpful.
4) NLB does not launch new instances, it is purely software-based load balancer which routes traffic amongst available instances efficiently.
5) Supports static IPs and also elastic IP. It will have one static IP /AZ.

NOTE: SSL TERMINATION: This is a feature supported by application load balancer, where data when travels via HTTPS protocol, the HTTPS gets terminated at the load balancer and after that traffic goes via HTTP. This saves a big amount of CPU cycles.

AUTOSCALING

- ➔ Autoscaling is one of the main features because of which we use autoscaling. It allows us to scale our infra as per our requirement.
- ➔ Scale up (Vertical scaling): This is the process of increasing the resources on our existing system. Like increasing RAM, CPU etc
- ➔ Scale out (Horizontal scaling) : This means increasing the number of instances.
- ➔ Autoscaling allows us to add more instances for our application so that instances don't get saturated and there are more instances to distribute the load. Traffic is distributed amongst available instances by ELB.
- ➔ When the load decreases the instances are terminated and we no longer need to pay for them.
- ➔ All these actions are taken based on the cloud watch metrices, we need to set a policy , e.g. if CPU > 70% , launch more instances.

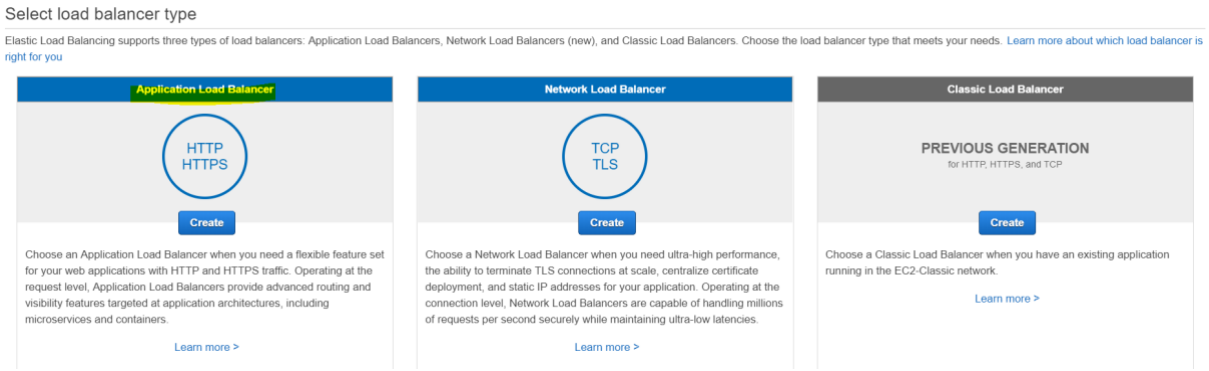
➔ **AUTOSCALING COMPONENETS**:

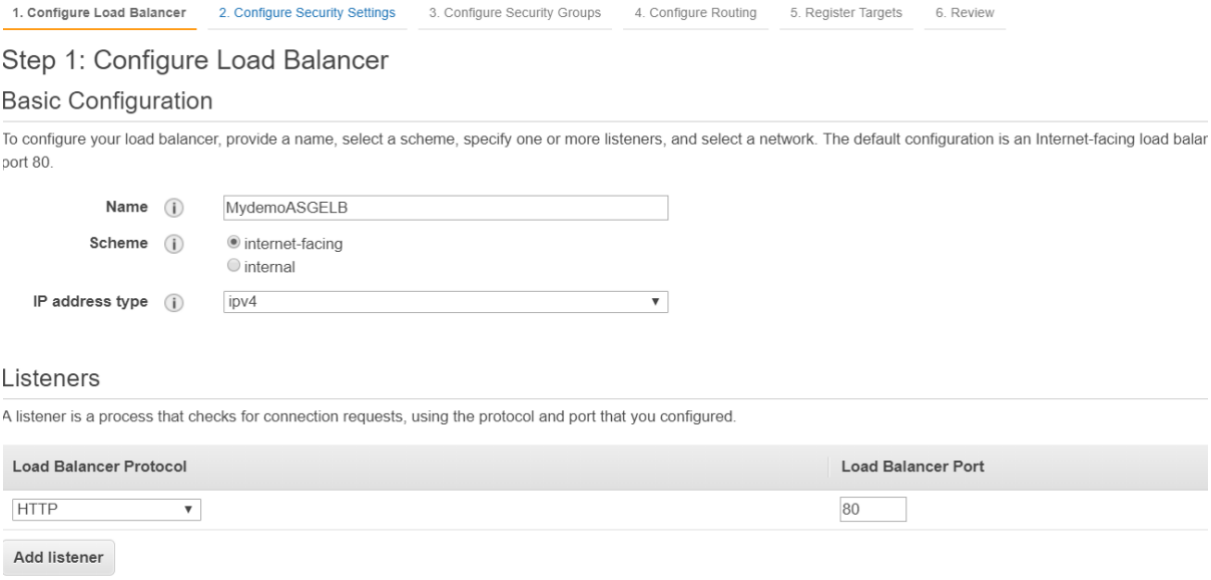
- **LAUNCH CONFIGURATION:** The EC2 template used when we need to provision new instances (Like the AMI, security groups, user-data, storage etc.)
- **AUTOSCALING GROUP:** All the rules and settings that govern when an instance is launched or terminated.
    a) Number of minimum and maximum instances to launch
    b) VPC/ AZ to launch the instance in.
    c) If provisioned instances should receive traffic from ELB
    d) Scaling policies (CloudWatch metrices and threshold that triggers autoscaling)
    e) SNS notifications (To send notification regarding autoscaling event)
- **CLOUDWATCH ALARMS:**
    a) Metrices are selected that indicate load on instances.
    b) Alarms are triggered when metrices exceeds threshold.
    c) Alarms trigger autoscaling policies.
    d)

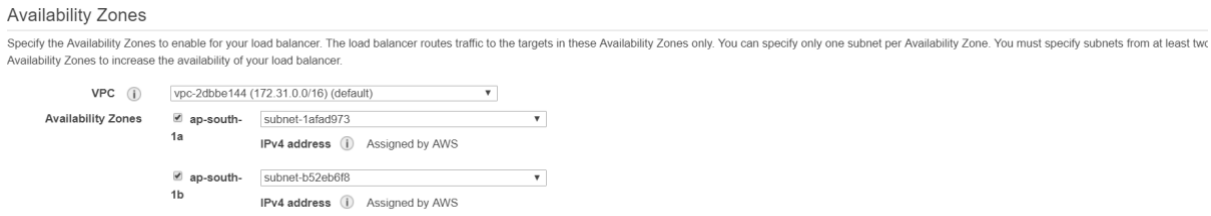
Let’s see how we can configure autoscaling.

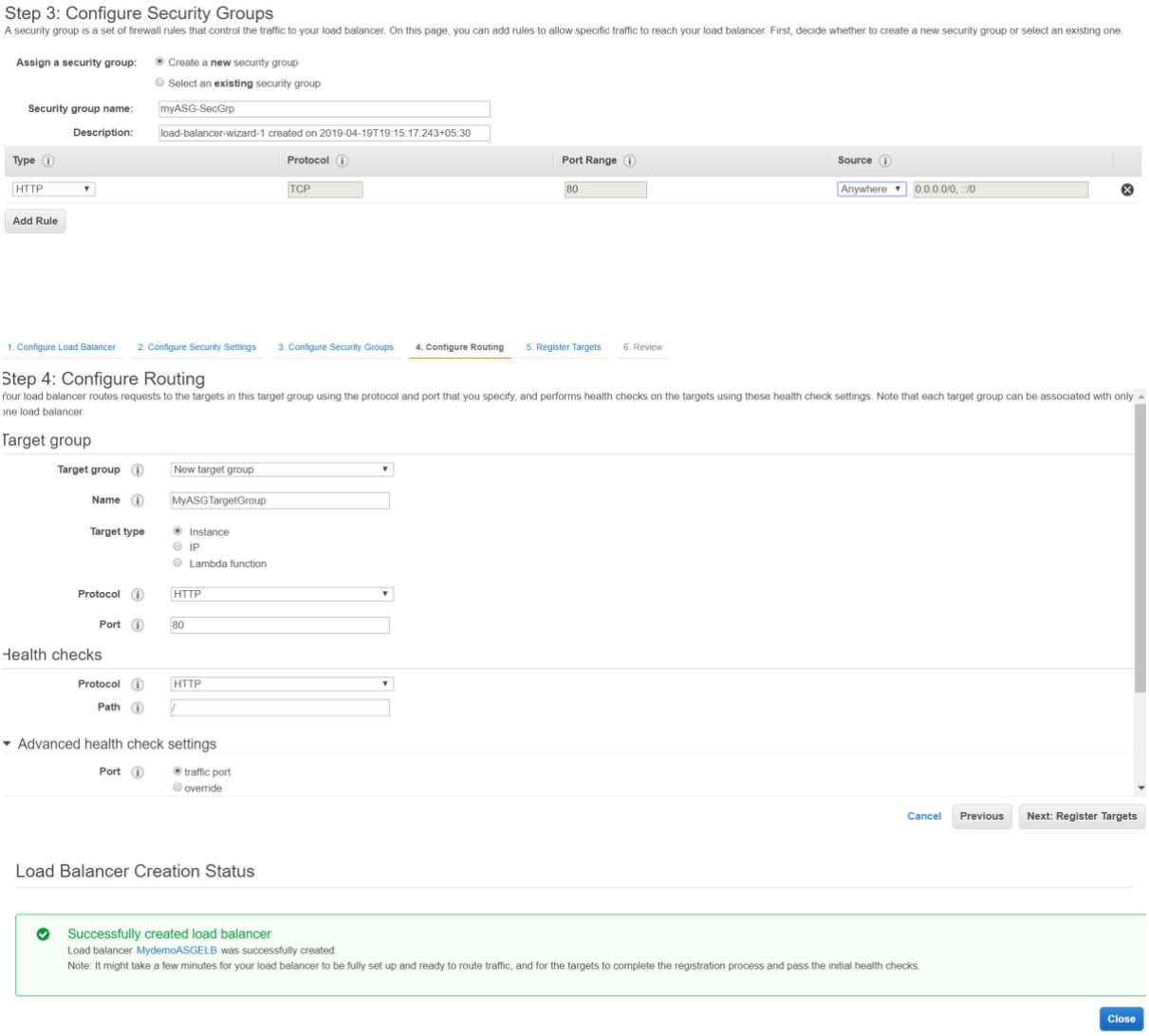
➔ CREATE A LOAD-BALANCER(ALB)

Go to EC2 dashboard-> Create load balancer-> Select ALB



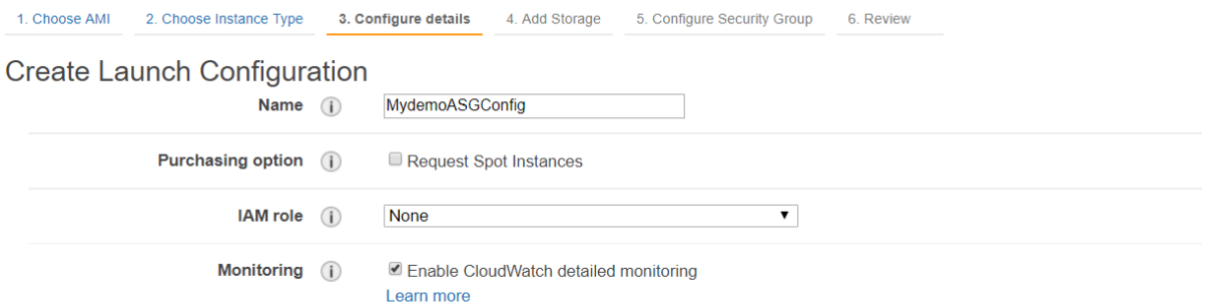


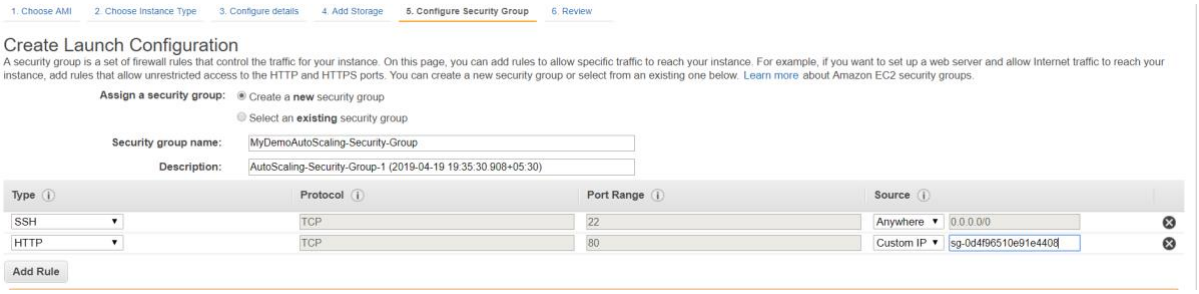




➔ **CONFIGURE LAUNCH CONFIGURATION**

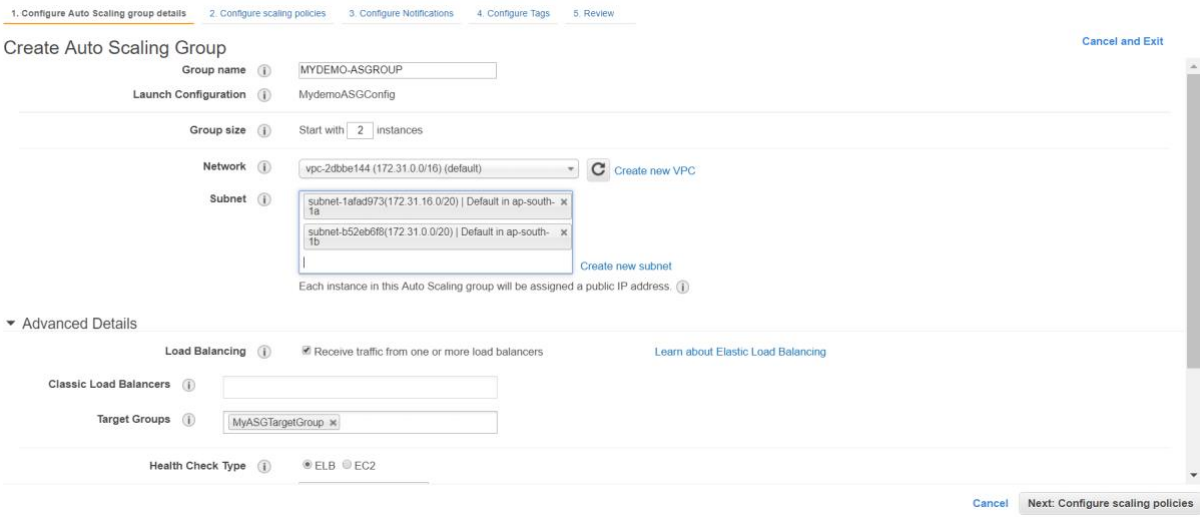
- Create autoscaling group

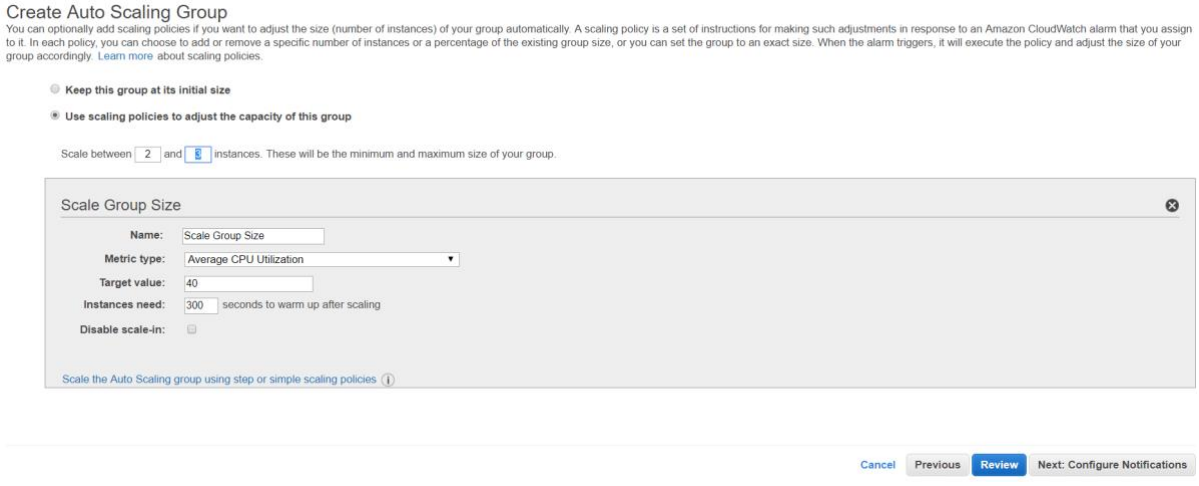


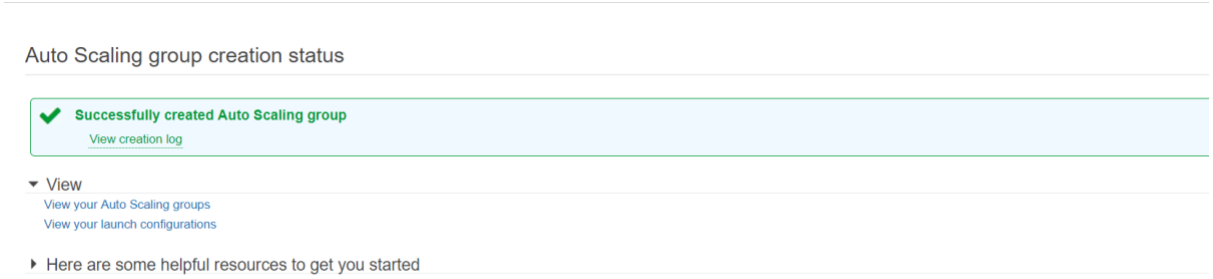


- Here add the security group that we configured for ALB.
- Create the Launch configuration

➔ CREATE AUTOSCALING GROUP





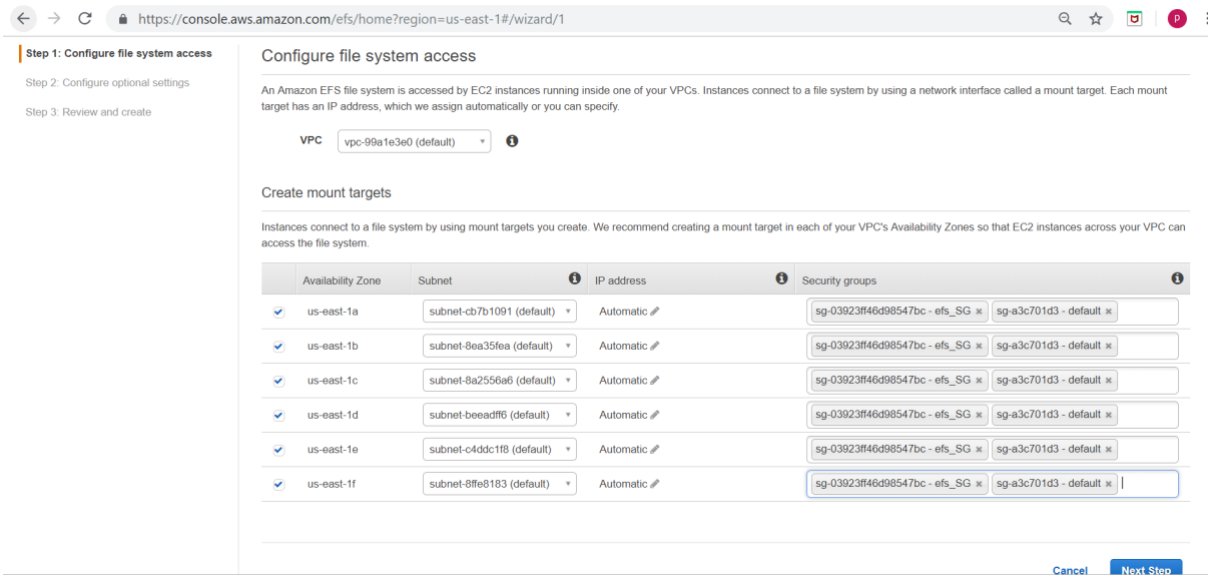

Auto Scaling group creation status
Successfully created Auto Scaling group
View creation log
View
View your Auto Scaling groups
View your launch configurations
Here are some helpful resources to get you started

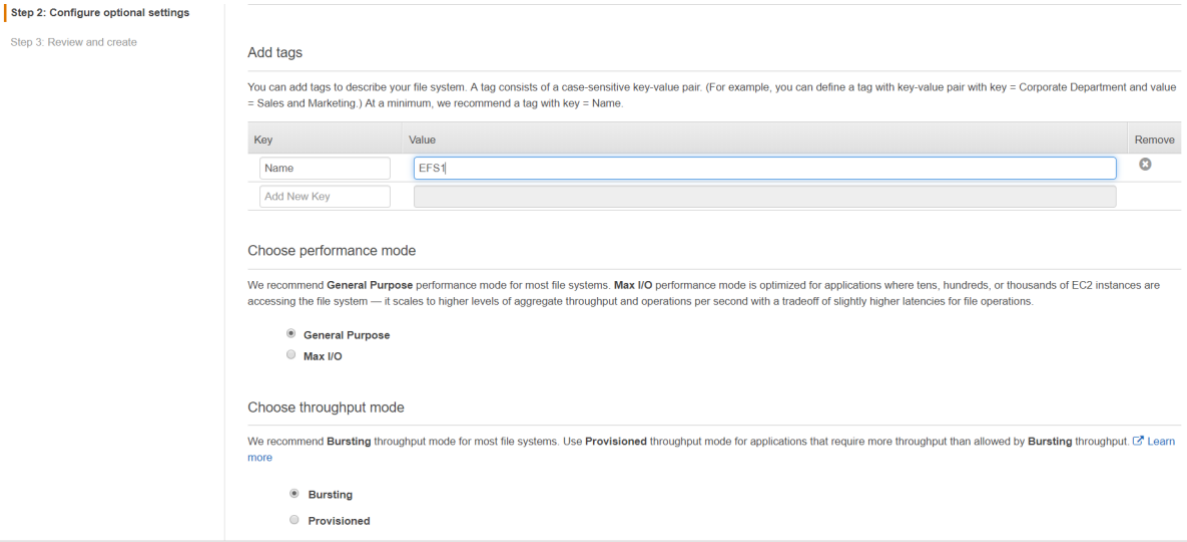
ELASTIC FILE SYSTEM (EFS)

- ➔ Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2. With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files, so your applications have the storage they need, when they need it. Amazon EFS has a simple web services interface that allows you to create and configure file systems quickly and easily. The service manages all the file storage infrastructure for you, meaning that you can avoid the complexity of deploying, patching, and maintaining complex file system configurations.
- ➔ Based on NFS (Network file system)
- ➔ Whenever we will configure it, always open port 2049 in the outbound rules.
- ➔ Multiple Amazon EC2 instances can access an Amazon EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance or server.
- ➔ With Amazon EFS, you pay only for the storage used by your file system and there is no minimum fee or setup cost.

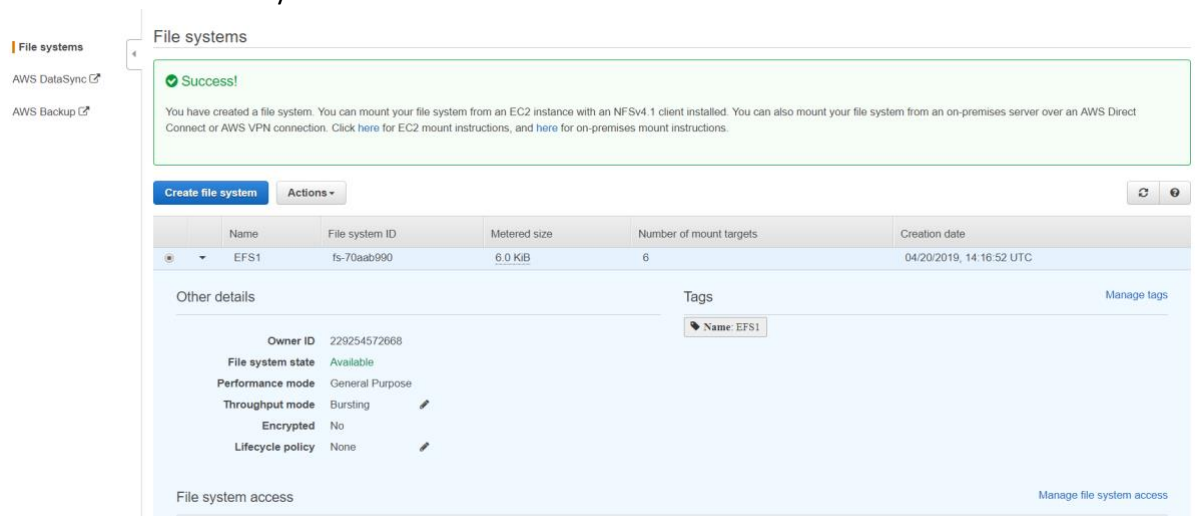
Launch two new EC2 instances (we have launched node1 and node2)

- Go to Services → EFS

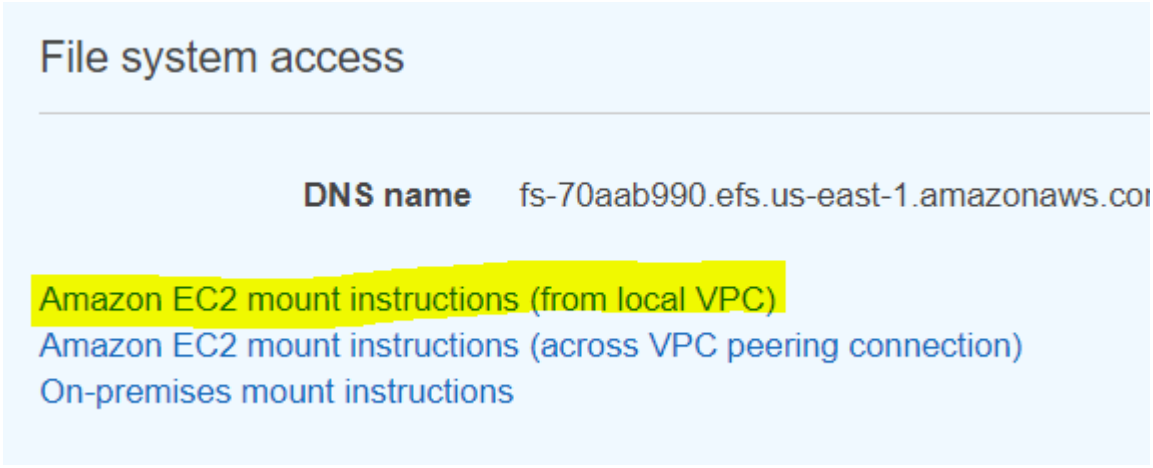




- Review and create file system



- Login to the ec2 instance where you need to mount the file system .
- Install nfs-utils rpm
- Follow the steps to mount the file system, go to EFS service and see the instructions



- In the security group , allow the NFS port 2049 in the outbound rules
- Login to the instance and create a directory (mkdir /efs)
- sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-70aab990.efs.us-east-1.amazonaws.com:/ /efs

AMAZON CLOUDWATCH

- Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, and set alarms. Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files your applications generate. You can use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health. You can use these insights to react and keep your application running smoothly.
- Default monitoring time period is 5 min
- Detailed monitoring time period is 1 min
- By default, metrices are available for

- CPU
- Disk
- Network
- Status Checks

- We can at any time , create our own custom metrices, below we will create one for memory (RAM) utilization.

- On Amazon Linux 2 , install the following RPMs

  yum install -y perl-Switch perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https perl-Digest-SHA.x86_64

- The following steps show you how to download, uncompress, and configure the CloudWatch Monitoring Scripts on an EC2 Linux instance.

- curl https://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.2.zip -O

  yum install unzip
  [root@ip-172-31-30-71 ~]# unzip CloudWatchMonitoringScripts-1.2.2.zip && rm CloudWatchMonitoringScripts-1.2.2.zip && cd aws-scripts-mon
  Archive: CloudWatchMonitoringScripts-1.2.2.zip
  extracting: aws-scripts-mon/awscreds.template
  inflating: aws-scripts-mon/AwsSignatureV4.pm
  inflating: aws-scripts-mon/CloudWatchClient.pm
  inflating: aws-scripts-mon/LICENSE.txt
  inflating: aws-scripts-mon/mon-get-instance-stats.pl
  inflating: aws-scripts-mon/mon-put-instance-data.pl
  inflating: aws-scripts-mon/NOTICE.txt
  rm: remove regular file 'CloudWatchMonitoringScripts-1.2.2.zip'? y

➔ Run the verification test

[root@ip-172-31-30-71 aws-scripts-mon]# ./mon-put-instance-data.pl --mem-util --verify --verbose

MemoryUtilization: 16.1094200924599 (Percent)

No credential methods are specified. Trying default IAM role.

Using IAM role <Admin_Access_Role>

Endpoint: https://monitoring.ap-south-1.amazonaws.com

Payload: {"MetricData":[{"Timestamp":1555774493,"Dimensions":[{"Value":"i-07ebb578ed1de5e44","Name":"InstanceId"}],"Value":16.1094200924599,"Unit":"Percent","MetricName":"MemoryUtilization"}],"Namespace":"System/Linux","__type":"com.amazonaws.cloudwatch.v2010_08_01#PutMetricDataInput"}

Verification completed successfully. No actual metrics sent to CloudWatch.

➔ Collect metrics and send to Cloudwatch

./mon-put-instance-data.pl --mem-used-incl-cache-buff --mem-util --mem-used --mem-avail

./mon-put-instance-data.pl --mem-used-incl-cache-buff --mem-util --mem-used --mem-avail

Successfully reported metrics to CloudWatch. Reference Id: 33db29ea-6382-11e9-99c8-6930459d8716

➔ Set the cron

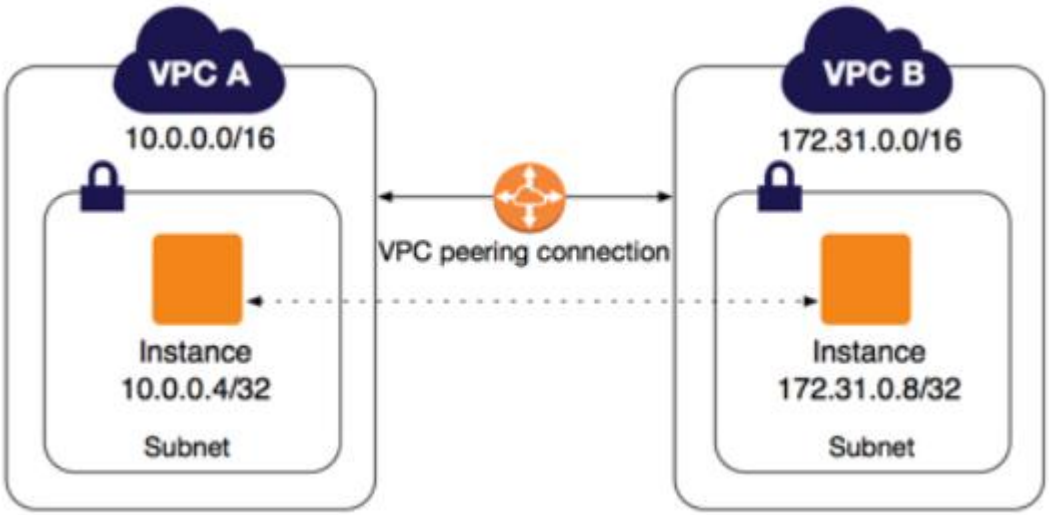
*/5 * * * * ~/aws-scripts-mon/mon-put-instance-data.pl --mem-used-incl-cache-buff --mem-util --disk-space-util --disk-path=/ --from-cron

We are executing the command every 5 minutes to collect the metrices.

VPC (VIRTUAL PRIVATE CLOUD)

- ➔ VPC is a logical datacentre
- ➔ Can span multiple AZ
- ➔ Amazon Virtual Private Cloud (Amazon VPC) enables us to launch AWS resources into a virtual network that we have defined. This virtual network closely resembles a traditional network that we would operate in our own data center, with the benefits of using the scalable infrastructure of AWS.
- ➔ We have complete control over this, including section of our IP range .subnets, routing and network gateways.
- ➔ We can make our network as public facing or deploy it in a private network isolated from the outside world.
- ➔ We can also create a hardware VPN connection between our corporate data centre and VPC
- ➔ There are three ranges of private IP address that we can use
  - 10.0.0.0-10.255.255.255 (10/8)
  - 172.16.0.0-172.31.255.255(172.16/12)
  - 192.168.0.0-192.168.255.255(192.168/16)
- ➔ When we use VPC, the maximum address size that amazon gives us is /16.
  - Security Groups and NACL can span multiple subnets.
  - One subnet = 1 AZ
  - 1 Internet gateway/VPC
- ➔ **DEFAULT VS CUSTOM VPC**
  - Default VPC by default will have a route out to the internet.
  - In simple terms we can say that, instances launched in default VPC will have both private as well as public IP addresses.
  - We can delete a default VPC but then we need to specify other VPC or else we will not be able to launch instances.
  - Each region has its own default VPC
  - Security groups cant span VPCs

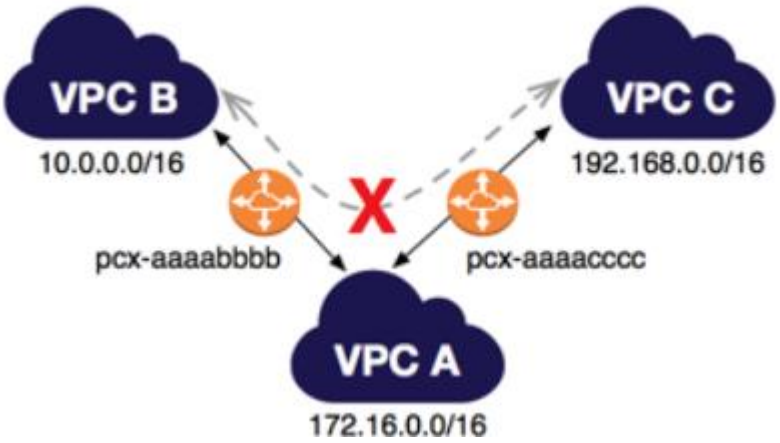
➔ **VPC PEERING**



- A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.
- Instances in either VPC can communicate with each other as if they are within the same network.
- We can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.
- The VPCs can be in different regions (also known as an inter-region VPC peering connection)
- AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck
- A VPC peering connection helps you to facilitate the transfer of data. For example, if you have more than one AWS account, you can peer the VPCs across those accounts to create a file sharing network. You can also use a VPC peering connection to allow other VPCs to access resources you have in one of your VPCs.

➔ **LIMITATIONS OF VPC PEERING**

- Peering is always done in star configuration, transitive peering is not allowed.

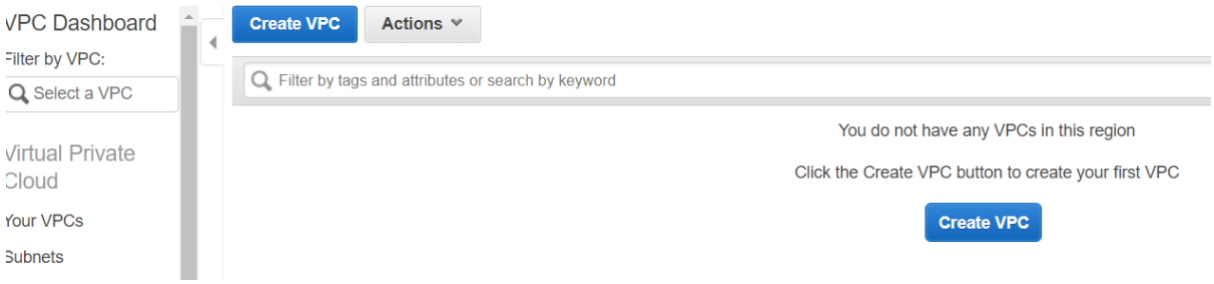


- Peering cannot be done between connections with overlapping CIDR blocks.

- Edge to edge routing through a gateway or private connection.
- A VPN connection or an AWS Direct Connect connection to a corporate network
- An internet connection through an internet gateway
- An internet connection in a private subnet through a NAT device

➔ CREATE A CUSTOM VPC

- Go to services -> VPC



Create VPC

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You must specify an IPv4 address range for your VPC. Specify the IPv4 address range as a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. You cannot specify an IPv4 CIDR block larger than /16. You can optionally associate an Amazon-provided IPv6 CIDR block with the VPC.

Name tag MysampleVPC

IPv4 CIDR block* 10.0.0.0/16

IPv6 CIDR block No IPv6 CIDR Block / Amazon provided IPv6 CIDR block

Tenancy Default

* Required

Cancel | Create

- When we create a VPC , by default below things are created as well
- Route table
- NACL
- Security group
- In order to use our VPC , we are now going to create our subnets

Subnets > Create subnet

Create subnet

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /28 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

Name tag MySampleSubnet-10.0.1.0-aps1a

VPC* vpc-05f41294df8107e27

VPC CIDRs

CIDR	Status	Status Reason
10.0.0.0/16	associated	
2406:da1a:5b:bc00::/56	associated	

Availability Zone ap-south-1a

IPv4 CIDR block* 10.0.1.0/24

IPv6 CIDR block Don't Assign Ipv6

* Required

Cancel | Create

- Now we are going to create one more subnet

Create subnet

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /28 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

Name tag MySampleSubnet-10.0.2.0-aps1b

VPC* vpc-05f41294df8107e27

VPC CIDRs

CIDR	Status	Status Reason
10.0.0.0/16	associated	
2406:da1a:5b:bc00::/56	associated	

Availability Zone ap-south-1b

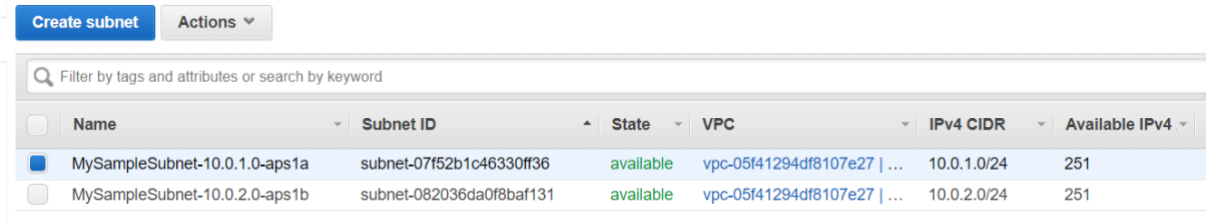
IPv4 CIDR block* 10.0.2.0/24

IPv6 CIDR block Don't Assign Ipv6

* Required

Cancel | Create

- So below are our subnets that we have created



- To check the number of available IP addresses , visit the website

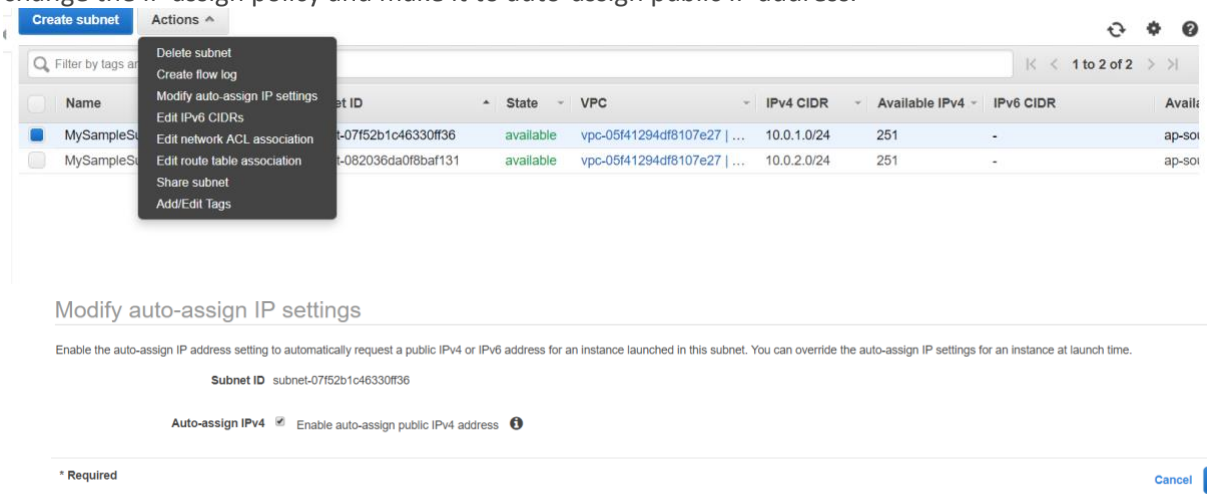
  Cidr.xyz

- It should be noted that :

  The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use, and cannot be assigned to an instance. For example, in a subnet with CIDR block 10.0.0.0/24, the following five IP addresses are reserved:

- 10.0.0.0: Network address.
- 10.0.0.1: Reserved by AWS for the VPC router.
- 10.0.0.2: Reserved by AWS. The IP address of the DNS server is always the base of the VPC network range plus two; however, we also reserve the base of each subnet range plus two. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR..
- 10.0.0.3: Reserved by AWS for future use.
- 10.0.0.255: Network broadcast address. We do not support broadcast in a VPC, therefore we reserve this address.

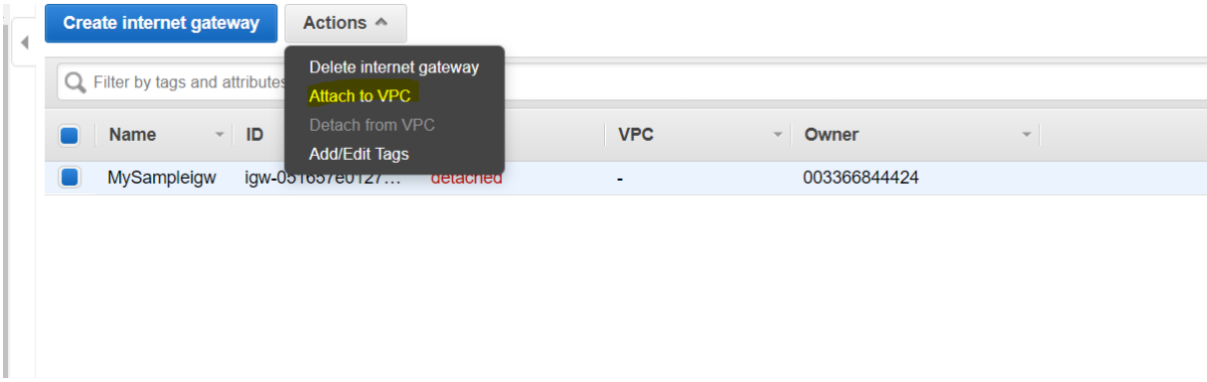
➔ We are going to make one subnet (10.0.1.0/24) as public facing , for this we need to change the IP assign policy and make it to auto-assign public IP address.

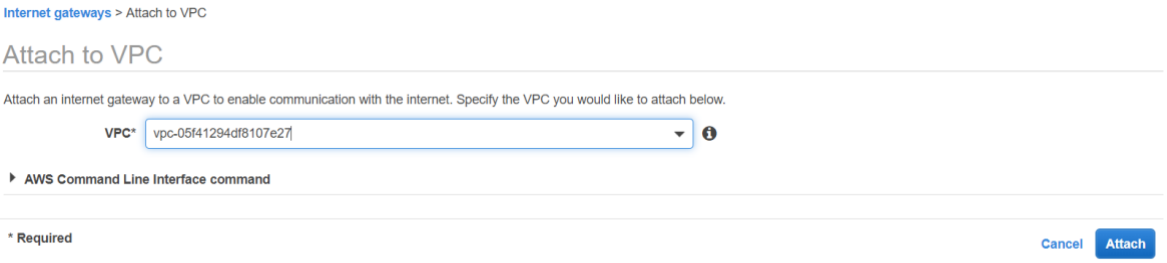


- Now to make our subnet reach out to the internet, we need to have an internet gateway associated with our subnet

*** One subnet always can have only one VPC

- We are now going to attach it to our VPC



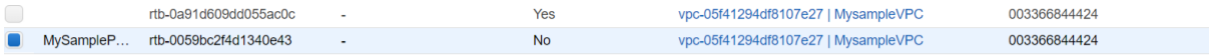


- Now we will configure the route table to add a route out the internet.

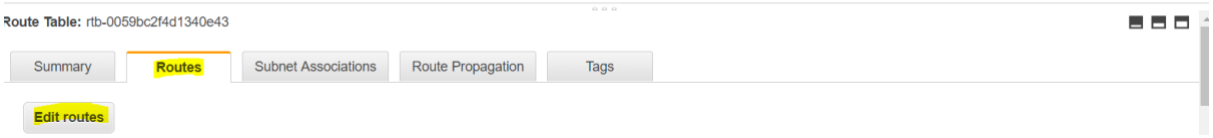
➔ We will have a default main route table created as a part of VPC creation , but if we make it public then , other instances launched within that subnet , will be accessible to public ; and we don't want that in any case as it may lead to security breach.

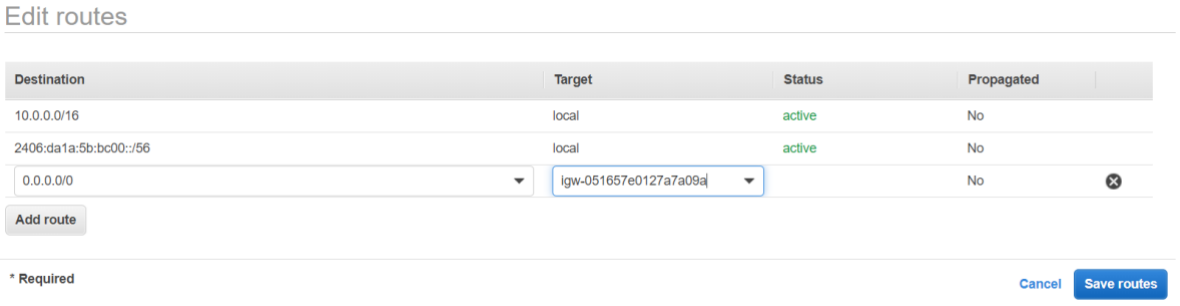
- For this we are going to create one more route table .

- Now we will edit our public route and create a route out to the internet

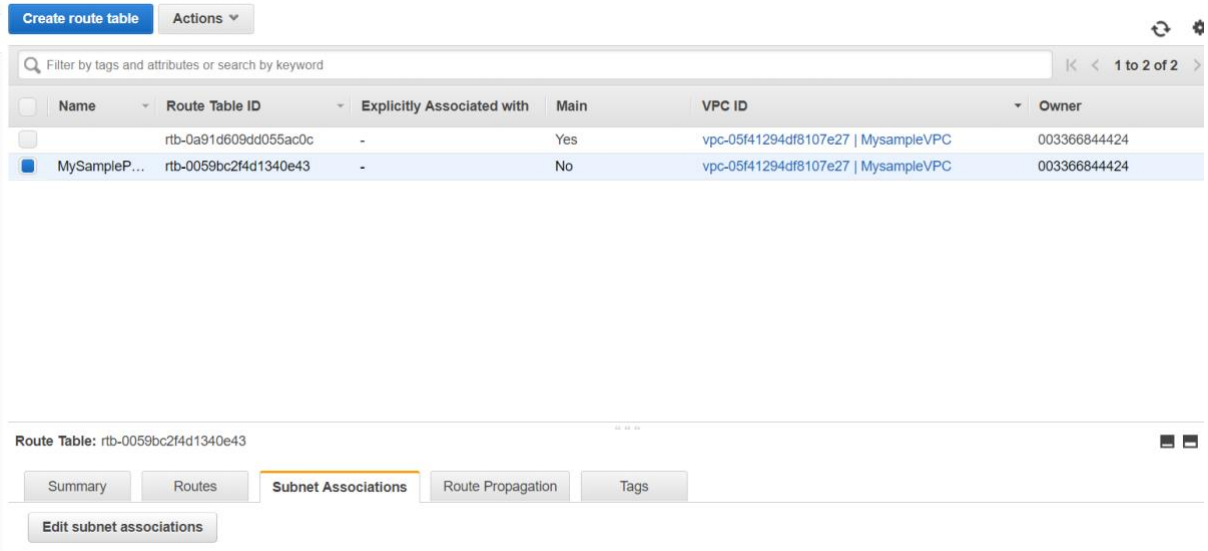


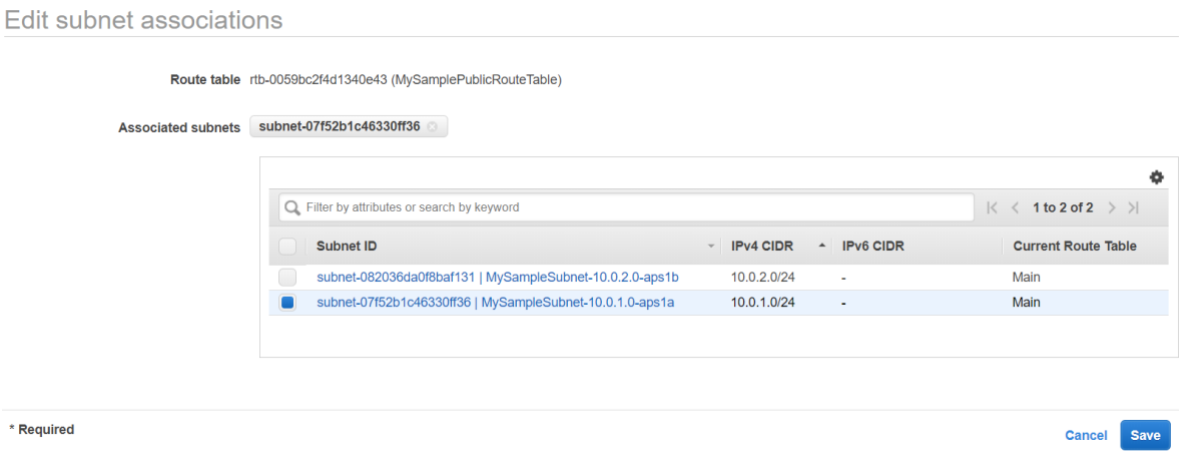
		rtb-0a91d609dd055ac0c	-	Yes	vpc-05f41294df8107e27 | MysampleVPC	003366844424
	MySampleP…	rtb-0059bc2f4d1340e43	-	No	vpc-05f41294df8107e27 | MysampleVPC	003366844424



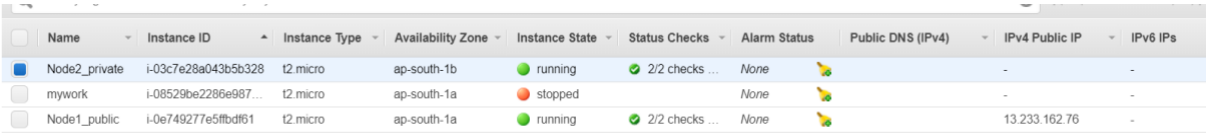


- Now we are going to associate our subnet with the route table.

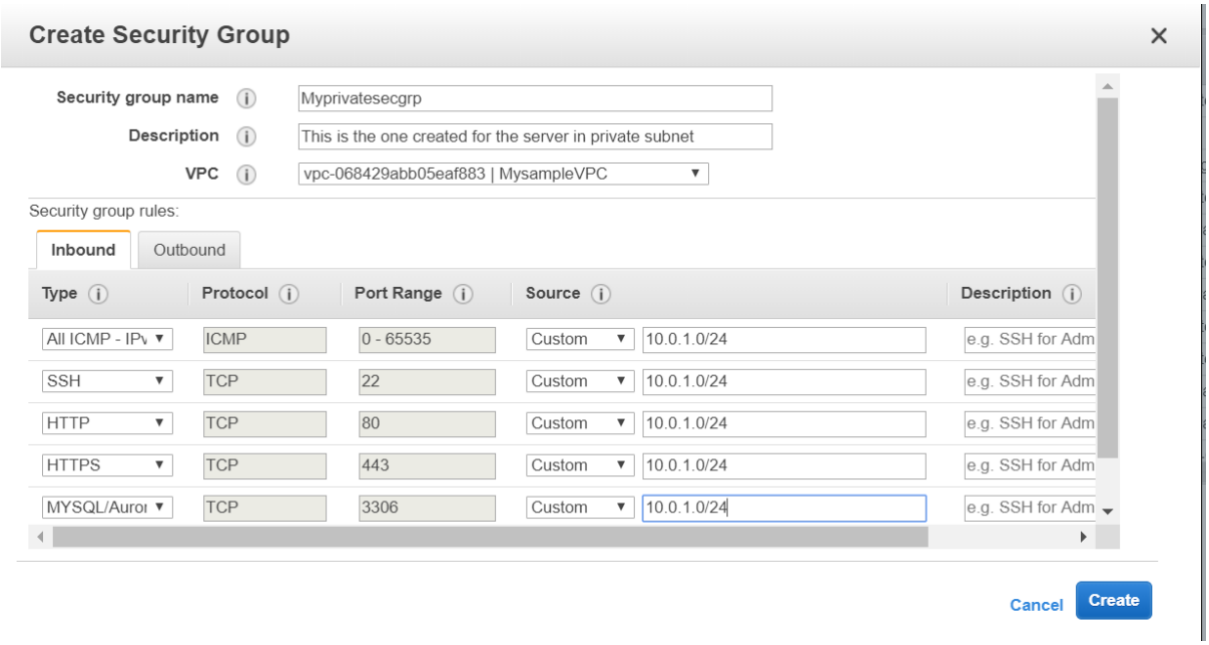




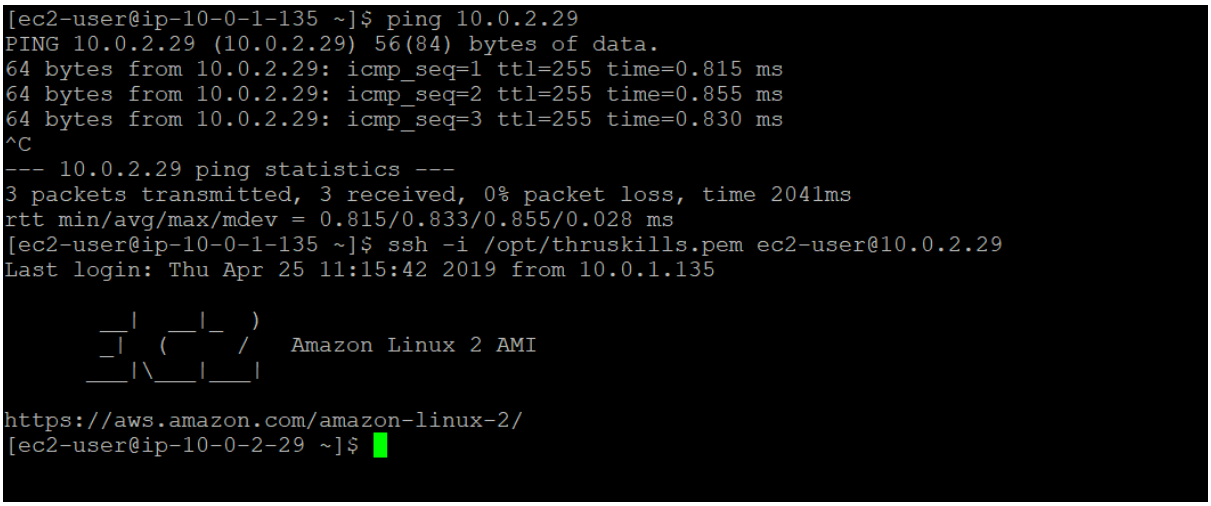
- ➔ Now we will launch two instances, one in each of our subnet , the one with in the public subnet will have a public IP and we will be able to ssh in to it. Here create an attach a new security group , allow ports 80 and 22 in inbound rules.
- ➔ For the other instance in our private subnet , there will be no public IP , here attach default security group
- ➔ We will see that there is no way as of now to ssh in to the server with the private IP address.



- ➔ We will now find a way to ssh in to the server with the private IP address only.
  Now we are going to create a new security group and attach to our node2 , the one in private subnet.We will allow traffic from our public subnet to node2



➔ Now we will see that we will be able to ping and ssh node2 from node1



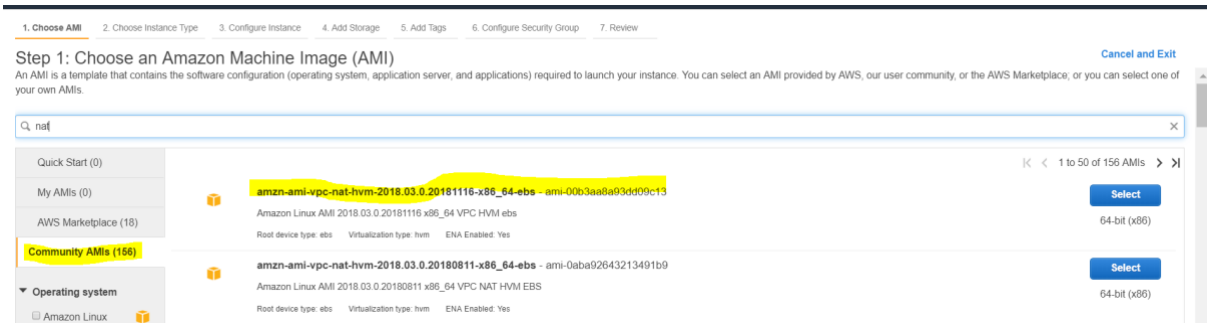
- Login to node1 from ec2-user
- Copy the keypair and save it in /opt
- Provide the necessary permissions
- chmod 400 keypair
- [ec2-user@ip-10-0-1-135 ~]$ chmod 400 /opt/thruskills.pem
- [ec2-user@ip-10-0-1-135 ~]$ ssh -i /opt/thruskills.pem ec2-user@10.0.2.29

➔ Note that we are only able to ssh to node2 , but it is not connected to internet yet.

➔ Next question is how to connect server in private subnet to the internet ? For this we have NAT instances and NAT gateways .

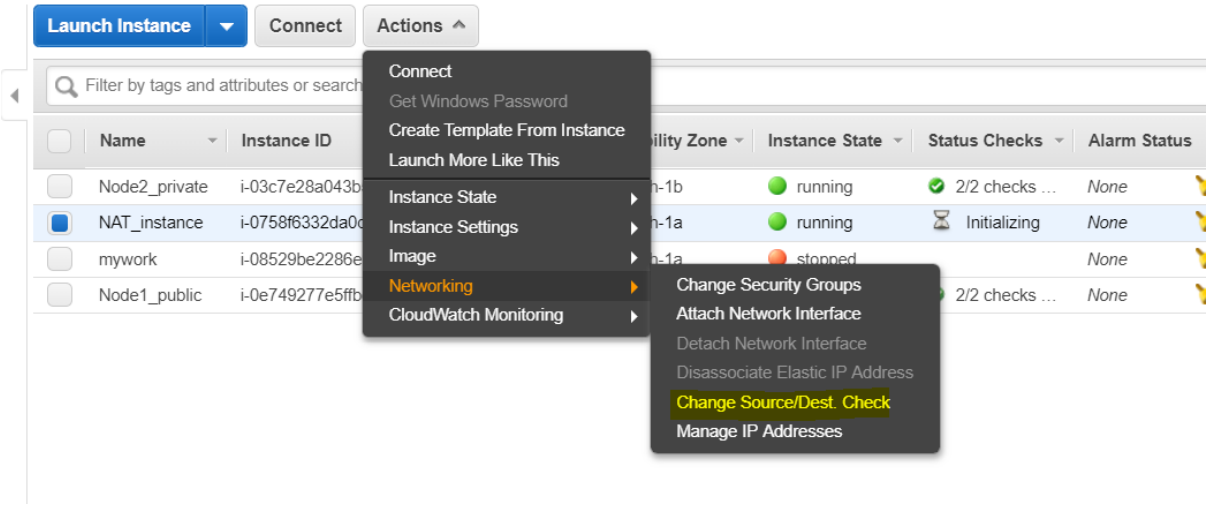
NAT INSTANCES AND NAT GATEWAYS

- NAT instances are just normal EC2 instances whereas NAT gateways are highly available virtual devices from AWS.
- They are not highly available by default, we can do it manually by placing them behind load balancers.
- The amount of traffic NAT instance can take , depends on the instance type.
- NAT gateways are about to be decommissioned but still for the knowledge sake ,we are going to configure one.
- We will launch a NAT instance from AWS market place as an AMI is already available Will attach it to our VPC and launch it in public subnet.

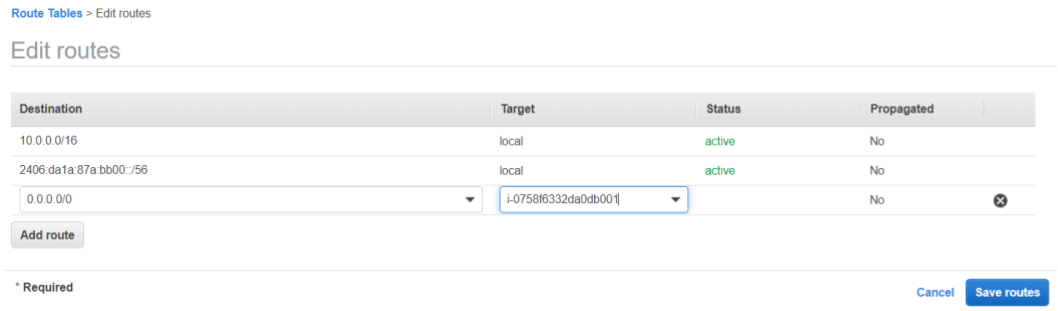


- Allow ports 22 and 80 in the outbound rules.

➔ **NOTE**: Each EC2 instance performs source/destination checks by default. This means that the instance must be the source or destination of any traffic it sends or receives. However, a NAT instance must be able to send and receive traffic when the source or destination is not itself. Therefore, you must disable source/destination checks on the NAT instance.



➔ Now we will edit our main route table and add a rule which will say that when the destination is public i.e. 0.0.0.0/0 , the traffic should be routed to our NAT instance.



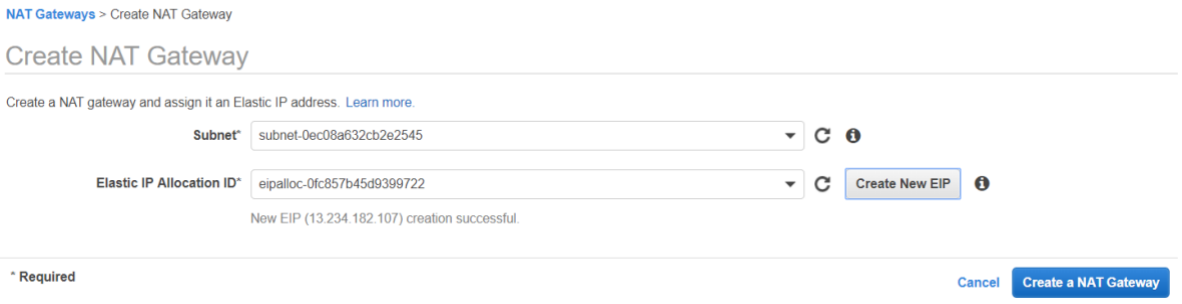
➔ Now we will see that the yum update command will work.

NAT GATEWAYS

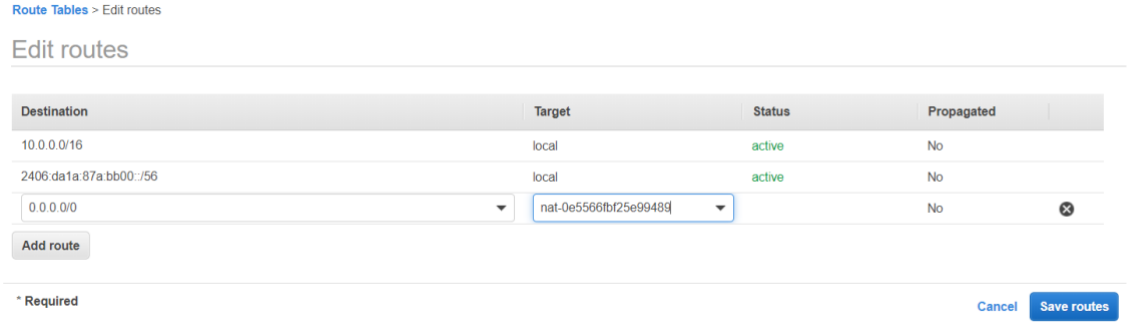
- NAT gateways are much more efficient and highly available way to configure the NAT.
- It is fully managed by AWS
- Now we are going to create a NAT gateway.

➔ Go to VPC->NAT Gateways->Create NAT gateway

➔ Select our public subnet and attach an elastic IP



➔ Now we will edit our main route table and add the route for NAT instance.



- NAT gateways are highly available , for redundancy we can multiple NAT gateways in different AZ. There should be a route for every NAT gateway.
- NAT gateways are not associated with any security group.
- There is no need to disable the source and destination checks as in case of NAT instances.

VPC FLOW LOGS

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs and Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination.

Flow logs can help you with a number of tasks; for example, to troubleshoot why specific traffic is not reaching an instance, which in turn helps you diagnose overly restrictive security group rules. You can also use flow logs as a security tool to monitor the traffic that is reaching your instance.

Logs can be configured at

- VPC level
- Subnet level
- Network interface level

BASTION HOSTS/JUMP SERVER

It is an instance from where we can ssh to all our other instances in private subnet. This is the only server that is exposed to public, we just need to maintain security for bastion host.

AMAZON DIRECT CONNECT

AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard Ethernet fibre-optic cable. One end of the cable is connected to your router, the other to an AWS Direct Connect router. With this connection, you can create *virtual interfaces* directly to public AWS services (for example, to Amazon S3) or to Amazon VPC, bypassing internet service providers in your network path. An AWS Direct Connect location provides access to AWS in the Region with which it is associated.

VPC ENDPOINTS

A 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.

Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.

There are two types of VPC endpoints: *interface endpoints* and *gateway endpoints*.

1) **Interface Endpoint**: An interface endpoint is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported service.
2) **Gateway Endpoints**: A gateway endpoint is a gateway that is a target for a specified route in your route table, used for traffic destined to a supported AWS service.

CLOUD FRONT

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

- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.
- If the content is not in that edge location, CloudFront retrieves it from an origin that you've defined—such as an Amazon S3 bucket, a MediaPackage channel, or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.
- CloudFront speeds up the distribution of your content by routing each user request through the AWS backbone network to the edge location that can best serve your content. Typically, this is a CloudFront edge server that provides the fastest delivery to the viewer. Using the AWS network dramatically reduces the number of networks that your users' requests must pass through, which improves performance. Users get lower latency—the time it takes to load the first byte of the file—and higher data transfer rates.
- You also get increased reliability and availability because copies of your files (also known as *objects)* are now held (or cached) in multiple edge locations around the world.

ROUTE 53

- Its AWS DNS service (Domain Name Server)
- Got its name because DNS service runs on port 53
- DNS resolves IP address to hostname and vice versa
- IPv4 has 32 bit addressing space
- IPv6 is 128 bit addressing space
- Com,edu,net,org are top level domain
- . co.in ,.co.uk ,here .co is a second level domain name.
- A **domain name registrar** is an authority that can assign top level domains directly e.g. Godaddy.com

➔ **EXAMPLE**

- Let us say that a user typed samplewebsite.com in the browser
- Browser doesn't know the IP address so goes to the top level domain server (.com in our case)
- TLD server has the name server record , then the name servers are queried and ultimately give us the SOA(Start of Authority) record and here we are going to have all DNS records .
- DNS records consists of multiple fields , the most fundamental is the "A" record , the address record which have domain name to IP mapping .
- TTL – TTL stands for time to live which is the value in seconds for which the DNS record is cached either on the resolving server or at the user's PC. Default TTL is 48 hours.
- CName – Canonical name are those which can help us resolve one hostname to another
- Alias record are also similar to cname but a cname cant be used for naked or apex domain names.
- Naked domain names are names without www.

■ **SIMPLE ROUTING POLICY**

- We can have one record with multiple IP addresses
- Route 53 returns all values in random order

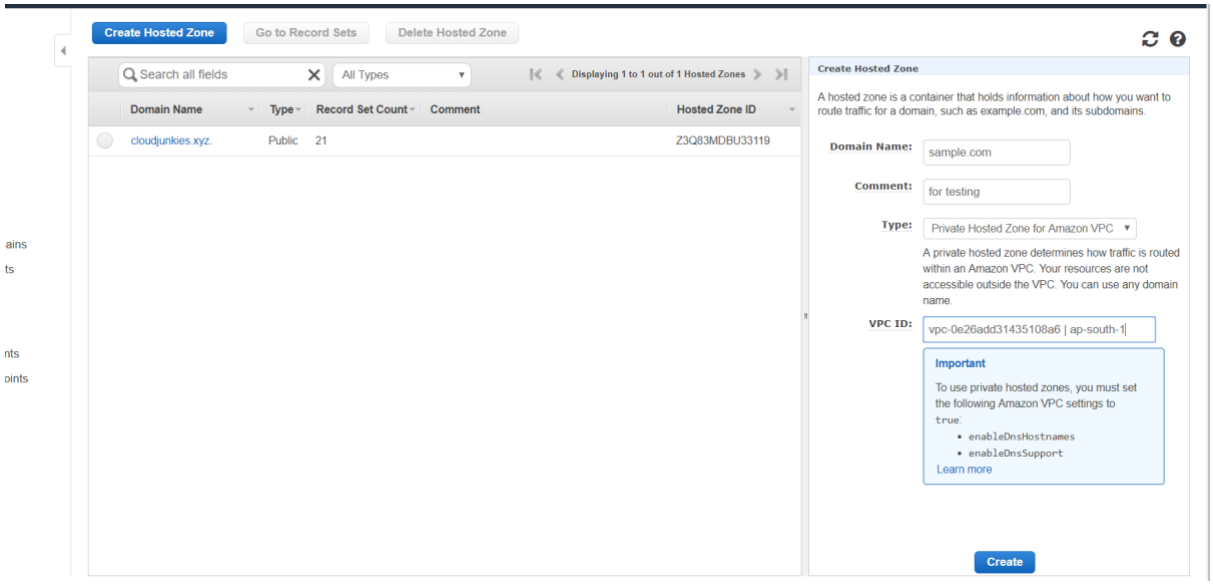
➔ Now we will create a simple routing policy record.

➔ We have an instance with
Public IP: 52.66.196.205
Private IP: 10.0.1.89

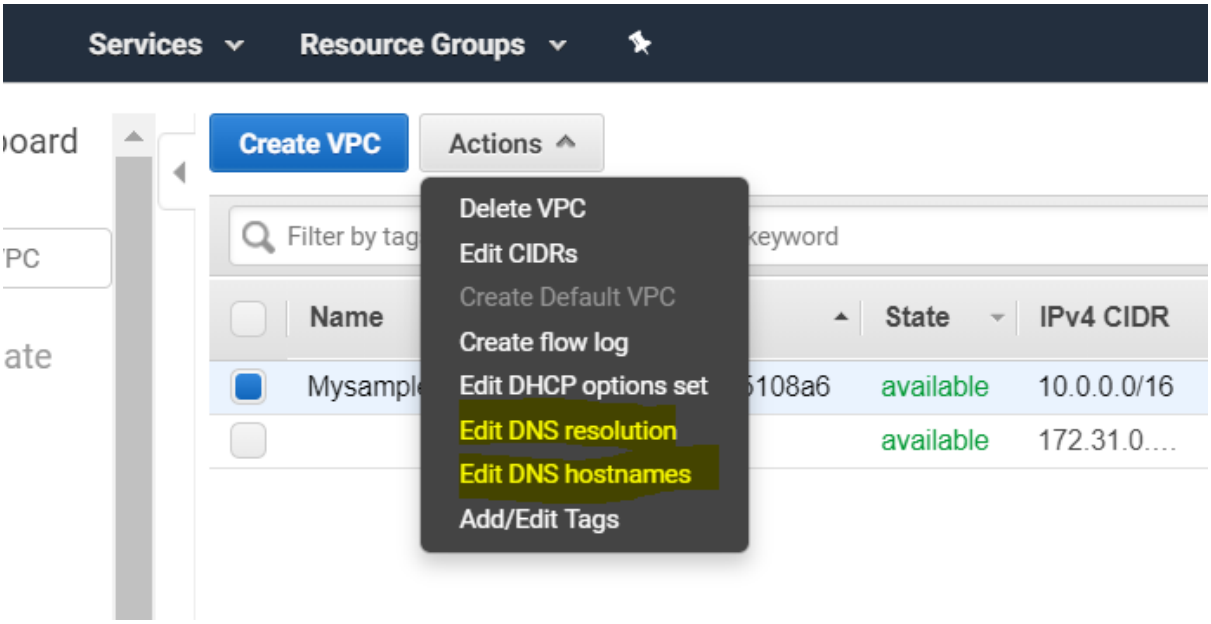
➔ Now we will install httpd
yum install httpd –y
echo “This is a test website” >> /var/www/html
systemctl start httpd
systemctl enable httpd

➔ Now we will test our httpd server
[root@ip-172-31-26-221 html]# curl http://ip-172-31-26-221.ap-south-1.compute.internal
this is a test website

➔ Now we will go to Route53 service and create our own private hosted zone.
Services->Route53->Create Hosted zone



➔ Modify VPC setting, set enableDNSHostname and enable DNSSupport to true.



VPCs > Edit DNS resolution

Edit DNS resolution

VPC ID vpc-0e26add31435108a6

DNS resolution ☑ enable

*** Required**

Edit DNS hostnames

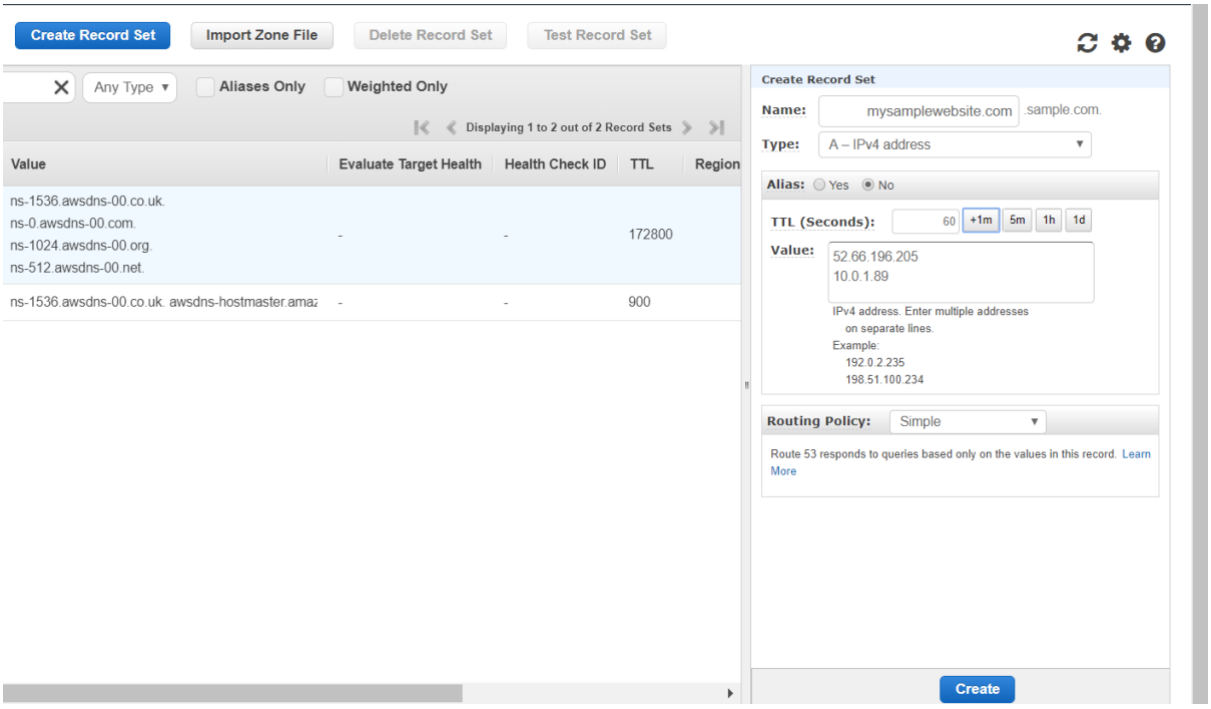
VPC ID vpc-0e26add31435108a6

DNS hostnames ☑ enable

*** Required**

➔ After this go back to our hosted zone (sample.com)

➔ Create a record of “A” type



➔ Create a new hosted zone example.com and select the VPC , the default one

➔ Now we will see that ,the mew domain name provided by us will also resolve to the webpage

[root@ip-172-31-26-221 html]# curl http://mywebsite.example.com

this is a test website

[root@ip-172-31-26-221 html]#

- **Failover routing policy** – Use when you want to configure active-passive failover.
- **Geolocation routing policy** – Use when you want to route traffic based on the location of your users.
- **Geoproximity routing policy** – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another. Based on traffic flows only. We can configure complex routing policies using traffic flows

- **Latency routing policy** – Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency.
- **Multivalue answer routing policy** – Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.
  It is similar to simple routing policy but here additionally we can configure health checks.
- **Weighted routing policy** – Use to route traffic to multiple resources in proportions that you specify.

DATABASES IN AWS

➔ **RELATIONAL DATABASE** : A collection of tables having rows and columns .Just like a simple spreadsheet when we talk in terms of MS excel .

➔ Each table has a primary key that is unique and helps in querying data .

➔ AWS offers following databases

- Microsoft SQL server
- Mysql
- Postgresql
- Oracle
- Mariadb
- Aurora(Amazon proprietry)

➔ RDS has two key features

- **Multi AZ** – For high Availability: Here the RDS instances are deployed in multiple AZ, each instance has a different DNS record, in case one instance goes down , amazon then points to the other DNS record on its own and hence the DB is highly available. We can say that failover is automatic with Multi AZ.
- **Read Replicas** – For high performance: Read replicas are read only copies of the actual DB, it is very much in sync with the primary DB .Read replicas can be really helpful and can be used to distribute read load i.e. they can be configured to read data and hence reduce load on primary DB. We can have 5 read replicas.
- A new DB instance can be created from a read replica, but an automatic failover will not happen as in case of Multi AZ RDS.
- Automatic backups must be enabled to have Read replicas.
- Read replicas cannot be multi AZ but there can be read replicas of multi AZ RDS.

➔ **DATAWAREHOSING**

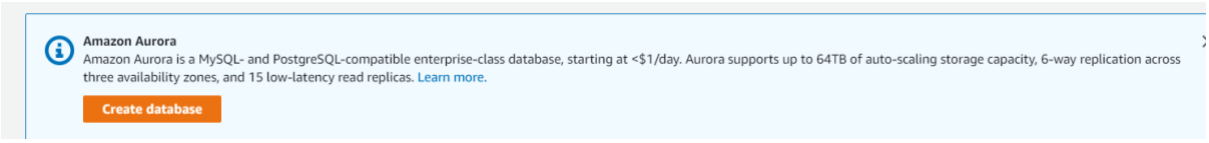
- Relates to storing huge amount of data used for business intelligence tools like cognos,jaspersoft etc used to query data and derive results such as current performance vs targets

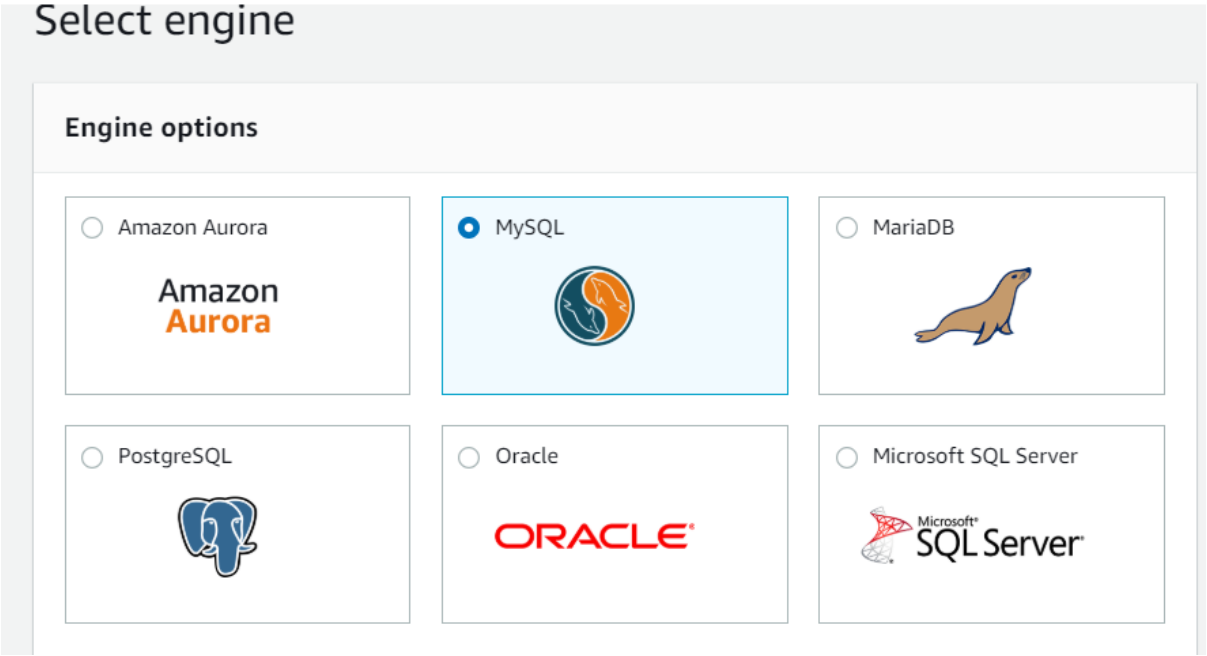
➔ **OLTP VS OLAP**

- Both processes are different in nature,in the way they run. Online transaction processing are related to queries on RDBMS where we get precise results from query eg select * from table_name where ID=2
- OLAP : Online analytics processes analyse a large volume of data e.g lets take a scenario where a sales manager wants to find net profit on the sale of laptop, here there will be variety of data sets like model , customer location, different prices at different location. Etc.

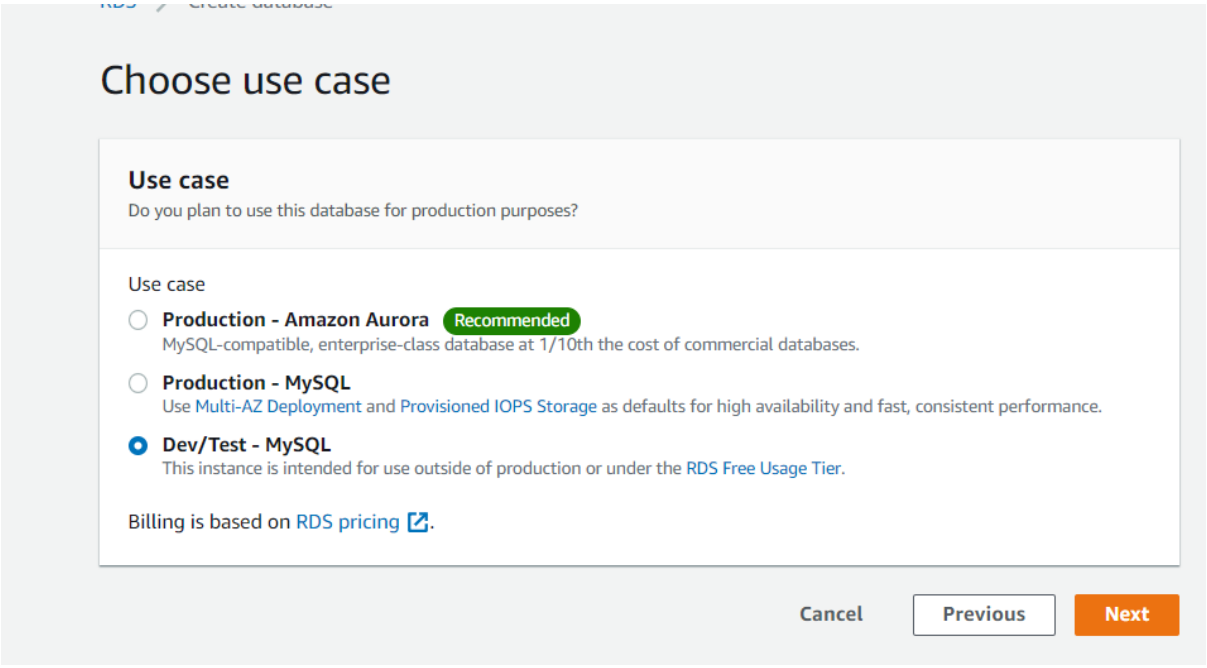
➔ **LAUNCHING AN RDS INSTANCE**

- Go to Services->RDS->Create database





➔



Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the **AWS Simple Monthly Calculator**

DB engine

MySQL Community Edition

License model **Info**

general-public-license

DB engine version **Info**

MySQL 5.7.22

Known Issues/Limitations

Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

Free tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GiB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions here.

Only enable options eligible for RDS Free Usage Tier Info

DB instance class **Info**

db.t2.micro — 1 vCPU, 1 GiB RAM

Multi-AZ deployment **Info**

- Create replica in different zone
  Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
- (●) No

Storage type **Info**

General Purpose (SSD)

Allocated storage

20 GiB

(Minimum: 20 GiB, Maximum: 16384 GiB) Higher allocated storage **may improve** IOPS performance.

Provisioning less than 100 GiB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. Click here for more details.

Estimated monthly costs

DB Instance	17.52 USD
Storage	2.62 USD
Total	**20.14 USD**

DB Instance	17.52 USD
Storage	2.62 USD
Total	**20.14 USD**

Billing estimate is based on on-demand usage as described in Amazon RDS Pricing. Estimate does not include costs for backup storage, IOs (if applicable), or data transfer.

Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator

Settings

DB instance identifier **Info**

Specify a name that is unique for all DB instances owned by your AWS account in the current region.

mytestmysqldb

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Must contain from 1 to 63 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.

Master username **Info**

Specify an alphanumeric string that defines the login ID for the master user.

piyush

Master Username must start with a letter. Must contain 1 to 16 alphanumeric characters.

Master password **Info**

••••••••

Master Password must be at least eight characters long, as in "mypassword". Can be any printable ASCII character except "/", """, or "@".

Confirm password **Info**

••••••••

Cancel Previous Next

RDS > Create database

Configure advanced settings

Network & Security

Virtual Private Cloud (VPC) **Info**
VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-2dbbe144)

Only VPCs with a corresponding DB subnet group are listed.

Subnet group **Info**
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default

Public accessibility **Info**

- Yes
  EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances. You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.
- No
  DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

Availability zone **Info**

ap-south-1a

VPC security groups
Security groups have rules authorizing connections from all the EC2 instances and devices that need to access the DB instance.

- Create new VPC security group
- Choose existing VPC security groups

Database options

Database name **Info**

piyush

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Port **Info**

TCP/IP port the DB instance will use for application connections.

3306

DB parameter group **Info**

default.mysql5.7

Option group **Info**

default:mysql-5-7

IAM DB authentication **Info**

- ○ Enable IAM DB authentication
  Manage your database user credentials through AWS IAM users and roles.
- ◉ Disable

Encryption

Encryption

- ○ Enable encryption Learn more
  Select to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service(KMS) console.
- ◉ Disable encryption

Backup

⚠ Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail here.

Backup retention period **Info**
Select the number of days that Amazon RDS should retain automatic backups of this DB instance.

0 days

ⓘ A backup retention period of zero days will disable automated backups for this DB Instance.

Backup window **Info**

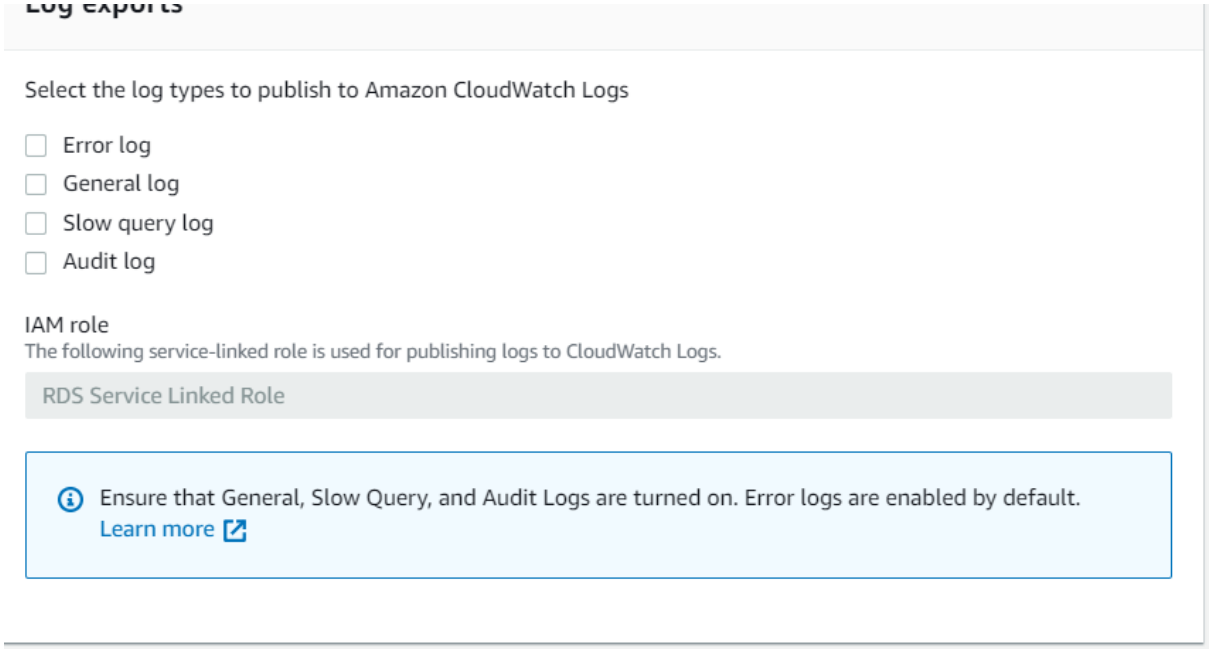
- ○ Select window
- ○ No preference

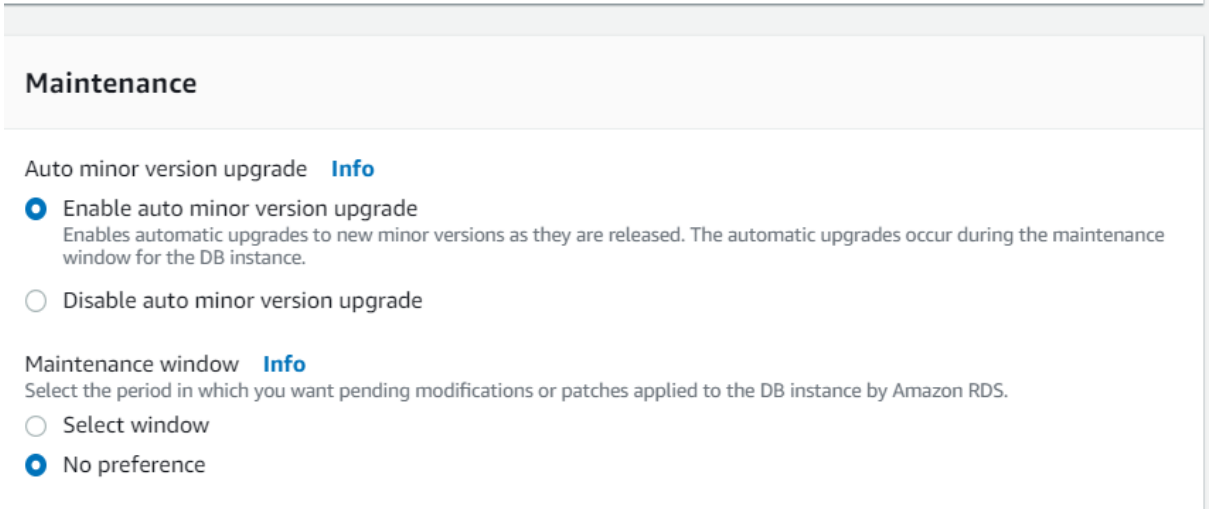
☑ Copy tags to snapshots

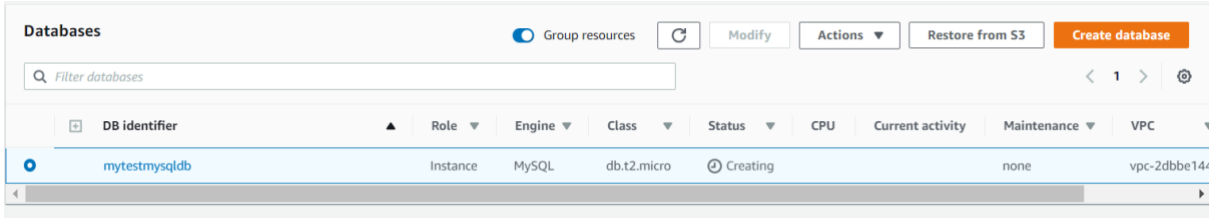
Monitoring

Enhanced monitoring

- ○ Enable enhanced monitoring
  Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU.
- ◉ Disable enhanced monitoring







➔ RDS is fully managed service with no access to EC2 instance. We have third party tools to access the databases.

➔ **AUTOMATIC BACKUPS** : Allows us to restore DB at any point in time within a retention period which can be between 1 and 35 days. Default is 7 days

- Will take a full daily snapshots and will also store transaction logs throughout the day.
- Automatic backups are enabled by default , backup snapshots are stored in S3 buckets and we get a free space in S3 equal to the size of DB.

- Automatic backups may put load on DB so are taken when the load is low may be in non-production hours.
- These are deleted when original RDS are deleted.

➔ **DB SNAPSHOTS**: These are taken manually when required and data cannot be restored point in time.

- These will persist even after RDS instance is deleted.

NOTE: DB instance restored from any of the type of backup will be a new DB with a new endpoint.

NOTE: To enable read replica, automatic snapshots should be enabled

➔ **DATA ENCRYPTION**: Data encryption at rest is taken care by Amazon KMS.

- Encryption of existing database is currently not supported.
- To encrypt existing DB , create its snapshot , create a new DB from snapshot and encrypt it.

➔ When a RDS is deleted , a final snapshot is taken by default.

➔ RDS uses InnoDB storage engine

➔ **InnoDB vs MYISAM ** NOT PART OF AWS EXAM****

The most commonly used storage engine in MySQL are MyISAM and InnoDB.

With these storage engine there are some advantages and disadvantages according to application needs.

As you all know, the default storage engine chosen by MySQL database is MyISAM.

The main difference between MyISAM and INNODB are :

- MyISAM does not support transactions by tables while InnoDB supports.
- There are no possibility of row-level locking, relational integrity in MyISAM but with InnoDB this is possible. MyISAM has table-level locking.
- InnoDB does not support FULLTEXT index while MyISAM supports.
- Performance speed of MyISAM table is much higher as compared with tables in InnoDB.
- InnoDB is better option while you are dealing with larger database because it supports transactions, volume while MyISAM is suitable for small project.
- As InnoDB supports row-level locking which means inserting and updating is much faster as compared with MyISAM.
- InnoDB supports ACID (Atomicity, Consistency, Isolation and Durability) property while MyISAM does not support.

- In InnoDB table,AUTO_INCREMENT field is a part of index.
- Once table in InnoDB is deleted then it can not re-establish.
- InnoDB does not save data as table level so while implementation of select count(*) from table will again scan the whole table to calculate the number of rows while MyISAM save data as table level so you can easily read out the saved row number.
- MyISAM does not support FOREIGN-KEY referential-integrity constraints while InnoDB supports.

DYNAMODB

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database, so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. Also, DynamoDB offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data
- It's a NOSQL DB so don't have a perfect schema, that means , entries in the database can have different attributes.
- It is highly available.
- The only requirement to have a DynamoDB is a partition key (primary key), if we don't have any unique value, we will then have a set of two values that will together make up a unique pair. The other value is called the sort key.
- Supports data encryption.
- Ideal type of database for mobile gaming, IOT and storing user session data.
- DB will scale out auto scale not only on the basis of increase in storage demand but also depends on read and write capacity units . nodes here are called read capacity units and write capacity units.
- Connection or queries to DynamoDB is purely through API and there are no serves accessible.
- By default, provides 5 read and write per second.
- Build on top of SSD (solid state drive)

➔ **SECONDARY INDEX**

- Amazon DynamoDB provides fast access to items in a table by specifying primary key values. However, many applications might benefit from having one or more secondary (or alternate) keys available, to allow efficient access to data with attributes other than the primary key. To address this, you can create one or more secondary indexes on a table, and issue Query or Scan requests against these indexes.
- A *secondary index* is a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations. You can retrieve data from the index using a Query, in much the same way as you useQuery with a table. A table can have multiple secondary indexes, which gives your applications access to many different query patterns.

➔ **CONSISTENCY MODEL FOR DYNAMODB**

When reading data from DynamoDB, the user can specify that the reading is final consistent or strongly consistent.

- **Final/Eventual Consistent Read (default)** - The final consistency option maximizes read throughput. However, the final consistent read may not reflect the results of recently completed writes. All copies of data are usually consistent within one second. After a short time, you should repeat reading to return the updated data.
- **Strong Consistent Readability** - DynamoDB provides flexibility and control to request consistent, consistent reads when required by elements of an application or application, in addition to eventual consistency. Strong Consistent Read returns results that reflect all writes that received a successful response before reading.

➔ **READ REQUEST UNITS AND WRITE REQUEST UNITS**

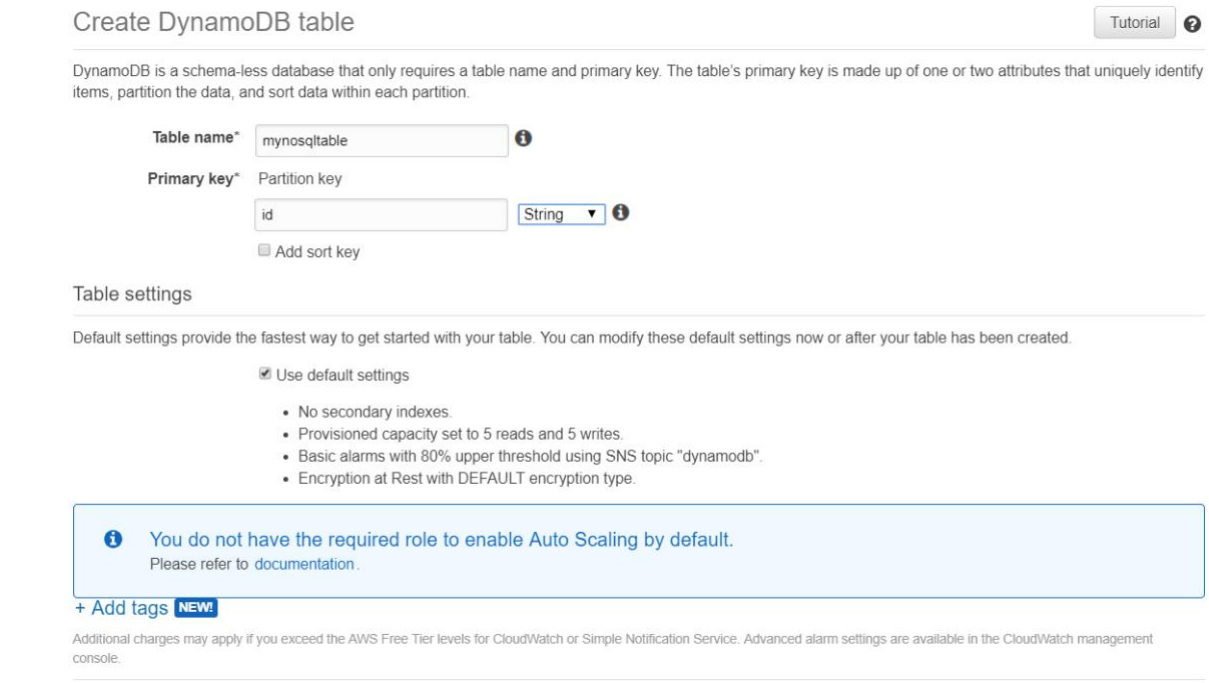
- One *read request unit* represents one strongly consistent read request, or two eventually consistent read requests, for an item up to 4 KB in size.
- One *write request unit* represents one write for an item up to 1 KB in size. If you need to write an item that is larger than 1 KB, DynamoDB needs to consume additional write request units

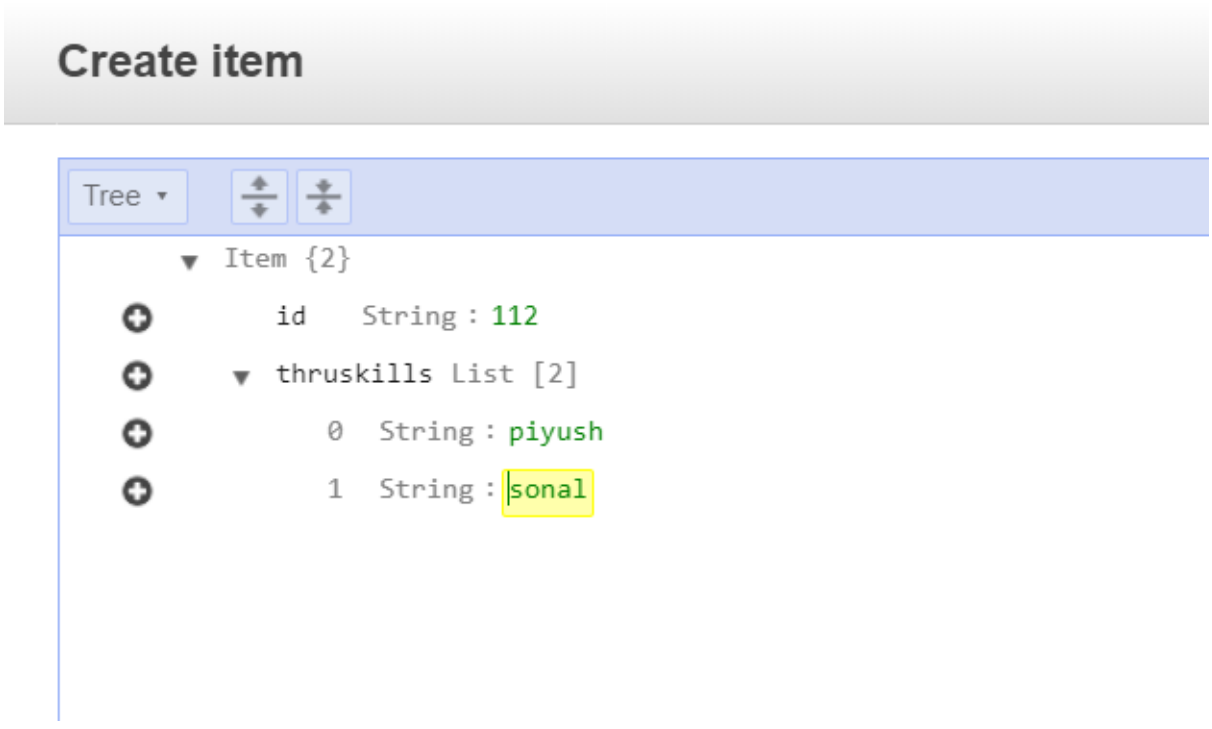
➔ **SCAN VS QUERY OPERATION**

- A scan operation scans all of the table and returns the result.
- Query operation gets us result as per our query.

➔ **CREATE A DYNAMODB TABLE**

- Services->DynamoDB->Create table





mynosqltable Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Backups | Triggers | Access control | Tags

Create item | Actions

Scan: [Table] mynosqltable: id

Scan | [Table] mynosqltable: id

Add filter

Start search

id	Department	Name	thruskills
101	devops	Piyush	
112			[{ "S" : "piyush" }, { "S" : "sonal" }]

ELASTICACHE

➔ WHAT IS ELASTICACHE?

- Used to offload DynamoDB and RDS
- Perfect solution for storing user session data.
- Amazon ElastiCache is a web service that makes it easy to deploy and run Memcached or Redis protocol-compliant server nodes in the cloud. Amazon ElastiCache improves the performance of web applications by allowing you to retrieve information from a fast, managed, in-memory system, instead of relying entirely on slower disk-based databases.
- The in-memory caching provided by Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads (such as social networking, gaming, media sharing and Q&A portals) or compute-intensive workloads (such as a recommendation engine). In-memory caching improves application performance by storing critical pieces of data in memory for low-latency access. Cached information may include the results of I/O-intensive database queries or the results of computationally-intensive calculations.

➔ TYPES OF CACHING ENGINES

- **MEMCACHED**
  - Simple model
  - Easy to scale
  - Multithreaded
- **REDIS**
  - Complex data types
  - MultiAZ failover
  - Data persistence
  - Snapshots for backup and restore
  - Supports data encryption (HIPPA compliant)

REDSHIFT

- Optimised for data analytics
- Good for OLAP
- When data is loaded, its converted to columnar DB.
- Data is stored in what we call as compute nodes
- Leader nodes manages the cluster, it takes queries from user, give it to compute nodes and send result back to user.
- Efficient data compression as columns mostly consists of similar data types.
- Massive parallel processing i.e. queries are distributed equally amongst all nodes
- NOT MULTI AZ, by default but we can take snapshots and make it multi AZ

AMAZON AROURA

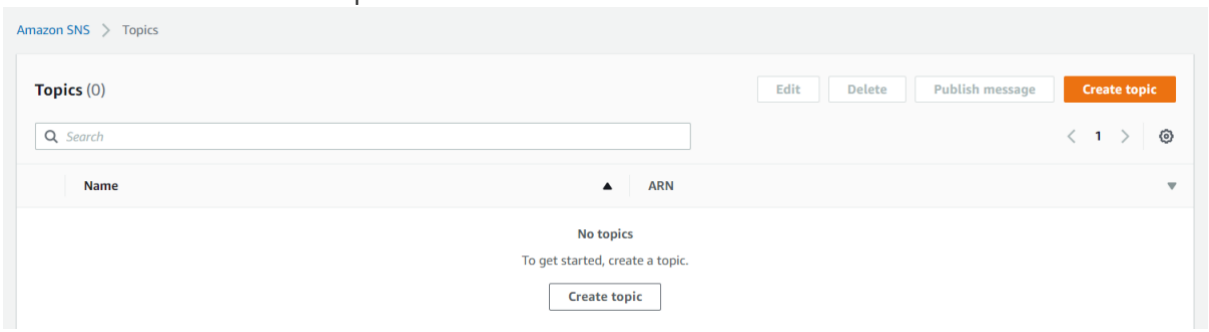
- Proprietary AWS RDS runs on AWS environment only
- MYSQL compatible
- Highly scalable, starts with 10 GB, and can go up to 64TB in the increment of 10 GB
- Can scale up to 32 virtual CPU and 244 GB memory
- Contain 2 copies of data in minimum 3 AZ
- Maintain 15 Aurora replicas.

SNS (SIMPLE NOTIFICATION SERVICE)

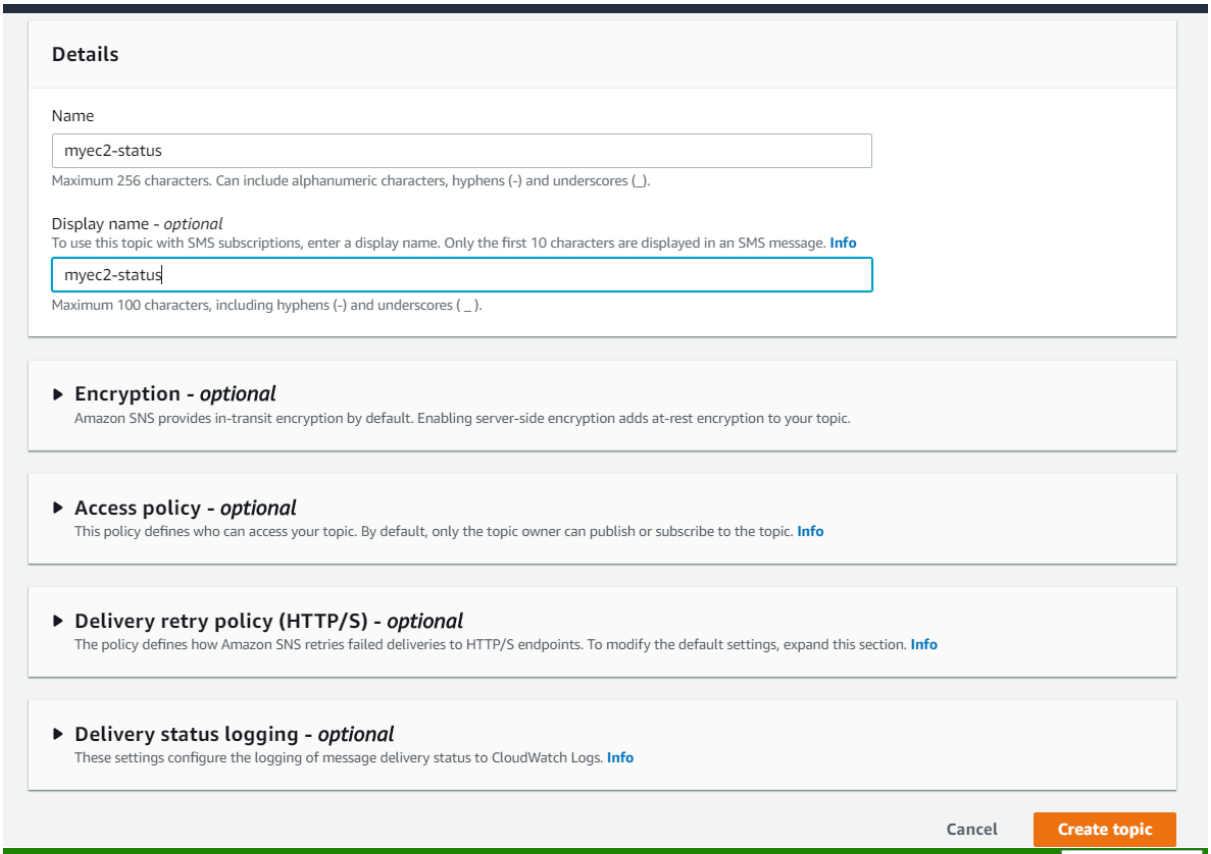
- Amazon Simple Notification Service (SNS) is a highly available, durable, secure, fully managed pub/sub messaging service
- Based on pub-sub model and is push based.
- Used to send out notifications to the users, in the event when certain event occurs.
- **SNS TOPICS**: Objects to which you publish your message
    - Subscribers subscribe to topics to receive notifications
    - Maximum text size can be 256 KB
- **Subscriber**: The end points to which messages are sent.
    a) HTTP
    b) HTTPS
    c) E-mail
    d) E-mail json
    e) SQS
    f) Lambda
    g) SMS
- **Publisher**: The entity that triggers the message such as S3 event, cloud watch alarm etc.

➔ Now let us see how it is done

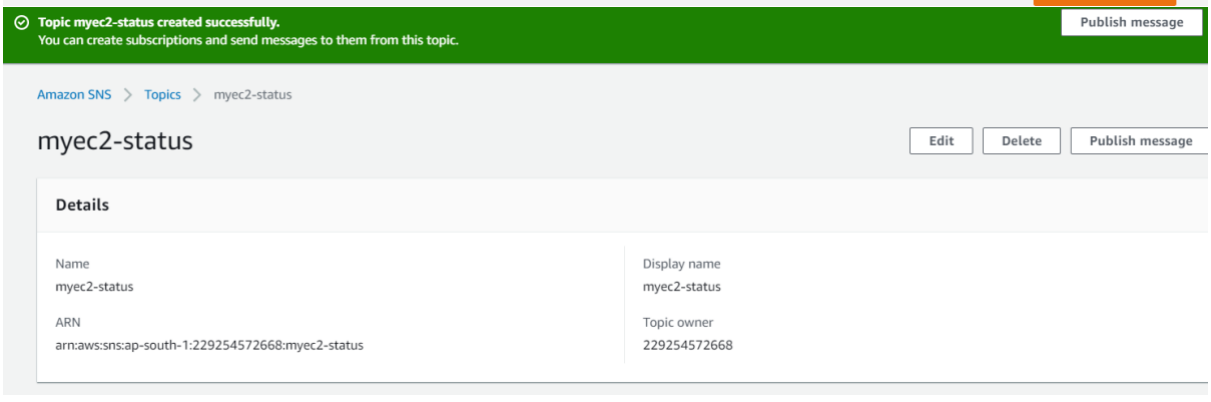
1) Services -> SNS-> Create topic



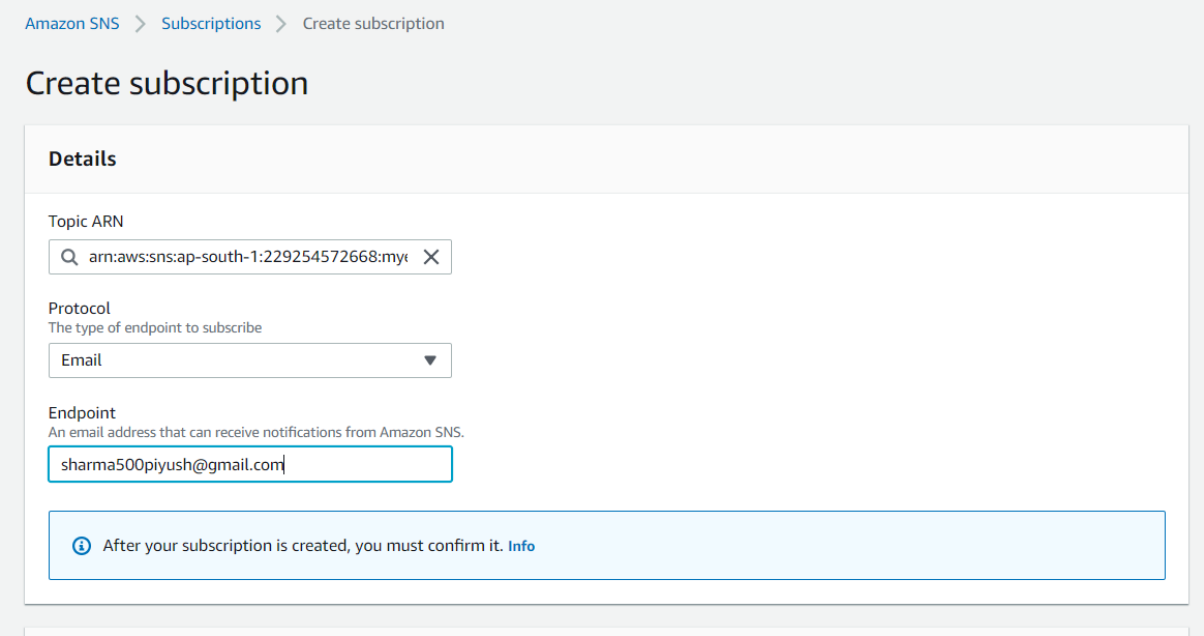
2)

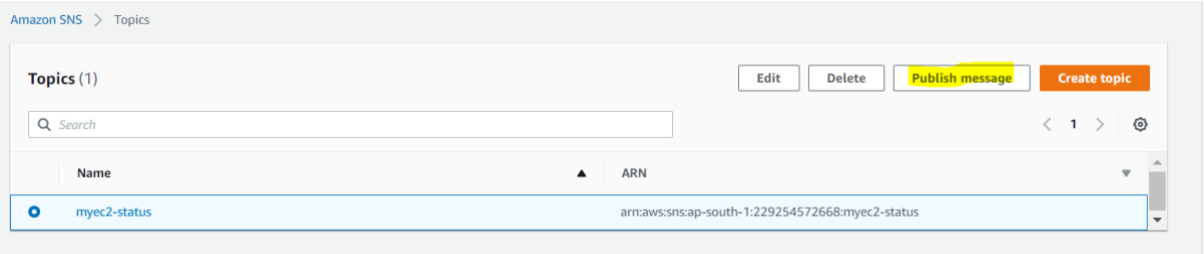

Details
Name
myec2-status
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).
Display name - optional
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. Info
myec2-status
Maximum 100 characters, including hyphens (-) and underscores (_).
Encryption - optional
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.
Access policy - optional
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. Info
Delivery retry policy (HTTP/S) - optional
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. Info
Delivery status logging - optional
These settings configure the logging of message delivery status to CloudWatch Logs. Info
Cancel
Create topic

3)


Topic myec2-status created successfully.
You can create subscriptions and send messages to them from this topic.
Publish message
Amazon SNS > Topics > myec2-status
myec2-status
Edit
Delete
Publish message
Details
Name
myec2-status
Display name
myec2-status
ARN
arn:aws:sns:ap-south-1:229254572668:myec2-status
Topic owner
229254572668

4) Now we will create subscription





Note that SNS is not the right service to send bulk mails as the text size can be a maximum of 256 KB. If bulk mails need to be sent with attachments, we can use anther service called SES (Simple Email service)

SQS – SIMPLE QUEUE SERVICE

- Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components
- **Tightly coupled systems**: These are applications where their architecture components are not highly dependent on each other. This is a scenario where if one component fails , the system will fail.
- **Loosely coupled/decoupled systems**: Multiple components that can process information without being connected. This means that if one component fails, the system can still keep processing.
- There are other queuing systems available in the market as well like IBM MQ, MQ etc but SQS is way better, Amazon SQS requires no administrative overhead and little configuration. Amazon SQS works on a massive scale, processing billions of messages per day. You can scale the amount of traffic you send to Amazon SQS up or down without any configuration. Amazon SQS also provides extremely high message durability, giving you and your stakeholders added confidence.
- **SQS USE CASE:** SQS is a great way to decouple services, especially when there is a lot of heavy-duty, batch-oriented processing required.

  For example, let's say you have a service where people upload photos from their mobile devices. Once the photos are uploaded your service needs to do a bunch of processing of the photos, e.g. scaling them to different sizes, applying different filters, extracting metadata, etc.

  One way to accomplish this would be to post a message to an SQS queue (or perhaps multiple messages to multiple queues, depending on how you architect it). The message(s) describe work that needs to be performed on the newly uploaded image file. Once the message has been written to SQS, your application can return a success to the user because you know that you have the image file and you have scheduled the processing.

  In the background, you can have servers reading messages from SQS and performing the work specified in the messages. If one of those servers dies another one will pick up the message and perform the work. SQS guarantees that a message will be delivered eventually so you can be confident that the work will eventually get done.

➔ **HOW DOES IT REALLY WORK?**

Messages between servers are retrieved through polling which can be of 2 types

- **SHORT POLLING**: returns immediately, even if the message queue being polled is empty. It should be noted that every API call made is charged so this increases the costs because we also pay for empty responses.
- **LONG POLLING**: Long polling doesn't return a response until a message arrives in the message queue, or the long poll times out. Long polling time period is between 1 to 20 seconds

➔ **VISIBILITY TIMEOUT**: When a consumer receives and processes a message from a queue, the message remains in the queue. Amazon SQS doesn't automatically delete the message. While the message is taken care or processed by a consumer instance, for that time period its hidden from other consumers. This is called visibility timeout. If in case message didn't get process in that time span, message is visible again and can be taken care by other worker instance. Default visibility time is 30 second, minimum is 0 seconds and max is 12 hours.

➔ **TYPES OF SQS QUEUES**

a) **Standard Queue**: Standard queues support a nearly unlimited number of transactions per second (TPS) per action.
   - At least once delivery
   - Best effort ordering

b) **FIFO Queue**: Can process 300 messages/second
   - Ensures the order of message processing
   - Message is processed exactly once

   ➔ **DEAD LETTER QUEUE**: Amazon SQS supports *dead-letter queues*, which other queues (*source queues*) can target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system

AMAZON MQ

- ➔ We studied in the last part that we have SQS as highly scalable message queuing service but all that stuff runs on Amazon proprietary infra and environment
- ➔ This is fine for new applications or applications under development
- ➔ To migrate the application that are already running on any other MQ like Rabbit MQ or Active MQ, we have Amazon MQ
- ➔ It is not highly scalable and runs on a single instance to start with.

SWF (SIMPLE WORKFLOW SERVICE)

- Amazon Simple Workflow Service (SWF) is a web service that makes it easy to coordinate work across distributed application components.
- Its perfect for the situations where while processing a code or a logic, decision needs to be taken and need to act differently as per different scenarios
- It ensures that task is only assigned once and there are no duplicates.
- Workflow validity is 1 year maximum.

➔ **SWF USE CASE**: It can be used to track order from amazon.com, it will involve checking availability of the product, payment method, Human staff going to the warehouse, packing and delivering, delivery confirmation etc.

➔ **SWF WORKERS**: Programs that receive the task, process the task and returns the result.

➔ **SWF DECIDER**: Programs that controls coordination of tasks i.e. their ordering, scheduling according to the application logic

➔ **SWF DOMAIN**: In SWF, you define logical containers called domains for your application resources. Domains can only be created at the level of your AWS account and may not be nested. A domain can have any user-defined name. Each application resource, such as a workflow type, an activity type, or an execution, belongs to exactly one domain. During registration, you specify the domain under which a workflow or activity type should be registered. When you start an execution, it is automatically created in the same domain as its workflow type. The uniqueness of resource identifiers (e.g. type-ids, execution ID) is scoped to a domain, i.e. you may reuse identifiers across different domains.

KINESIS

- ➔ Service used to store and analyse streaming data.
- ➔ **STREAMING DATA**: Data that is generated continuously by thousands of data sources that sends data continuously and in small size (in KB)
- ➔ Example use cases are stock prices, storing gaming stats, social network data eg Facebook, twitter likes, comments, number of views etc, uber location, our current location is constantly streamed and depending upon the traffic(which can be get from other data sources) , shortest path is decided.
- ➔ Kinesis streams are of 3 types:
  a) **Kinesis streams**: Data producers stream the data to Kinesis. Kinesis streams are a place to store data. Minimum 24 hours to 7 days. Data is contained in shards. EC2 instances, the consumes take data from shards and analyse it. From EC2 instances, the analysed data can be stored in DynamoDB , S3 or Redshift.
  b) **Kinesis Firehose**: This does not have shards and the data storage is non persistent unlike kinesis streams. As soon as the data comes in, it is processed and stored in S3.
  c) **Kinesis Analytics**: Used in conjunction with any of kinesis firehose or Kinesis streams, allow us to analyse data inside Kinesis.

CLOUD FORMATION

➔ AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; AWS CloudFormation handles all of that.

Template Sections

Templates include several major sections. The Resources section is the only required section. Some sections in a template can be in any order. However, as you build your template, it can be helpful to use the logical order shown in the following list because values in one section might refer to values from a previous section.

Format Version (optional)

The AWS CloudFormation template version that the template conforms to. The template format version is not the same as the API or WSDL version. The template format version can change independently of the API and WSDL versions.

Description (optional)

A text string that describes the template. This section must always follow the template format version section.

Metadata (optional)

Objects that provide additional information about the template.

Parameters (optional)

Values to pass to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.

Mappings (optional)

A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function in the Resources and Outputs sections.

Conditions (optional)

Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

Transform (optional)

For serverless applications (also referred to as Lambda-based applications), specifies the version of the AWS Serverless Application Model (AWS SAM) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it is processed.

You can also use AWS::Include transforms to work with template snippets that are stored separately from the main AWS CloudFormation template. You can store your snippet files in an Amazon S3 bucket and then reuse the functions across multiple templates.

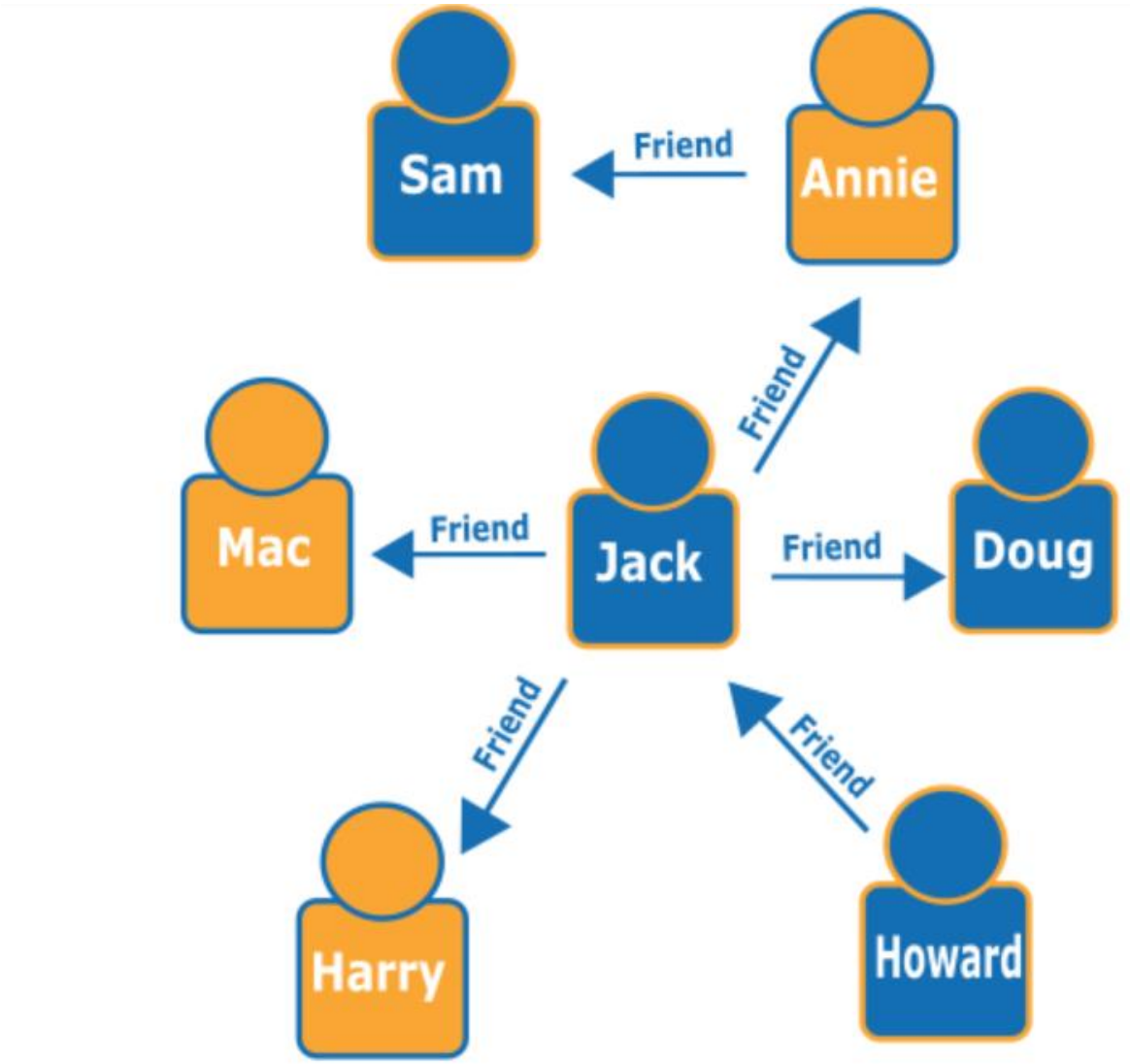
Resources (required)

Specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.

Outputs (optional)

Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name and then call the aws cloudformation describe-stacks AWS CLI command to view the name.

NEPTUNE



- ➔ Fully managed NOSQL DB
- ➔ Highly available
- ➔ Storage up to 64 TB
- ➔ 15 read replicas
- ➔ It is a graph database (The one of the types that Facebook uses)
- ➔ Graphs are built based on the relationships

LAMBDA

WHERE DOES LAMBDA SITS?

- ➔ AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.
- ➔ You pay only for the compute time you consume - there is no charge when your code is not running. With AWS Lambda, you can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. All you need to do is supply your code in one of the languages that lambda supports

➔ Lambda supports below languages
- NodeJS
- Ruby
- Python
- Java
- Go
- .Net
- Powershell

➔ You can use AWS Lambda to run your code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table.

➔ Also , to run your code in response to HTTP requests using Amazon API Gateway; or invoke your code using API calls made using AWS SDKs. With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB,

➔ Lambda can trigger another lambda function.

➔ Each request for lambda function triggers a new lambda function. 100 requests mean 100 lambda functions will be triggered.

➔ Lambda functions may be invoked via API calls.

➔ Lambda is really economical, first one million requests are free, and $ 0.20 per one million requests thereafter

➔ Lambda cost is calculated from the time your code begins executing to the time it returns result or terminates. It is priced on the basis of memory you allocate for the function execution.

➔ When we talk to Alexa, we are communicating with lambda

➔ Lambda scales out (not up) automatically.

➔ Lambda functions are independent, one event equals one function.

➔ Lambda is serverless, no concept of EC2

➔ All RDS instances except Aurora are not serverless.

➔ Lambda architecture can get extremely complicated so we have a service called Amazon X ray to debug in case of any issues.

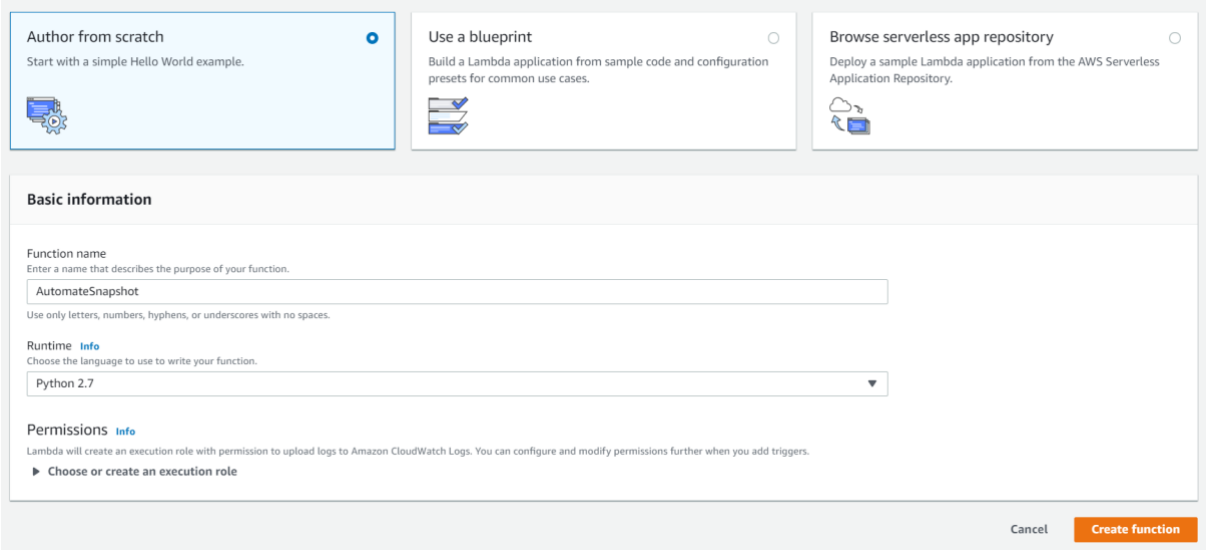
➔ Lambda function can execute for maximum of 5 minutes

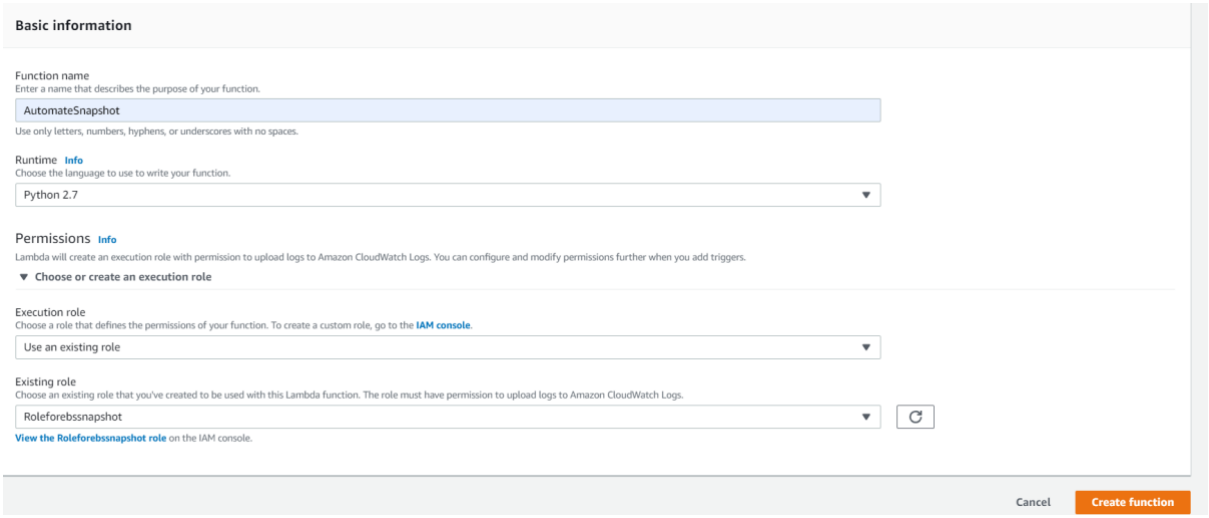
➔ Memory allocation can be between 128 MB to 3 GB

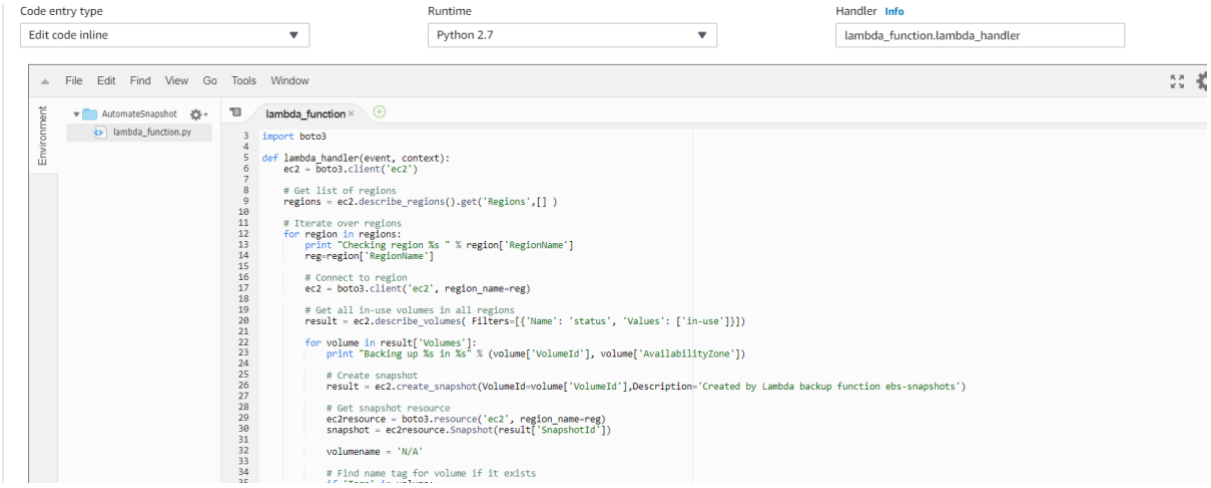
➔ Below services can trigger lambda

API Gateway
AWS IoT
Application Load Balancer
CloudWatch Events
CloudWatch Logs
CodeCommit
Cognito Sync Trigger
DynamoDB
Kinesis
S3
SNS
SQS

- **SAMPLE APPLICATION**: Here below lambda function will execute to take snapshots of all the EBS volumes attached to our EC2 instances.
  - Go to services -> Lambda -> Create a function-> Author from scratch







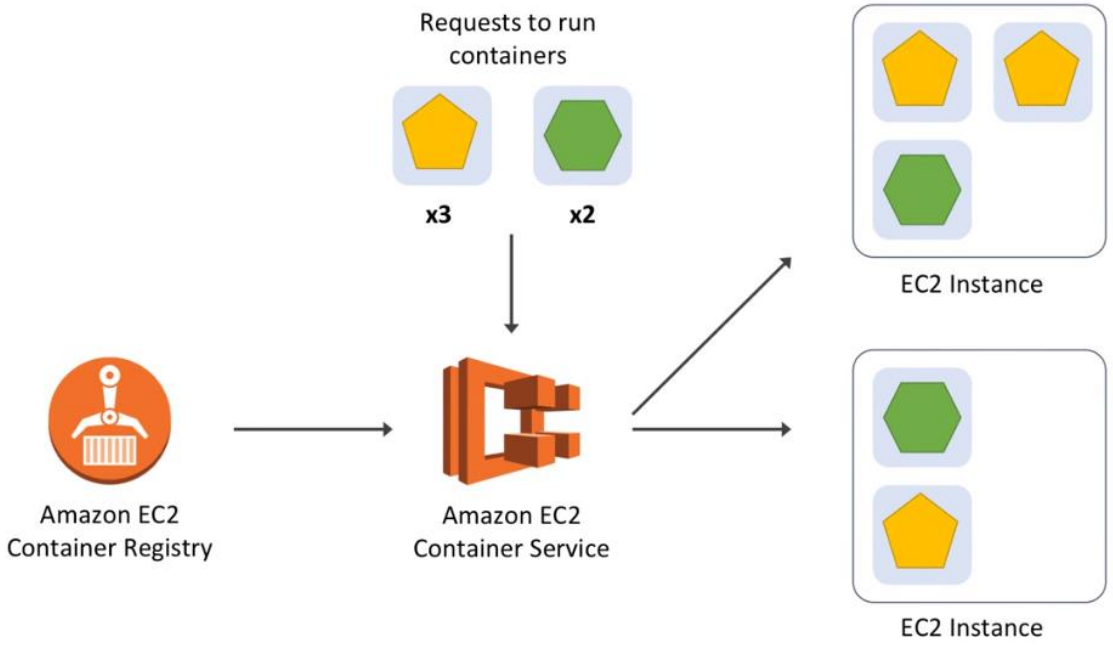
- ➔ Python code is taken from https://www.codebyamir.com/blog/automated-ebs-snapshots-using-aws-lambda-cloudwatch
- ➔ Save the code
- ➔ Click on test-> Create test event-> Just save it as it is as we are not triggering any event, we trigger lambda function manually.
- ➔ Once triggered , we will find the snapshots of the volumes created.

API GATEWAY

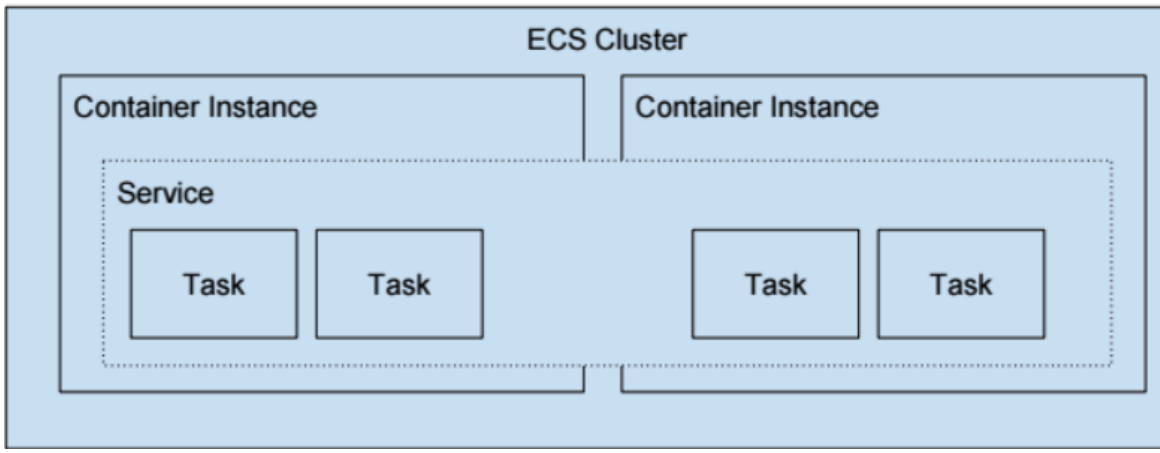
- ➔ While we work in development, we want to have such a backend that can interact with multiple frontend. E.g., an application can have a backend for mobile as well as a web interface.
- ➔ To make the backend sharable, what we have is an API (Application programming interface).
- ➔ API is nothing but collection of methods that can be called. With this, the frontend may be in any language, but that does not matter to the backend.
- ➔ API gateway is a fully managed service that allows you to create and manage your own API for your application
- ➔ API gateway acts as a front door for your application allowing access to your data logic/functionality to your backend.
- ➔ Restful API can be developed.(with GET,PUT,POST methods)
- ➔ API can be created for different environments i.e Dev, test, and stage each of them running independently.
- ➔ API version can be created by cloning an existing one and can be rolled back to previous API deployments.
- ➔ We can create Custom domain names can point to different APIs
- ➔ Create and manage API keys to restrict access for authorised users only and measure usage of the API through cloud watch.
- ➔ Set throttling rules based on the number of requests per second.
- ➔ Security via Signature version 4 to authorise API calls (for credentials generated through Amazon Cognito or STS (security Token service).
- ➔ We can enable API caching to cache your endpoint response ,with caching we can reduce the number of calls made to our APi and also reduce the latency.
- ➔ Scales automatically
    - **SAME ORIGIN ACCESS POLICY VS CROSS ORIGIN RESOURCE SHARING (CORS)**
        - In computing, the same origin policy is an important concept where web browser permits scripts contained in the first webpage to access data in the second webpage only if they have the same origin i.e. the same domain name. This is done to prevent cross-site scripting attacks.
        - But while working with AWS we need to deal with different origin pr different domain names i.e. an S3 wil have a different domain and so will a cloud front. For this we enable CORS.
- ➔ API requests can also be throttled (controlled)

ELASTIC CONTAINER SERVICE (ECS)

- ➔ Here everything starts with Docker container.
- ➔ Docker packages software in in to standardized units called containers that has everything your software needs to run including libraries, system tools code and runtime.
- ➔ Let us quickly scale and deploy applications in to any environment.
- ➔ It is operating system as well as architecture independent. Machine should be able to run Docker.
- ➔ Important component of ECS is the ECR (Elastic container registry). It helps to store manage and deploy container images. It is integrated with ECS and is redundant, highly available and encrypted.
- ➔ We can provide necessary permissions with IAM.



- ➔ Can utilise on-demand, spot or reserved EC2 instances.

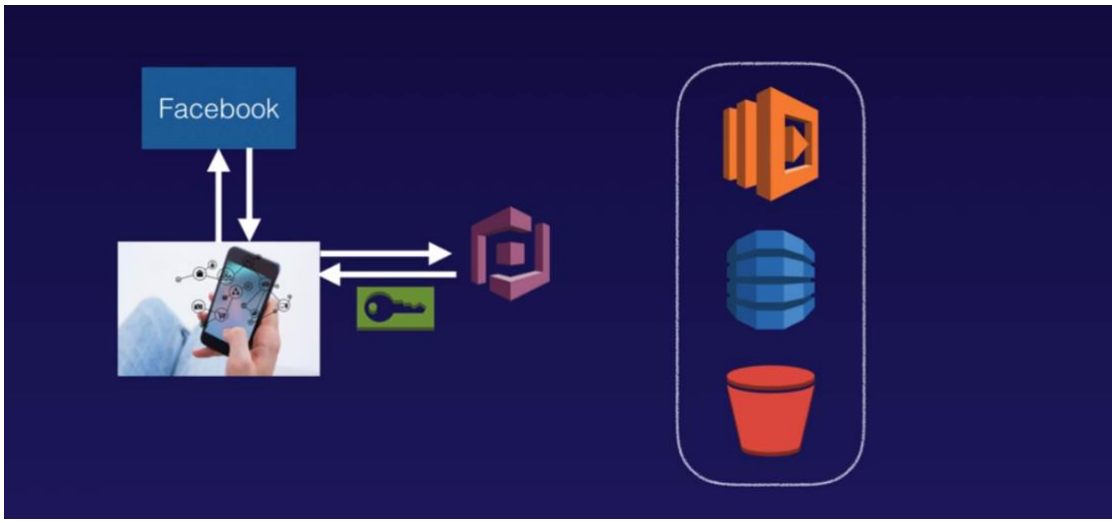


- **ECS TERMINOLOGY**
  - **Task Definition** — this a blueprint that describes how a docker container should launch. If you are already familiar with AWS, it is like a LaunchConfig except instead it is for a docker container instead of a instance. It contains settings like exposed port, docker image, cpu shares, memory requirement, command to run and environmental variables.
  - **Task** — this is a running container with the settings defined in the Task Definition. It can be thought of as an "instance" of a Task Definition.
  - **Service** — Defines long running tasks of the same Task Definition. This can be 1 running container or multiple running containers all using the same Task Definition**.**
  - **Cluster**: An ECS cluster is just a logical group of EC2 instances that we can place containers in to.
  - **Container Instance** — This is just an EC2 instance that is part of an ECS Cluster and has Docker and the ecs-agent running on it.

➔ **IAM role can be attached to an ECS at the container instance level or at the task level**

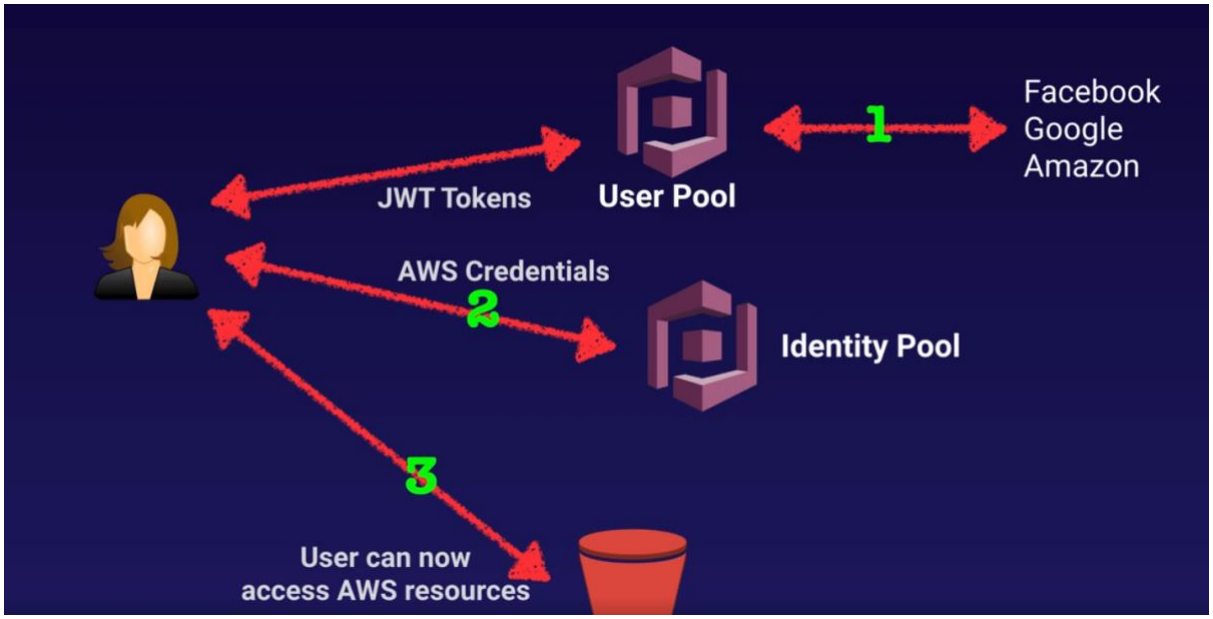
WEB IDENTITY FEDERATION AND COGNITO

- ➔ Web identity federation gives us the facility to give access to the AWS resources(to our users)after they have successfully authenticated with a web based identity provider like amazon, facebook,google etc
- ➔ Following successful authentication the users receive an authentication code which they can use to temporarily access AWS resources.
- ➔ Web identity federation service in AWS is known as cognito.
  - Sign in and sign up to our apps
  - Sync the credentials or user data across all user devices.



➔ Cognito brokers between the AWS app and facebook or google to provide temporary credentials that map to an IAM role so there is no need to store the keys anywhere.

- USER POOLS: User pools are user directories used to manage user sign up and sign in functionality. Successful authentication generates a JSON web token (JWT)
- IDENTITY POOL: Enable provide temporary AWS credentials to access AWS services like S3 or DynamoDB



ELASTIC BEANSTALK

- ➔ Aimed at developers who have no idea about AWS
- ➔ We just need to select our development environment and upload the code
- ➔ That's it , now everything will be taken care by EBS, all the infrastructure, security groups etc will be provisioned by Elastic beanstalk

AWS SECURITY WHITE PAPER AT GLANCE

- **AWS WELL ARCHITECTED FRAMEWORK**
  - ➔ **BEST GENERAL DESIGN PRINCIPLES**
    - Do not guess your capacity needs, this may lead to excessive number of resources, as cloud is flexible, you can scale anytime.
    - Test systems at production scale
    - Automate at maximum to make changes and experimentation easier.
    - Allow for evolutionary architecture I,e the architecture should not be static and should evolve with needs and accommodate changes.
    - Drive architecture using data i.e.in the cloud you can collect data on how your architectural choice effect the workload, data can be analysed and necessary changes should be made.
    - Improve through game days i.e. perform drill , smoke and DR test for checking the high availability of the infra.
    - Whenever building an application, below pillars need to be kept in mind.

The pillars of the AWS Well-Architected Framework

Pillar Name	Description
Operational Excellence	The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.
Security	The ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.
Reliability	The ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.
Performance Efficiency	The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
Cost Optimization	The ability to run systems to deliver business value at the lowest price point.

1) **OPERATIONAL EXCELLENCE PILLAR**
   - Perform operations as code
   - Annotate documentation //code comments can be our documentation
   - Make frequent, small reversible changes
   - Refine operations procedures frequently
   - Anticipate failures
   - Learn from all operational failures.
2) **SECURITY**

   AWS SHARED RESPNSIBILITY MODEL

AWS provides secure infrastructure and services, while you, the customer, are responsible for secure operating systems, platforms, and data. To ensure a secure global infrastructure, AWS configures infrastructure components and provides services and features you can use to enhance security, such as the Identity and Access Management (IAM) service, which you can use to manage users and user permissions in a subset of AWS services.

The shared responsibility model for infrastructure services, such as Amazon Elastic Compute Cloud (Amazon EC2) for example, specifies that AWS manages the security of the following assets:

- Facilities
- Physical security of hardware
- Network infrastructure
- Virtualization infrastructure

In this Amazon EC2 example, you as the customer are responsible for the security of the following assets:

- Amazon Machine Images (AMIs)
- Operating systems
- Applications
- Data in transit
- Data at rest
- Data stores
- Credentials
- Policies and configuration

➔ **USING THE TRUSTED ADVISOR TOOL**

Some AWS Premium Support plans include access to the Trusted Advisor tool, which offers a one-view snapshot of your service and helps identify common security misconfigurations, suggestions for improving system performance, and underutilized resources. Trusted Advisor checks for compliance with the following security recommendations:

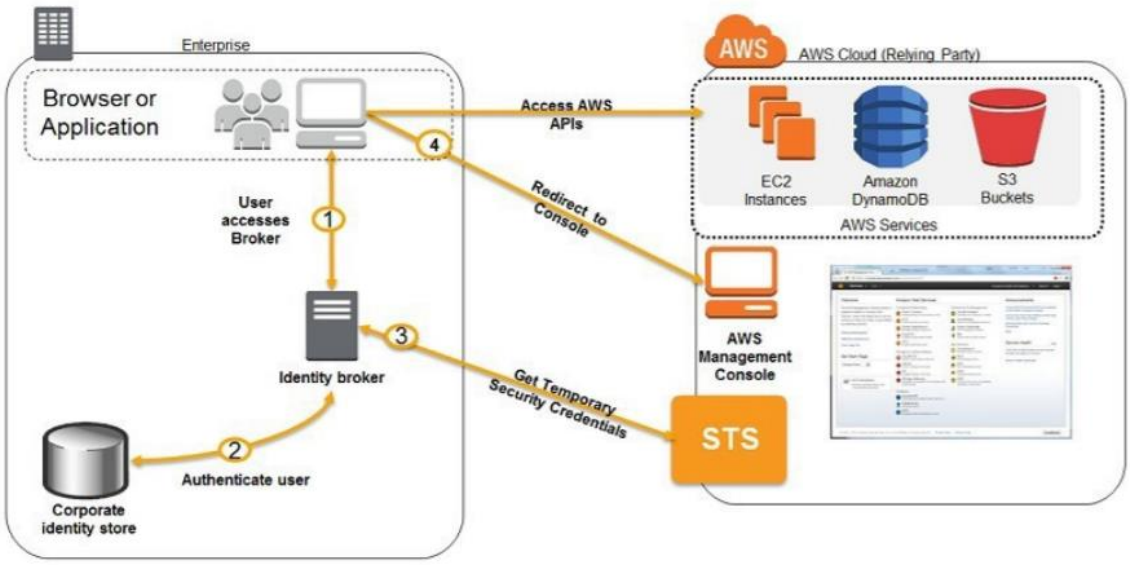
- Limited access to common administrative ports to only a small subset of addresses. This includes ports 22 (SSH), 23 (Telnet) 3389 (RDP), and 5500 (VNC).
- Limited access to common database ports. This includes ports 1433 (MSSQL Server), 1434 (MSSQL Monitor), 3306 (MySQL), Oracle (1521) and 5432 (PostgreSQL).
- IAM is configured to help ensure secure access control of AWS resources.
- Multi-factor authentication (MFA) token is enabled to provide two-factor authentication for the root AWS account

➔ **CROSS ACCOUNT ACCESS**

You can use IAM roles to enable cross account access or to allow users of one account access resources of another account.address IAM users from another AWS account to access resources within your AWS account. This process is referred to as cross-account access. Cross-account access lets you share access to your resources with users in other AWS accounts. To establish cross-account access, in the trusting account (Account A), you create an IAM policy that grants the trusted account (Account B) access to specific resources. Account B can then delegate this access to its IAM users. Account B cannot delegate more access to its IAM users than the permissions that it has been granted by Account A.

➔ **IDENTITY FEDERATION**

Identity federation is the process of making use of corporate Active directory authentication with AWS.



1. The enterprise user accesses the identity broker application.

2. The identity broker application authenticates the users against the corporate identity store.

3. The identity broker application has permissions to access the AWS Security Token Service (STS) to request temporary security credentials.

4. Enterprise users can get a temporary URL that gives them access to the AWS APIs or the Management Console. A sample identity broker application for use with Microsoft Active Directory is provided by AWS.

=> **HSM**

To secure your encryption keys, we can use a tamper proof hardware device known as **Hardware security module (HSM)**. They are also provided in cloud and are known as cloud HSM.If keys from HSM are lost, they cannot be recovered unless we have a copy of it.

➔ **DECOMMISSION DATA AND MEDIA SECURELY**

When you ask AWS to delete data in the cloud, AWS does not decommission the underlying physical media; instead, the storage blocks are marked as unallocated. AWS uses secure mechanisms to reassign the blocks elsewhere. When you provision block storage, the hypervisor or virtual machine manager (VMM) keeps track of which blocks your instance has written to. When an instance writes to a block of storage, the previous block is zeroed out, and then overwritten with your block of data. If your instance attempts to read from a block previously written to, your previously stored data is returned. If an instance attempts to read from a block it has not previously written to, the hypervisor zeros out the previous data on disk and returns a zero to the instance

When AWS determines that media has reached the end of its useful life, or it experiences a hardware fault, AWS follows the techniques detailed in Department of Défense (DoD)

➔ **AMAZON CORPORATE SEGREGATION**
Logically the amazon production network is separated from Amazon corporate network by means of complex set of security devices.

➔ Unauthorised port scanning by amazon customers is a violation of Amazon's acceptable use policy. Customers can run vulnerability scans but after permissions from AWS.

➔ Amazon uses XEN hypervisor

➔ Different instances on the same host are separated from each other in a highly secure way.

➔ Memory allocated to a guest OS is scrubbed (set to 0) by the hypervisor and when it is marked as unlocatable, it is returned to the free memory pool.

⇨ **RELIABILLITY**

➔ The third pillar is reliability is defined as ability to recover from failure and mitigate disruptions

➔ Our infra should be highly available and should take the least possible time to get recovered in the event of failure

➔ Recovery should be automated.

➔ Our infra should scale as required.

➔ We should ensure that we have proper service limits

➔ AWS has a tool called "Trusted Advisor" which can give us insight about the below things

- Service Limits
- Cost optimization (idle resources, capacity reservations)
- Performance (High Utilization)
- Security (IAM permissions, security groups)
- Fault tolerance(EBS snapshots, multi AZ etc)

➔ **AWS mitigate the DDOS attack with their service called "shield"**

■ **DISASTER RECOVERY DESIGN PATTERNS**

- **RTO (Recovery time objective)** : This is the time that we are allowed to take to recover in case of a disaster.
- **RPO (Recovery point objective)** : This is the amount of data loss which we can sustain when the data restore happens after recovering from disaster.

1) **Backup and Restore**: EBS snapshots are taken, AMIs are created in other regions and setup is restored from there in case of disaster.
2) **Pilot light**: A similar setup is there in another region, ready to go but servers are powered off (so that we pay less to AWS). In the event of disaster, servers are powered on. And DNS records are switched over.
3) **Low capacity standby:** This is similar to pilot light but here a minimum portion of our infra is always running 24*7. This needs to be tested always. Maybe we can use weighted routing policy to always send a small amount of traffic to our standby infra.
4) **Multi-site (Active Active):** Two twin sites running all the time in parallel

⇨ **PERFORMANCE EFFICIENCY**

1) Preferably go for managed services from AWS as far as possible.
2) Pick most efficient resource type eg the EC2 type, EBS volumes, database etc
3) Use direct connect while connecting on premises data centre to AWS
4) Implement caching using elasticache wherever possible.

⇨ **COST OPTIMIZATION**

1) The ability to avoid or eliminate unneeded costs
2) Analyse and attribute expenditure.
3) Choose right instance and storage option
4) Scale according to the load
5) Use cost allocation tags
6) Continuously re-evaluate.
7) We can generate reports from **cost explorer**.

OTHER SERVICES

➔ **VM IMPORT/EXPORT**: VM Import/Export enables you to easily import virtual machine images from your existing environment to Amazon EC2 instances and export them back to your on-premises environment. This offering allows you to leverage your existing investments in the virtual machines that you have built to meet your IT security, configuration management, and compliance requirements by bringing those virtual machines into Amazon EC2 as ready-to-use instances. You can also export imported instances back to your on-premises virtualization infrastructure, allowing you to deploy workloads across your IT infrastructure.

VM Import/Export is available at no additional charge beyond standard usage charges for Amazon EC2 and Amazon S3.

➔ **OPSWORKS**: This is an orchestration service which helps us perform configuration tasks on our EC2 instances using Chef. (orchestration)/chef

➔ **AWS ORGANIZATIONS AND CONSOLIDATED BILLING**
AWS organizations is a service that allows us to centrally managed multiple AWS accounts. The master account is called the paying account and others are called linked account. It saves money when we use consolidated billing. There can be maximum **20 linked accounts by default**.

- Paying account with multiple linked accounts
- Paying account is independent and should be only used for billing purpose
- Paying account cannot access resources of other accounts unless given exclusively access through Cross Account roles
- **allows unused Reserved Instances to be applied across the group**

IMPORTANT POINTS

- IAM
  - 1 MFA device per account is allowed
  - We can enable/disable user access keys
  - Each user can have maximum of 2 key pairs
  - Individual IAM users cannot have independent ssh keys
  - User names can or cannot be email addresses'
  - Access keys can be compromised so good to use roles
  - By default, 500 roles per account can be created.
  - A role can be assigned to already running instance
  - We cannot assign more than 1 role to an instance
  - A service linked role is the one which can be assumed by the service and not by a user
  - The IAM policies can be of 3 types
    a) AWS managed policy
    b) Customer managed policy
    c) Inline policy: These are policies that are directly assigned to user, group or role. Inline policies are useful if you want to maintain a strict one-to-one relationship between a policy and the principal entity that it's applied to. For example, you want to be sure that the permissions in a policy are not inadvertently assigned to a principal entity other than the one they're intended for.
- S3
  - Individual Amazon S3 objects can range in size from a minimum of **0 bytes** to a maximum of 5 terabytes. The largest object that can be uploaded in a single PUT is 5 gigabytes. For objects larger than 100 megabytes, customers should consider using the Multipart Upload capability.
  - Similarly, multipart delete option is also available.
  - S3 naming convention
    http://S3-region.amazonaws.com/bucket-name
    http://bucket.s3-awsregion.amazonaws.com
  - In general, bucket owners pay for all Amazon S3 storage and data transfer costs associated with their bucket. A bucket owner, however, can configure a bucket to be a Requester Pays bucket. With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing data.
- EC2

➔ User data is the script that we pass while launching an EC2 instance and the things that we mention in the script are created when the instance comes up.

➔ **How does spot pricing work?**

1) You (or an application running on your behalf) submits a bid to run a desired number of EC2 instances of a particular type. The bid includes the price that you are willing to pay to use the instance for an hour.

2) When your bid price exceeds the current Spot price (which varies based on supply and demand), your instances are run.
3) When the current Spot price rises above your bid price, the Spot instance is reclaimed by AWS so that it can be given to another customer.

- Copying a source AMI results in an identical but distinct target AMI with its own unique identifier.
- AWS does not copy launch permissions, user-defined tags, or Amazon S3 bucket permissions from the source AMI to the new AMI. After the copy operation is complete, you can apply launch permissions, user-defined tags, and Amazon S3 bucket permissions to the new AMI.
- Data IO, read write is slow when it is done for the first time on a disk created from an EBS snapshot because it is initialised and scanned first when put to use. This performance loss is only one time.
- Also there is a performance loss when a snapshot of an EBS volume is in progress.

New Spot Instance Termination Notice

Today we are improving the reclamation process with the addition of a two-minute warning, formally known as a Spot Instance Termination Notice. Your application can use this time to save its state, upload final log files, or remove itself from an Elastic Load Balancer.

- When we talk about ELB, instances behind the ELB can have two states InService and OutOfService. As soon as an instance is marked as Out of service , load balancer stops sending traffic to that instance.

- **AUTOSCALING TERMINATION POLICIES**

Before Auto Scaling selects an instance to terminate, it first identifies the availability zone used by the group that contains the most instances. If all availability zones have the same number of instances, Auto Scaling selects a random availability zone, and then uses the termination policy to select the instance within that randomly selected availability zone for termination.

By default, Auto Scaling chooses the instance that was launched with the oldest launch configuration. If more than one instance was launched using the oldest launch configuration, Auto Scaling the instance that was created first using the oldest launch configuration.

You can override the default instance termination policy for your Auto Scaling group with one of the following options:

OldestInstance	The oldest instance in the Auto Scaling group should be terminated.
NewestInstance	The newest instance in the Auto Scaling group should be terminated.
OldestLaunchConfiguration	The first instance created using the oldest launch configuration should t Be terminated.

Default	Use the default instance termination policy.

- DYNAMODB
  - Nosql database
  - **SECONDARY INDEX**
  - Amazon DynamoDB provides fast access to items in a table by specifying primary key values. However, many applications might benefit from having one or more secondary (or alternate) keys available, to allow efficient access to data with attributes other than the primary key. To address this, you can create one or more secondary indexes on a table, and issue Query or Scan requests against these indexes.
  - A secondary index is a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations. You can retrieve data from the index using a Query, in much the same way as you useQuery with a table. A table can have multiple secondary indexes, which gives your applications access to many different query patterns.
  - The size of the name and value pair in Dynamodb should not be more than 400KB.

  - **GLOBAL SECONDARY INDEX**

    This is an index with a partition key and a sort key that can be different from the base table. A global secondary index is very helpful when you need to query your data without a primary key.

- The primary key of a global secondary index can be partition key or composite (partition key and sort key).
- Global secondary indexes can be created at the same time that you create a table. You can also add a new global secondary index to an existing table or delete an existing global secondary index
- A global secondary index lets you query over the entire table and across all partitions.
- The index partition key and sort key (if present) can be any base table attributes of type string, number, or binary.
- With global secondary index queries or scans, you can only request the attributes that are projected into the index. DynamoDB will not fetch any attributes from the table.
- There are no size restrictions for global secondary indexes.

  - **LOCAL SECONDARY INDEX**

This is an index that has the same partition key as the base table, but a different sort key.

- The primary key of a local secondary index must be composite (partition key and sort key).
- Local secondary indexes are created at the same time that you create a table. You cannot add a local secondary index to an existing table, nor can you delete any local secondary indexes that currently exist.
- When you query a local secondary index, you can choose either eventual consistency or strong consistency.

- If you query or scan a local secondary index, you can request attributes that are not projected into the index. DynamoDB will automatically fetch those attributes from the table.
- Queries or scans on a local secondary index consume read capacity units from the base table. When you write to a table, its local secondary indexes are also updated; these updates consume write capacity units from the base table.

➔ **AMAZON DYNAMODB ACCELERATOR(DAX)**

Amazon DynamoDB is designed for scale and performance. In most cases, the DynamoDB response times can be measured in single-digit milliseconds. However, there are certain use cases that require response times in microseconds. For these use cases, *DynamoDB Accelerator (DAX)* delivers fast response times for accessing eventually consistent data.

DAX is a DynamoDB-compatible caching service that enables you to benefit from fast in-memory performance for demanding applications. DAX addresses three core scenarios:

1. As an in-memory cache, DAX reduces the response times of eventually-consistent read workloads by an order of magnitude, from single-digit milliseconds to microseconds.
2. DAX reduces operational and application complexity by providing a managed service that is API-compatible with Amazon DynamoDB, and thus requires only minimal functional changes to use with an existing application.
3. For read-heavy or bursty workloads, DAX provides increased throughput and potential operational cost savings by reducing the need to over-provision read capacity units. This is especially beneficial for applications that require repeated reads for individual keys.

➔ CloudFront supports geo restrictions, signed URL verification for content viewing.

➔ Direct connect is a service used to connect on premises data centre to AWS through a private line, another alternative is VPN which is quick to setup and should be chosen when we want a quick setup

➔ To get the instance meta data, go to the URL
curl http://169.254.169.254/latest/metadata
User data cannot be fetched, only instance meta data can be.

➔ Amazon Resource Names (ARNs) uniquely identify AWS resources. We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, Amazon Relational Database Service (Amazon RDS) tags, and API calls.

➔ WAF (Web application Firewall) : This is a security feature that works on layer 7
- For WAF we create an IAM user
- We create a web ACL , this blocks or allows web requests based on the conditions you specify. Below can be the valid condition type.
  - ➢ IP match condition
  - ➢ String match condition

- SQL injection match condition
- Size constraints condition
- Cross site scripting match condition

➔ SAML -Security Assertion Markup Language (SAML) is a standard for logging users into applications based on their sessions in another context.

➔ **HOW SAML WORK?**

SAML SSO works by transferring the user's identity from one place (the identity provider) to another (the service provider). This is done through an exchange of digitally signed XML documents.

Consider the following scenario: A user is logged into a system that acts as an identity provider. The user wants to log in to a remote application, such as a support or accounting application (the service provider). The following happens:

1. The user accesses the remote application using a link on an intranet, a bookmark, or similar and the application loads.
2. The application identifies the user's origin (by application subdomain, user IP address, or similar) and redirects the user back to the identity provider, asking for authentication. This is the authentication request.
3. The user either has an existing active browser session with the identity provider or establishes one by logging into the identity provider.
4. The identity provider builds the authentication response in the form of an XML-document containing the user's username or email address, signs it using an X.509 certificate, and posts this information to the service provider.
5. The service provider, which already knows the identity provider and has a certificate fingerprint, retrieves the authentication response and validates it using the certificate fingerprint.
6. The identity of the user is established and the user is provided with app access.