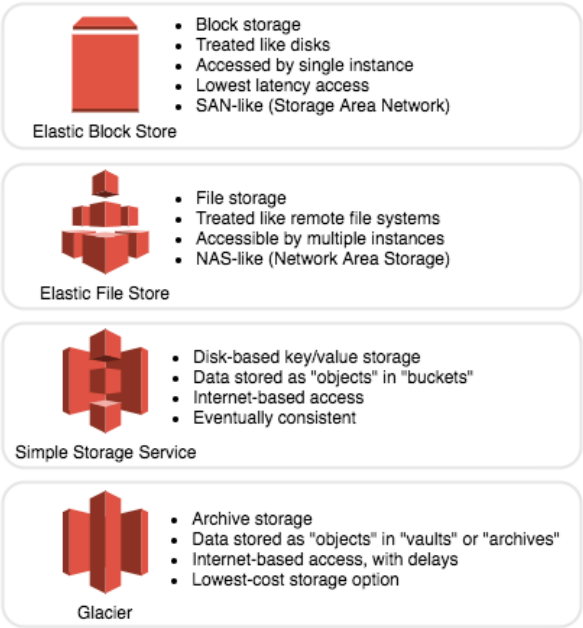
Chapter 4.0 : AWS S3 service

==========

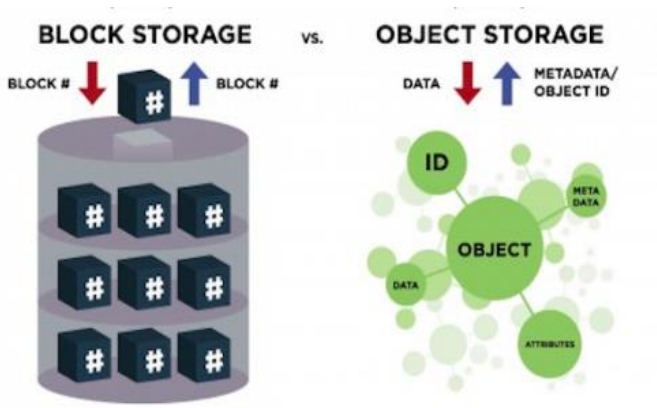
AWS Storage:



4.1 S3 -storage service:

What is object storage?

- Object storage is a computer data storage architecture that manages data as objects, as opposed to other storage architectures like file systems which manages data as a file hierarchy, and block storage which manages data as blocks.



Chapter 4.0 : AWS S3 service

==========

What is S3?

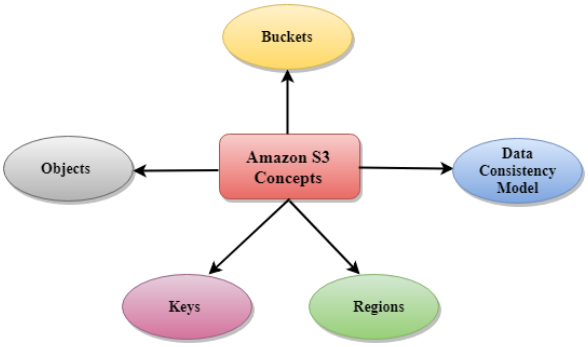
- S3 is a safe place to store the files.
- It is Object-based storage, i.e., you can store the images, word files, pdf files, etc.
- The files which are stored in S3 can be from 0 Bytes to 5 TB.
- It has unlimited storage means that you can store the data as much you want.
- Files are stored in Bucket. A bucket is like a folder available in S3 that stores the files.
- S3 is a universal namespace, i.e., the names must be unique globally. Bucket contains a DNS address. Therefore, the bucket must contain a unique name to generate a unique DNS address.
- By default, you can create up to 100 buckets in each of your AWS accounts. If you need additional buckets, you can increase your account bucket limit to a maximum of 1,000 buckets by submitting a service limit increase.
- You cannot create bucket inside bucket.
- This is similar to google drive but more advanced for technical people.
- This is not like block storage we cannot mount anywhere.
- It's a only drive we do not need to create any file system.
- It is designed to make web-scale computing easier for IT people.
- S3 has a distributed data-size architecture where objects are redundantly stored in multiple locations (min 3 zones).
- If you create a bucket, URL look like:

Amazon S3 Concepts

- Buckets
- Objects
- Keys
- Regions
- Data Consistency Model

Chapter 4.0 : AWS S3 service

==========



- **Buckets**
  - A bucket is a container used for storing the objects.
  - Every object is incorporated in a bucket.
  - For example, if the object named photos/tree.jpg is stored in the treeimage bucket, then it can be addressed by using the URL http://treeimage.s3.amazonaws.com/photos/tree.jpg.
  - No bucket can exist inside of other buckets.
  - S3 performance remains the same regardless of how many buckets have been created.
  - You can create or upload multiple folders in one bucket but you cannot create a bucket inside a bucket (Nested bucket not possible)
  - By default buckets and its objects are private, thus by default only the owner can access the buckets
  - Max capacity of 1 bucket is 5TB
  - You can have 100 buckets per account, this number can be increased.
  -

- **Objects**
  - Objects are the entities which are stored in an S3 bucket.
  - An object consists of object data and metadata where metadata is a set of name-value pair that describes the data.
  - It is uniquely identified within a bucket by key and version ID.

Chapter 4.0 : AWS S3 service

==========

- **Key**
  - A key is a unique identifier for an object.
  - Every object in a bucket is associated with one key.
  - An object can be uniquely identified by using a combination of bucket name, the key, and optionally version ID.
  - For example, in the URL http://jtp.s3.amazonaws.com/2019-01-31/Amazons3.wsdl where "jtp" is the bucket name, and key is "2019-01-31/Amazons3.wsdl"

- **Regions**
  - You can choose a geographical region in which you want to store the buckets that you have created.
  - A region is chosen in such a way that it optimizes the latency, minimize costs or address regulatory requirements.

- **Data Consistency Model**
  Amazon S3 replicates the data to multiple servers to achieve high availability.

Chapter 4.0 : AWS S3 service

==========

Amazon S3 (Simple Storage Service) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

Some key features of S3 include:

Object storage: S3 stores data as objects within buckets. Each object can be up to 5 terabytes in size and can include any type of data.

High durability: S3 stores data across multiple devices in multiple facilities, and automatically replicates data to help protect against failures.

High availability: S3 stores data in multiple locations and automatically replicates data across those locations to help ensure that data is always available.

Security: S3 provides multiple mechanisms for protecting data, including encryption, access controls, and identity and access management.

Scalability: S3 can scale to store and retrieve any amount of data, at any time, from anywhere on the web.

Global access: S3 data can be accessed from anywhere in the world, and can be configured to be delivered through a content delivery network (CDN) for fast, low-latency access.

Pay-as-you-go pricing: S3 pricing is based on the amount of data stored, the number of requests made, and the data transfer out of the service.

S3 also supports versioning: which means it stores all versions of an object (including all writes and deletes) allowing customers to easily recover from both unintended user actions and application failures.

Chapter 4.0 : AWS S3 service

==========

While Amazon S3 is a highly scalable and reliable service, there are a few potential disadvantages to consider:

Cost: While S3's pay-as-you-go pricing model can be cost-effective for some use cases, it can become expensive for very large amounts of data or high levels of traffic. Additionally, data transfer costs can also add up for transferring data to and from S3, especially if a lot of data is being transferred over long distances.

Latency: While S3 is designed for high availability, accessing data stored in a different region than the user can result in increased latency.

Limited file system support: S3 is an object storage service, not a file system, so it does not support some features commonly found in file systems such as a hierarchical directory structure, file locks, and metadata for individual files.

Complexity: While S3 is a simple storage service, managing and optimizing data storage can still require a significant amount of work, especially for large data sets and complex workflows.

Security: S3 provides multiple mechanisms for protecting data, but it's still the user's responsibility to properly configure them and the user will be liable in case of any data breaches.

Retrieval time: Retrieval time can be slow for large data sets stored in S3 Glacier (AWS's cold storage option) and it may take several hours to retrieve data from it.

S3 sub resources:

- Lifecycle
- Website
- Versioning
- Access control lists
- Cross region replication

Chapter 4.0 : AWS S3 service

Amazon S3 Storage Classes

	Storage class	Designed for	Availability Zones	Min storage duration	M b c
◉	Standard	Frequently accessed data (more than once a month) with milliseconds access	≥ 3	-	-
○	Intelligent-Tiering	Data with changing or unknown access patterns	≥ 3	-	-
○	Standard-IA	Infrequently accessed data (once a month) with milliseconds access	≥ 3	30 days	1
○	One Zone-IA	Recreatable, infrequently accessed data (once a month) stored in a single Availability Zone with milliseconds access	1	30 days	1
○	Glacier Instant Retrieval	Long-lived archive data accessed once a quarter with instant retrieval in milliseconds	≥ 3	90 days	1
○	Glacier Flexible Retrieval (formerly Glacier)	Long-lived archive data accessed once a year with retrieval of minutes to hours	≥ 3	90 days	-
○	Glacier Deep Archive	Long-lived archive data accessed less than once a year with retrieval of hours	≥ 3	180 days	-
○	Reduced redundancy	Noncritical, frequently accessed data with milliseconds access (not recommended as S3 Standard is more cost effective)	≥ 3	-	-

Amazon S3 Standard for frequent data access

- For frequently accessed data
- It is a default storage class
- Can be used for cloud applications, dynamic websites, content distribution, gaming applications, and Big data analytics

Amazon S3 Standard for infrequent data access

- For infrequently accessed data
- Demands rapid access
- Suitable for backups, disaster recovery and lifelong storage of data

Amazon Glacier

- Suitable for archiving data where data access is infrequent
- Vault-lock feature provides a long term data storage
- Provides the lowest cost availability

One Zone-IA Storage Class

- Suitable for infrequently accessed data
- Unlike other classes, this new storage class stores the data in a single AWS Availability Zone
- Data that doesn't require any high level of security can be stored here

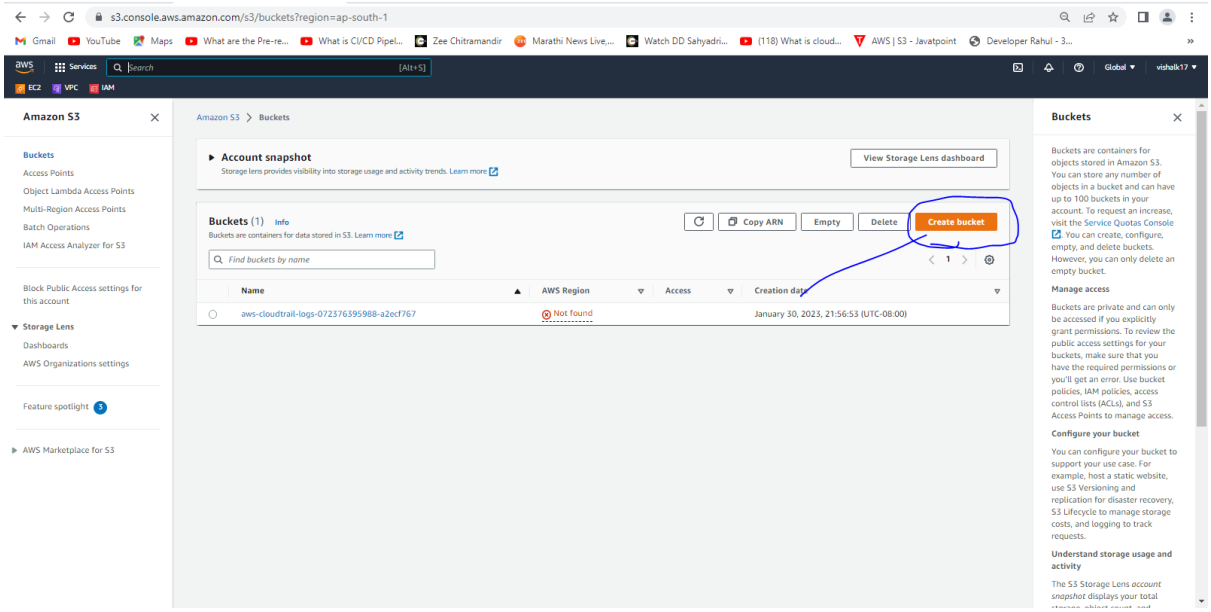
Amazon S3 Standard Reduced Redundancy storage

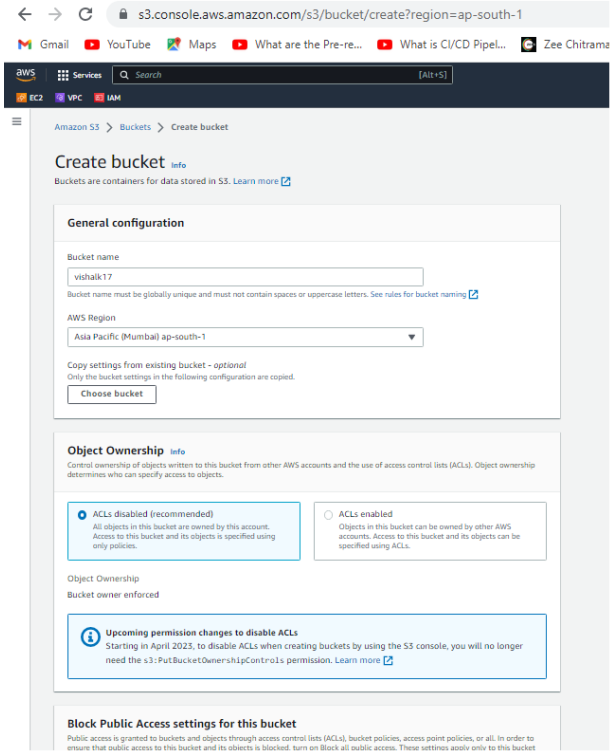
- For frequently accessed data
- Stores reproducible and non crucial data at lower cost
- A highly available solution designed for sharing or storing data that can be reproduced quickly

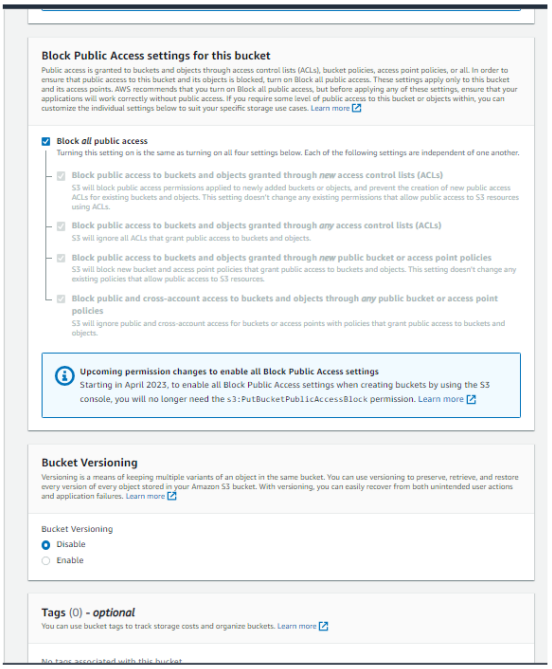
Chapter 4.0 : AWS S3 service

==========

Practical : create bucket

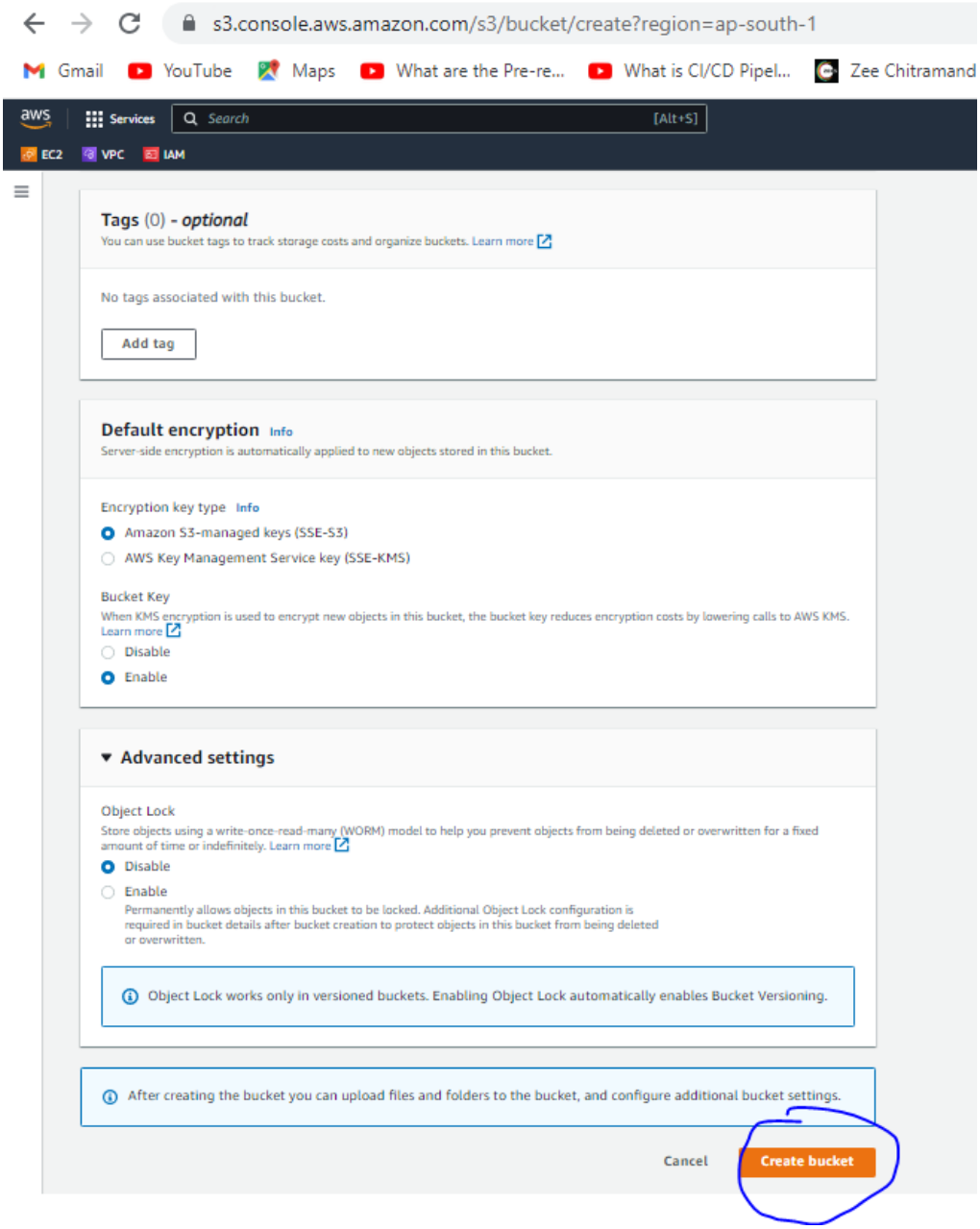






Chapter 4.0 : AWS S3 service

==========



Chapter 4.0 : AWS S3 service

==========

Bucket name : Bucket name must be globally unique and must not contain spaces or uppercase letters.

Example bucket names

The following example bucket names are valid and follow the recommended naming guidelines:

- `docexamplebucket1`
- `log-delivery-march-2020`
- `my-hosted-content`

The following example bucket names are valid but not recommended for uses other than static website hosting:

- `docexamplewebsite.com`
- `www.docexamplewebsite.com`
- `my.example.s3.bucket`

The following example bucket names are *not* valid:

- `doc_example_bucket` (contains underscores)
- `DocExampleBucket` (contains uppercase letters)
- `doc-example-bucket-` (ends with a hyphen)

Object Ownership:

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)

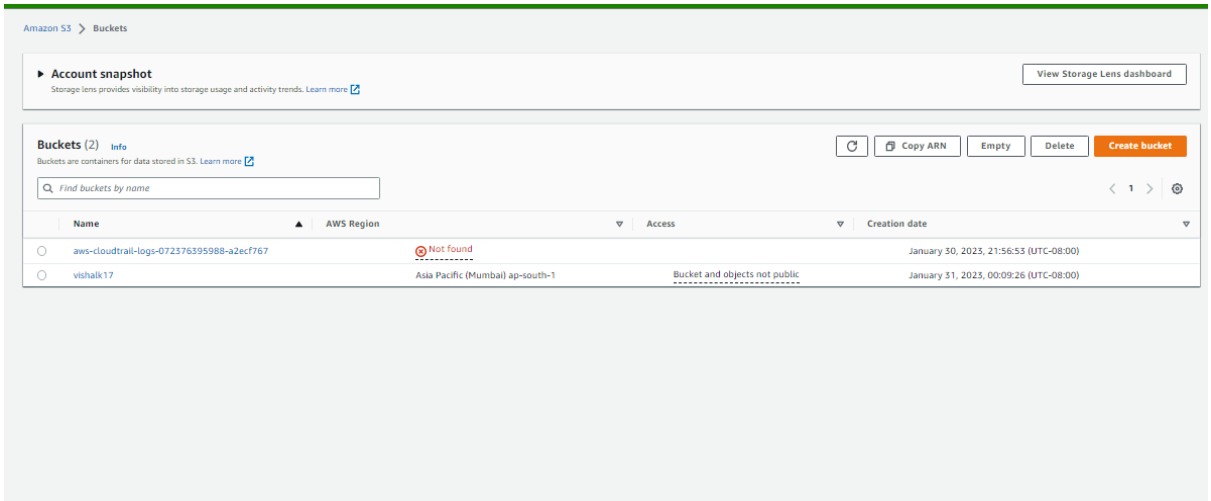
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled

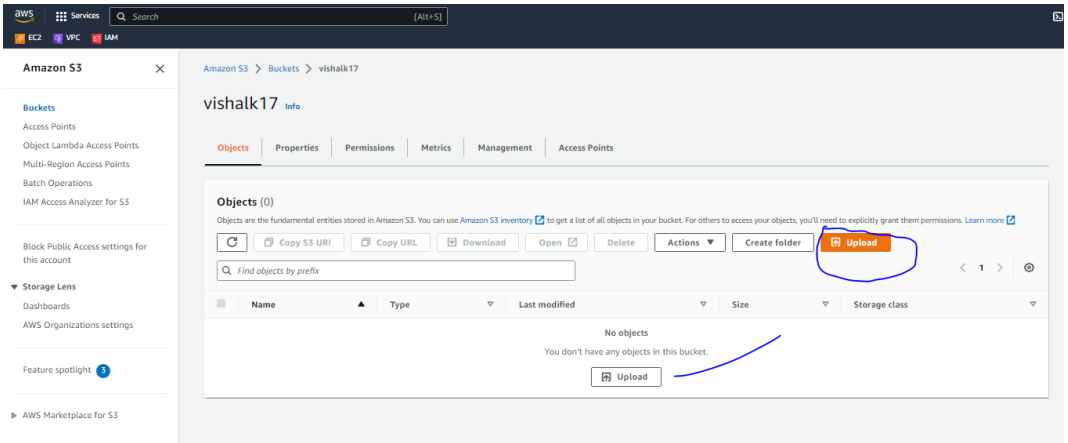
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

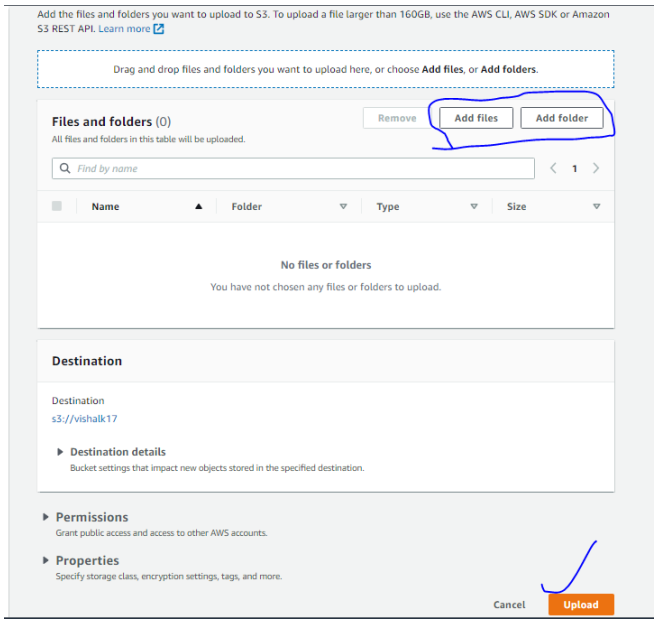
Chapter 4.0 : AWS S3 service

==========



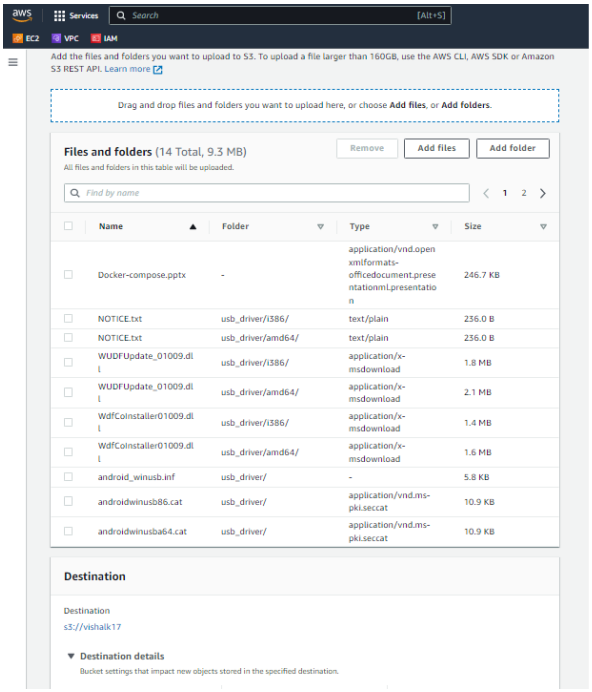
Practical 2: uploading data in s3 bucket

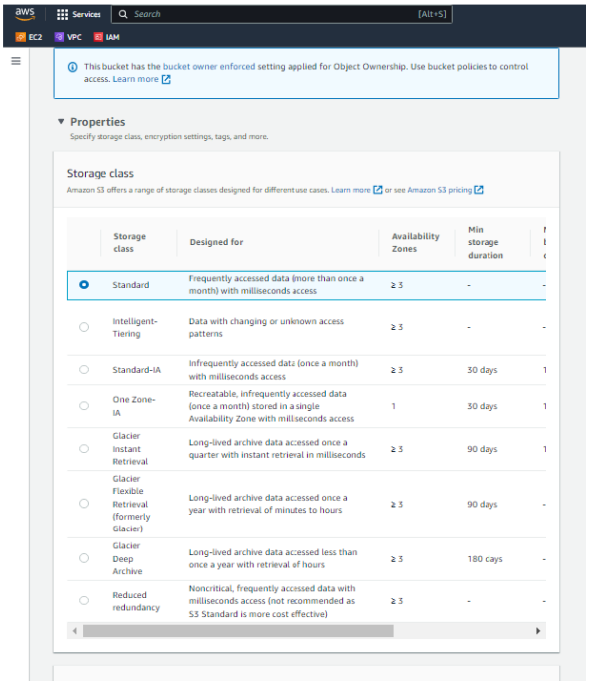


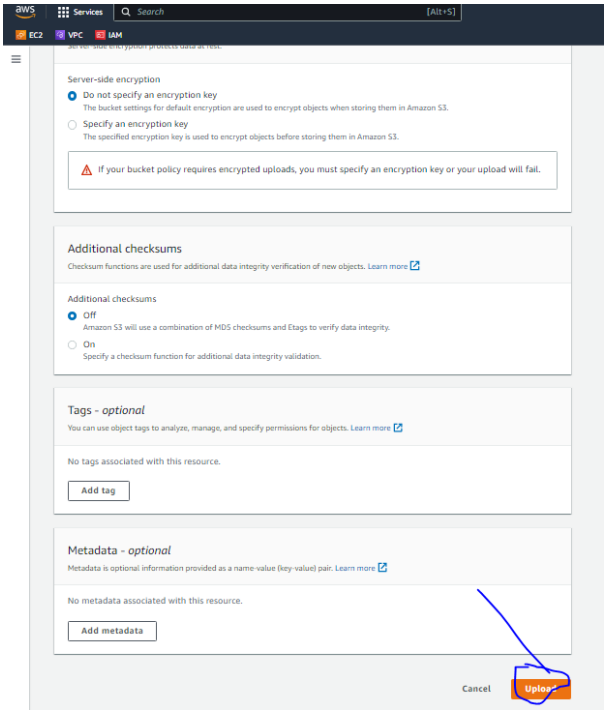


Chapter 4.0 : AWS S3 service

==========

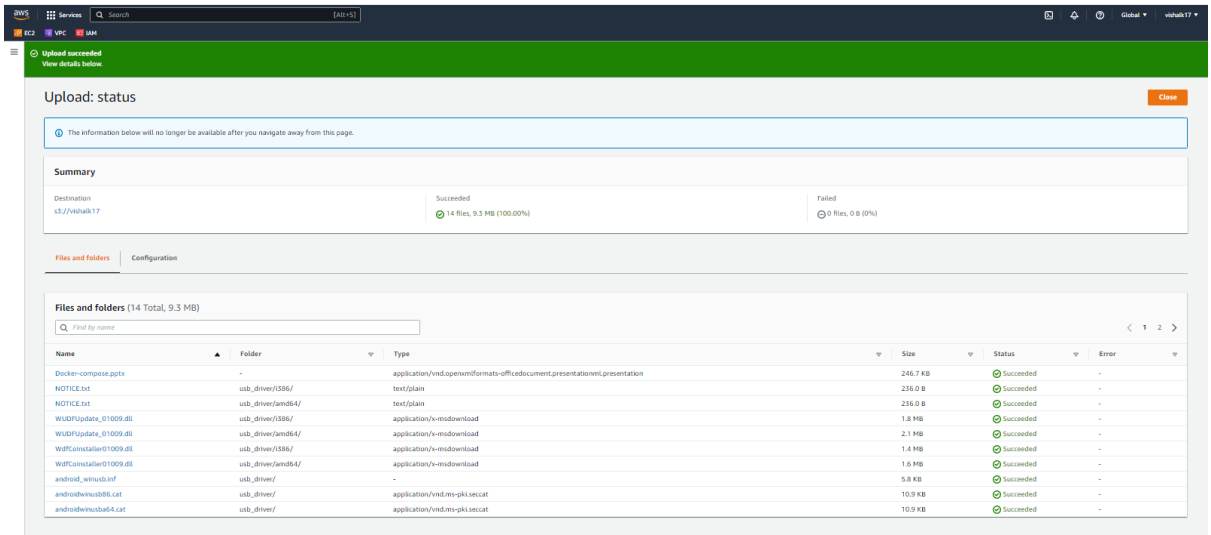


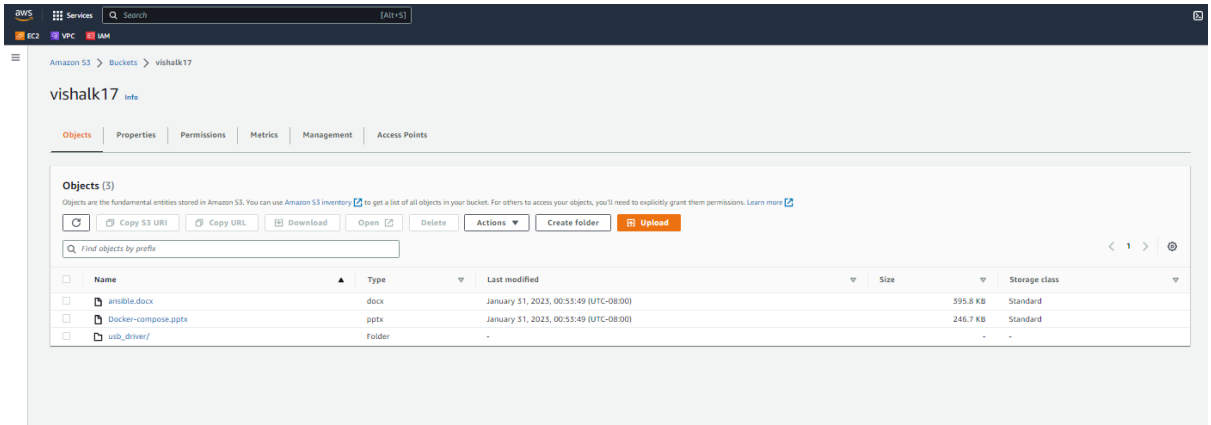




Chapter 4.0 : AWS S3 service

==========





Chapter 4.0 : AWS S3 service

==========

Bucket Versioning :

S3 bucket versioning allows multiple versions of an object to be stored in the same bucket, preserving all versions even if some are deleted. A delete marker is a special version of an object that is created when a version of an object is deleted.

To enable versioning for a bucket:

- Go to the S3 management console
- Select the bucket you want to enable versioning for
- Click on the Properties tab
- Click on the Versioning configuration button
- Select the "Enabled" option and click Save.

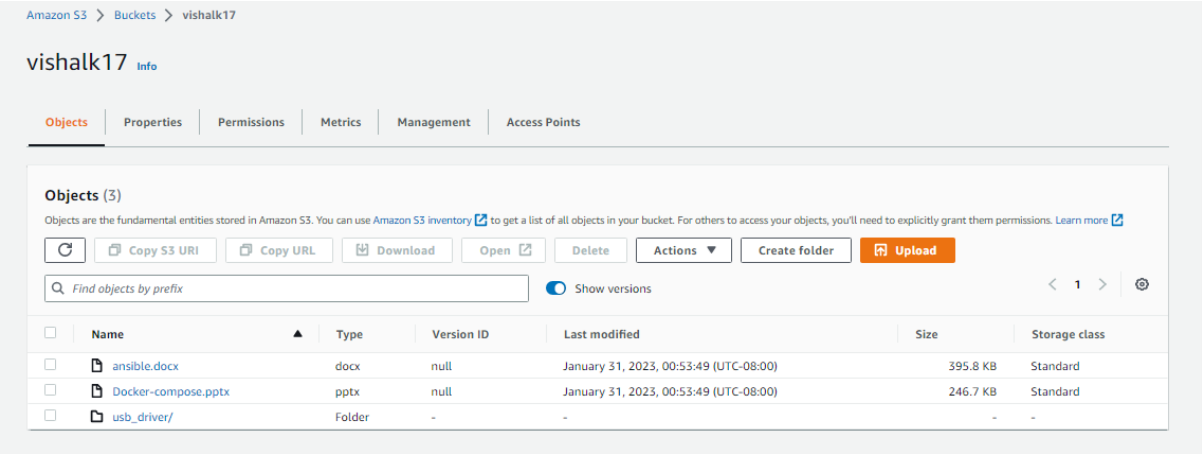
Versioning is useful for several reasons:

- Protects against accidental deletes and overwrites
- Enables recovery of previous versions of an object
- Facilitates object archiving by preserving all versions of an object in a bucket.

To recover a deleted object in an S3 bucket with versioning enabled:

1. Go to the S3 management console.
2. Select the bucket containing the deleted object.
3. Click on the "Versions" button to view all versions of objects in the bucket.
4. Find the deleted object (it will have a "delete marker" version) and select it.
5. Click the "Actions" button and select "Restore".
6. Choose the desired version of the object to restore.
7. The deleted object will now be restored and available in the bucket.

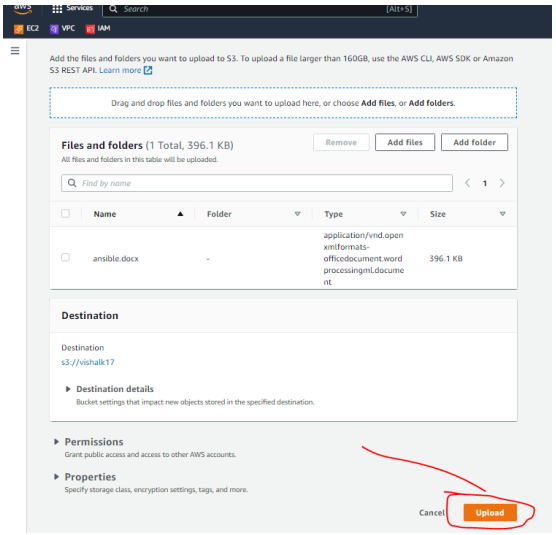
Note: You can only recover deleted objects & previous version of object if available in a bucket with versioning enabled. If versioning is not enabled, deleted objects are permanently lost.

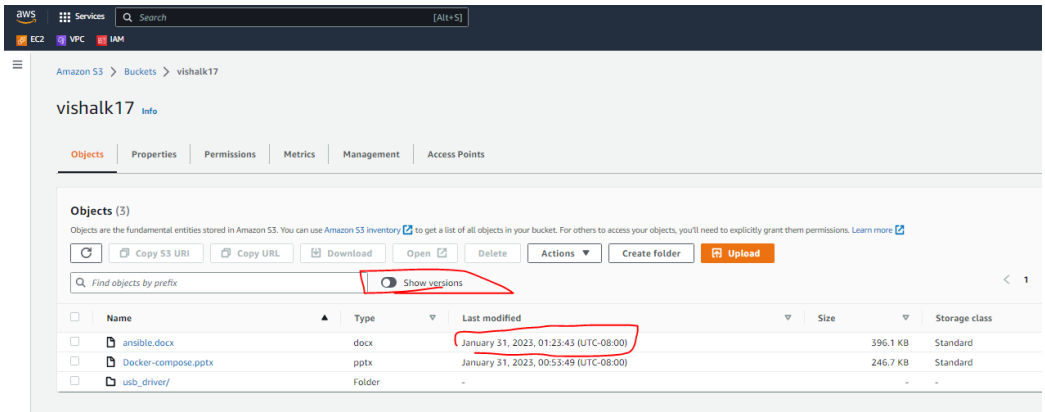


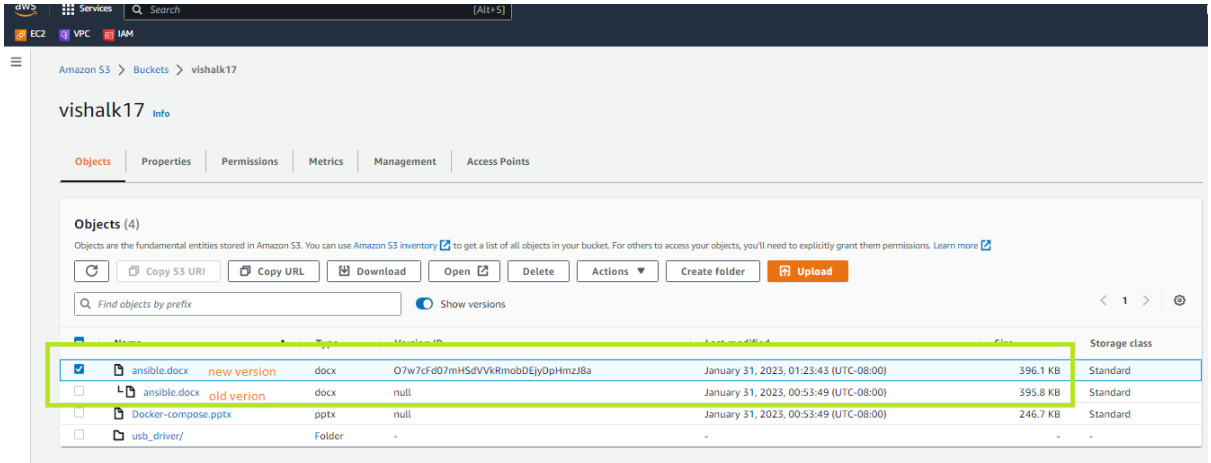
Chapter 4.0 : AWS S3 service

==========

Edited and upload again same ansible.docx





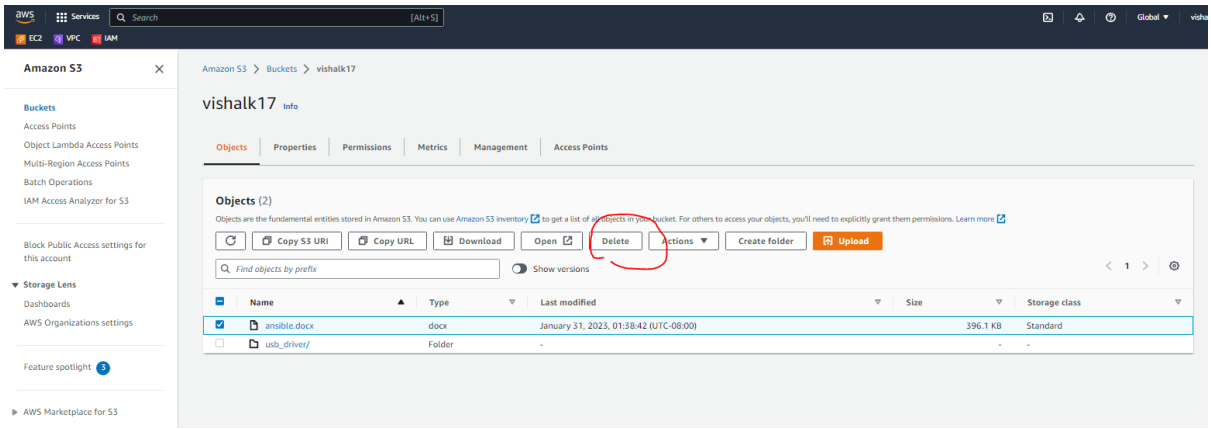


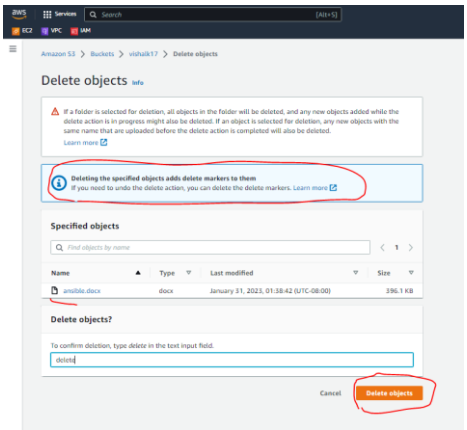
If I deleted latest version of ansible.docx then previous version of that will restored

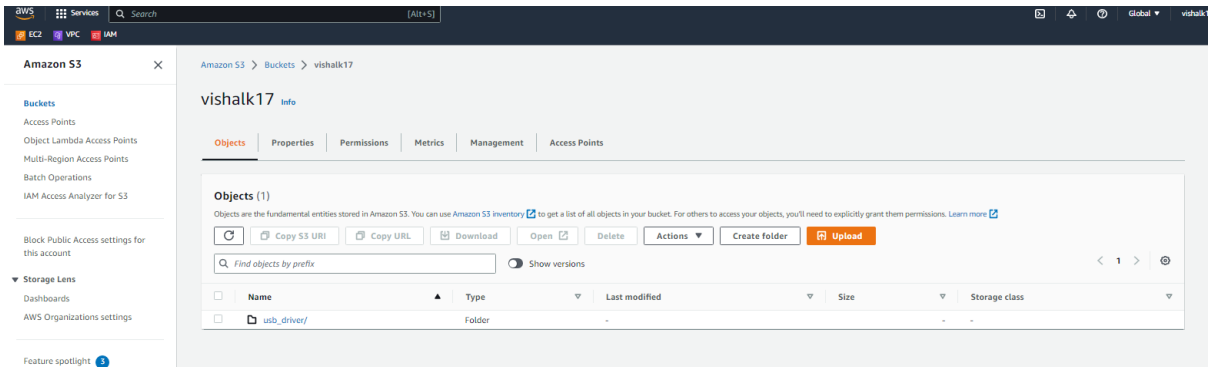
Chapter 4.0 : AWS S3 service

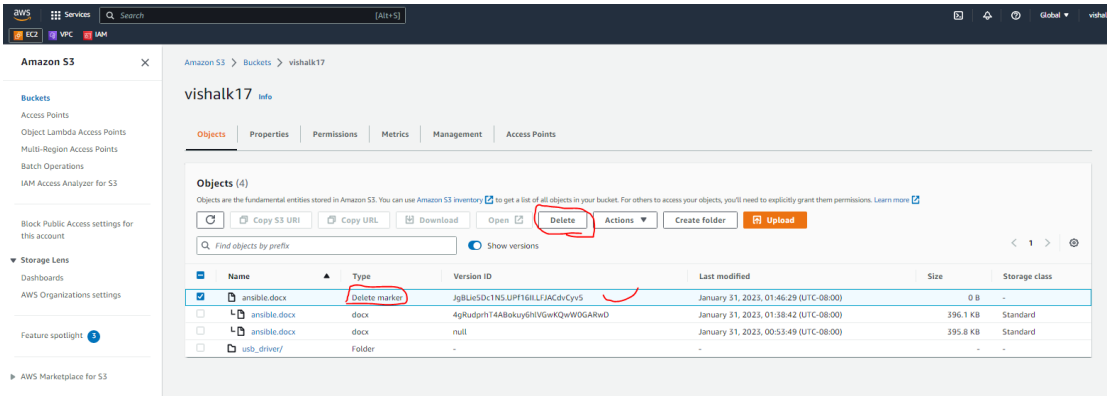
==========

A delete marker is a special version of an object that is created when a version of an object is deleted.



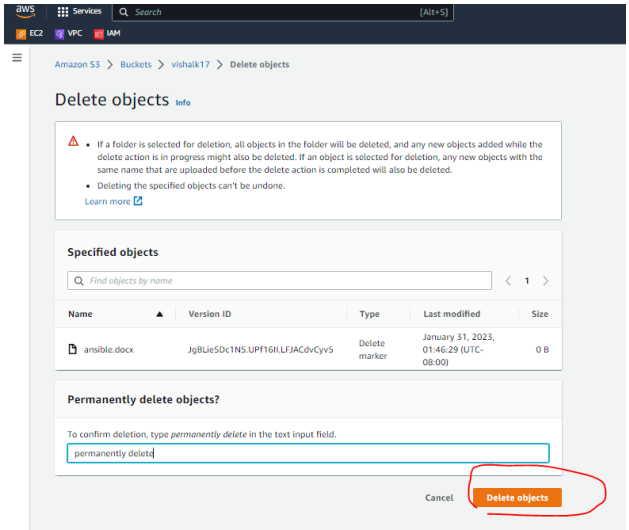


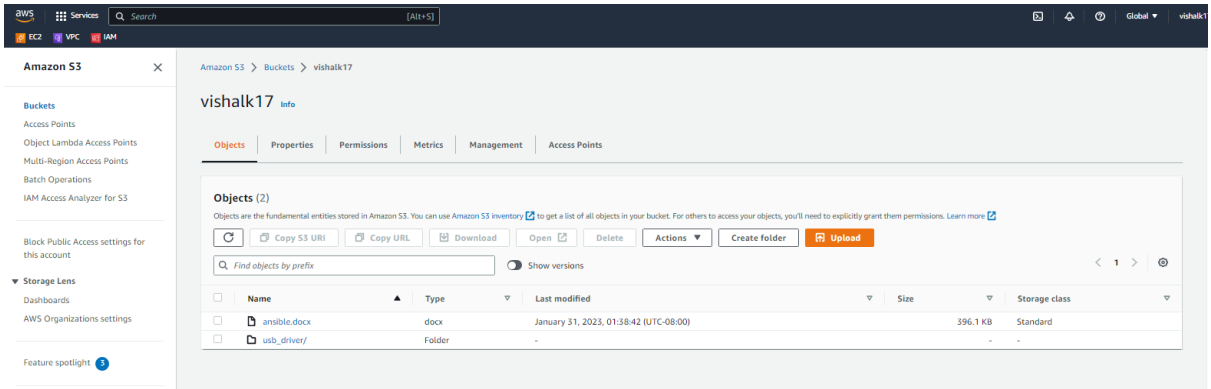




Chapter 4.0 : AWS S3 service

==========





Deleting the delete marker will restore the deleted object

Chapter 4.0 : AWS S3 service

==========

Lifecycle Management :

An **S3 Lifecycle Management** in simple terms when in an S3 bucket some data is stored for a longer time in standard storage even when not needed. The need to shift this old data to cheaper storage or delete it after a span of time gives rise to life cycle management.

Why is it needed?

Assume a lot of data is updated in an S3 bucket regularly, and if all the data is maintained by standard storage it will cost you more(even if previous data is of no use after some time). So, to avoid extra expenses and to maintain data as per requirement only life cycle management is needed.

There are 2 types of actions:

1. **Transition actions:** Moving objects from one storage class to another storage class. Each storage class has a different cost associated with it.
2. **Expiration actions:** When objects expire after a span of time (say 30 days,60 days, etc). Amazon S3 deletes expired objects on your behalf.

For example,

a transition lifecycle rule action can be set to automatically move Amazon S3 objects from the default S3 standard tier to Standard-IA (Infrequent Access) 30 days after they were created in order to reduce S3 storage costs.

The same rule can also be configured to archive the same objects after another three months by automatically moving them from S3 Standard-IA to Glacier or Glacier Deep Archive to further reduce storage costs.

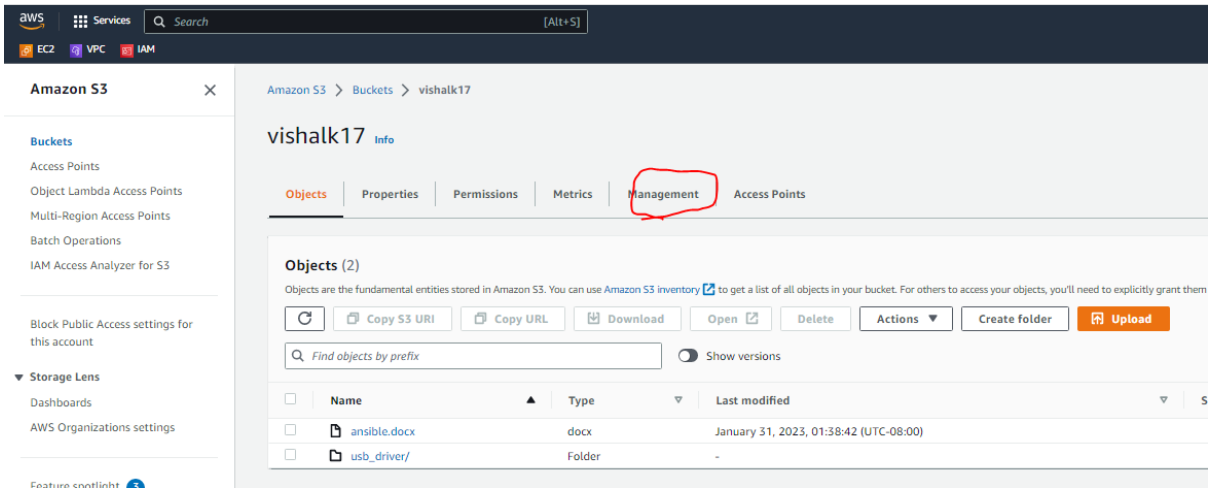
Here are the steps to create a lifecycle rule in an S3 bucket:

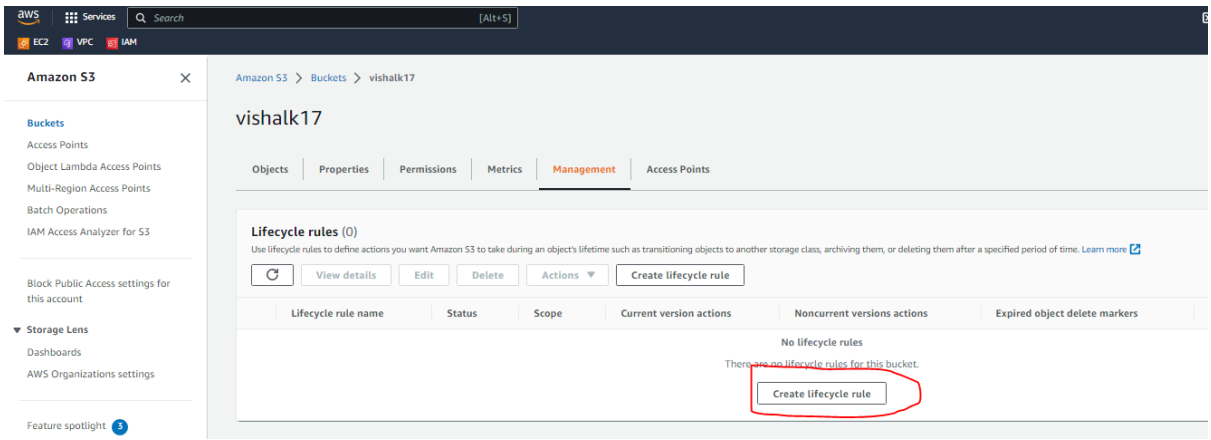
1. Sign in to the AWS Management Console and open the S3 service.
2. Select the bucket for which you want to create a lifecycle rule.
3. Click on the "Management" tab.
4. Click on "Add lifecycle rule" button.
5. Give the rule a name and choose the desired action (e.g. transition to Amazon S3 Standard-Infrequent Access or delete).
6. Define the transition/expiration rule, such as moving objects to another storage class after 30 days, or deleting objects after 90 days.
7. (Optional) You can also add conditions to the rule, such as only applying it to specific object prefixes or object tags.
8. Click "Save" to create the rule.

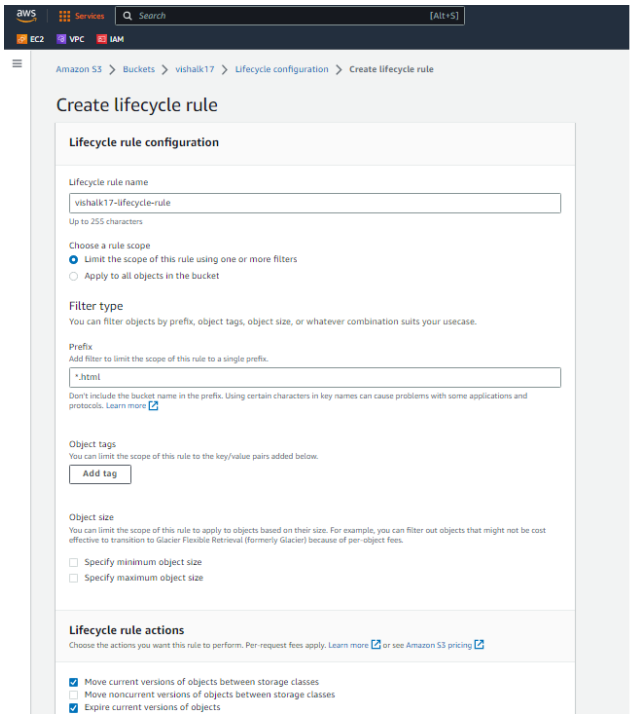
Note: Once the lifecycle rule is created, it will automatically be applied to all new and existing objects in the specified S3 bucket.

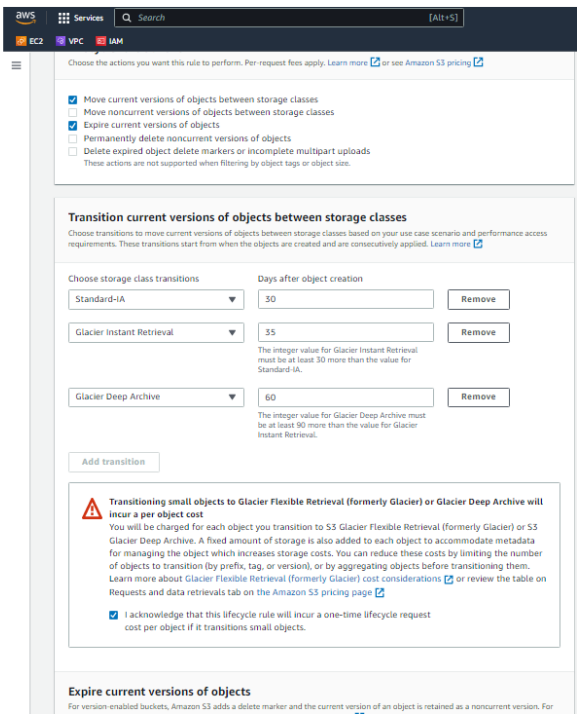
Chapter 4.0 : AWS S3 service

==========



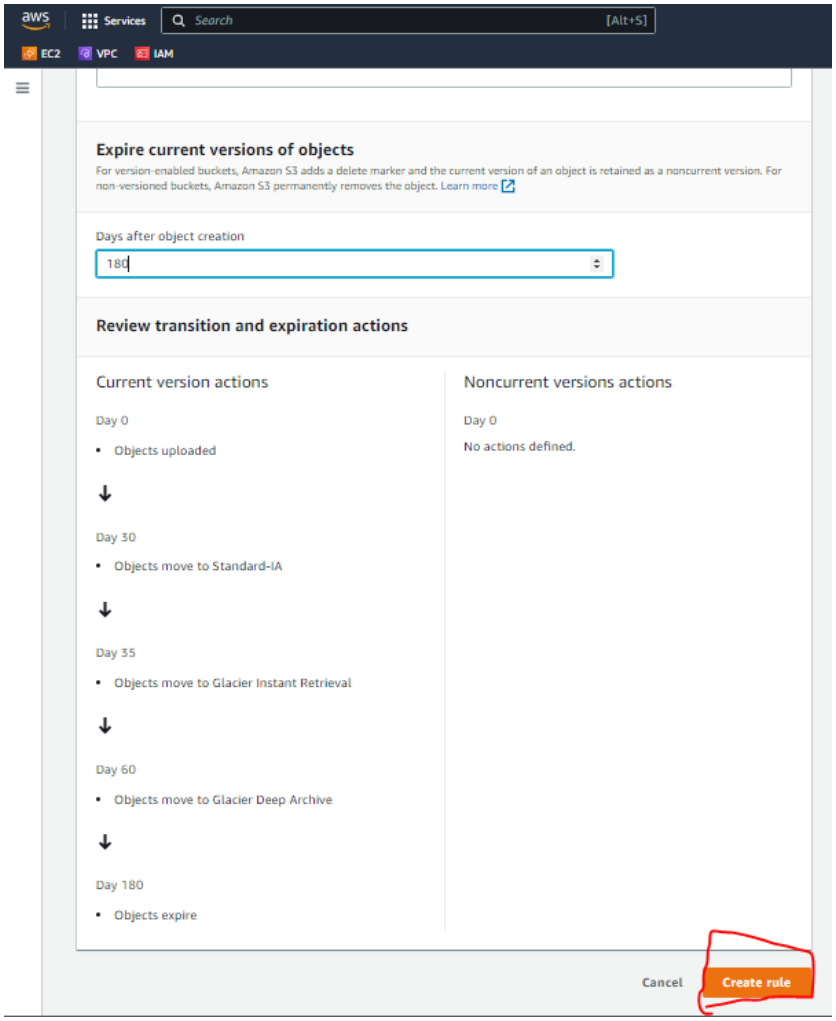






Chapter 4.0 : AWS S3 service

==========



Chapter 4.0 : AWS S3 service

==========

Cross Region Replication:

- Cross Region Replication is a feature that replicates the data from one bucket to another bucket which could be in a different region.
- It it is the Global service
- It provides asynchronous copying of objects across buckets. Suppose X is a source bucket and Y is a destination bucket. If X wants to copy its objects to Y bucket, then the objects are not copied immediately.

Some points to be remembered for Cross Region Replication

- **Create two buckets:** Create two buckets within AWS Management Console, where one bucket is a source bucket, and other is a destination bucket.
- **Enable versioning:** Cross Region Replication can be implemented only when the versioning of both the buckets is enabled.
- **Amazon S3 encrypts the data in transit across AWS regions using SSL:** It also provides security when data traverse across the different regions.
- **Already uploaded objects will not be replicated:** If any kind of data already exists in the bucket, then that data will not be replicated when you perform the cross region replication.
- **bi-directional CRR:** You can also enable bi-directional CRR by making the source bucket also the destination bucket for the destination bucket and vice versa.
- It is not necessary to have a destination bucket in the same account. AWS Cross-Region Replication can also be implemented in **cross accounts**

Use cases of Cross Region Replication

- **Compliance Requirements**
  By default, Amazon S3 stores the data across different geographical regions or availability zone to have the availability of data. Sometimes there could be compliance requirements that you want to store the data in some specific region. Cross Region Replication allows you to replicate the data at some specific region to satisfy the requirements.
- **Improved data access:** With CRR, data is stored in multiple regions, making it accessible from different locations with **low latency**
- **Maintain object copies under different ownership:** Regardless of who owns the source bucket, you can tell to Amazon S3 to change the ownership to AWS account user that owns the destination bucket. This is referred to as an owner override option.
- **Disaster recovery:** Having your data in more than one region will help you prepare and handle data loss due to some unprecedented circumstances.

To achieve Cross-Region Replication (CRR) in Amazon S3, follow these steps:

1. **Set up your S3 buckets:** Create two S3 buckets, one as the source bucket and one as the destination bucket. The destination bucket must be in a different region than the source bucket.
2. **Configure source bucket replication:** In the source bucket, navigate to the "Replication" section of the bucket properties and select "Add rule". In the replication rule, specify the destination bucket and the desired replication settings.
3. **Set up the destination bucket:** Ensure that the destination bucket has the proper permissions for the replication. The AWS account that owns the source bucket must have permission to write to the destination bucket.
4. **Enable versioning:** Ensure that versioning is enabled in both the source and destination buckets. CRR only replicates objects that have versioning enabled.

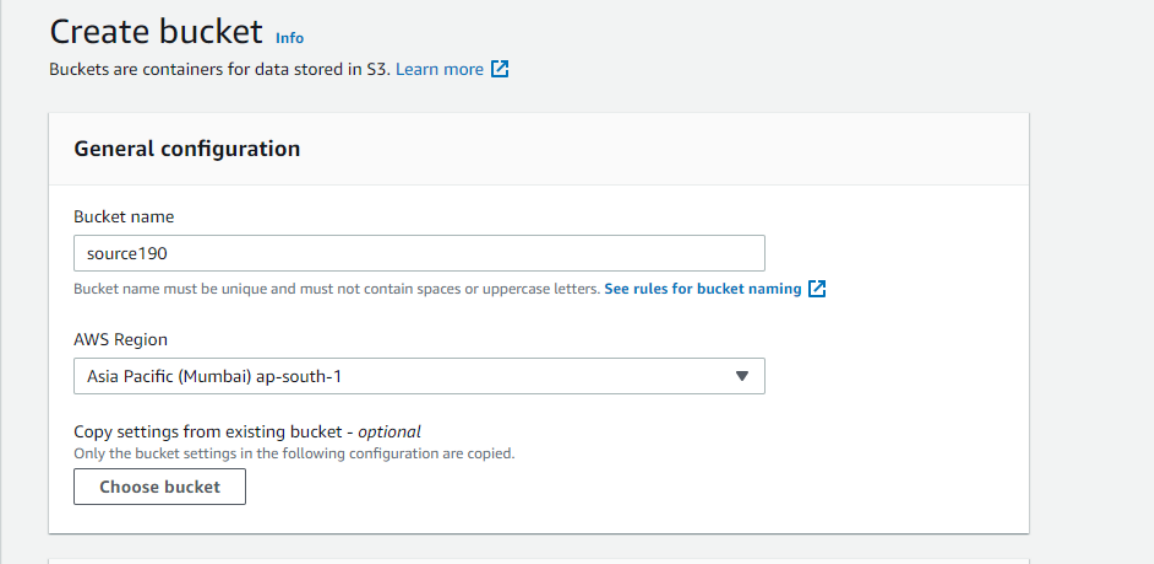
==========

5. **Start replication:** Once the replication rule is set up, S3 will automatically start replicating objects from the source bucket to the destination bucket. Only objects added after the replication rule is set up will be replicated.

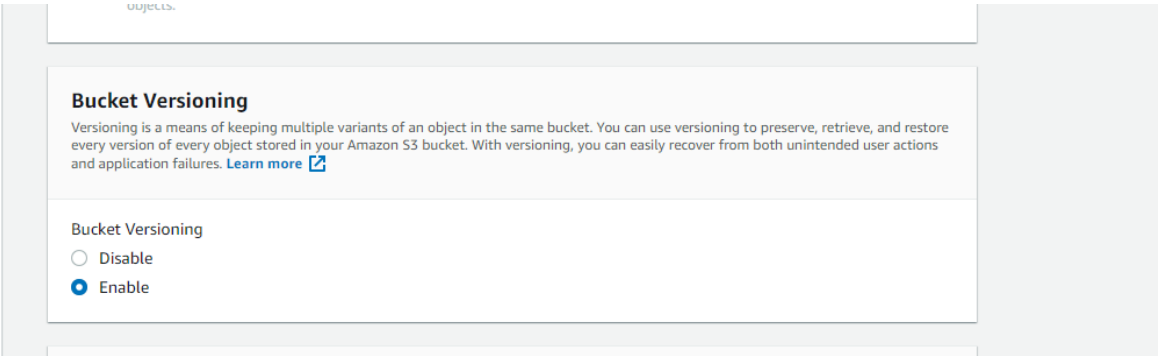
Setting up CRR:

Follow the below steps to set up the CRR:

- Go to the AWS s3 console and create two buckets.
- Let's name our source bucket as source190 and keep it in the Asia Pacific (Mumbai) ap-south 1 region. Do not forget to **enable versioning**. Also, note that the S3 bucket name needs to be globally unique and hence try adding random numbers after bucket name.



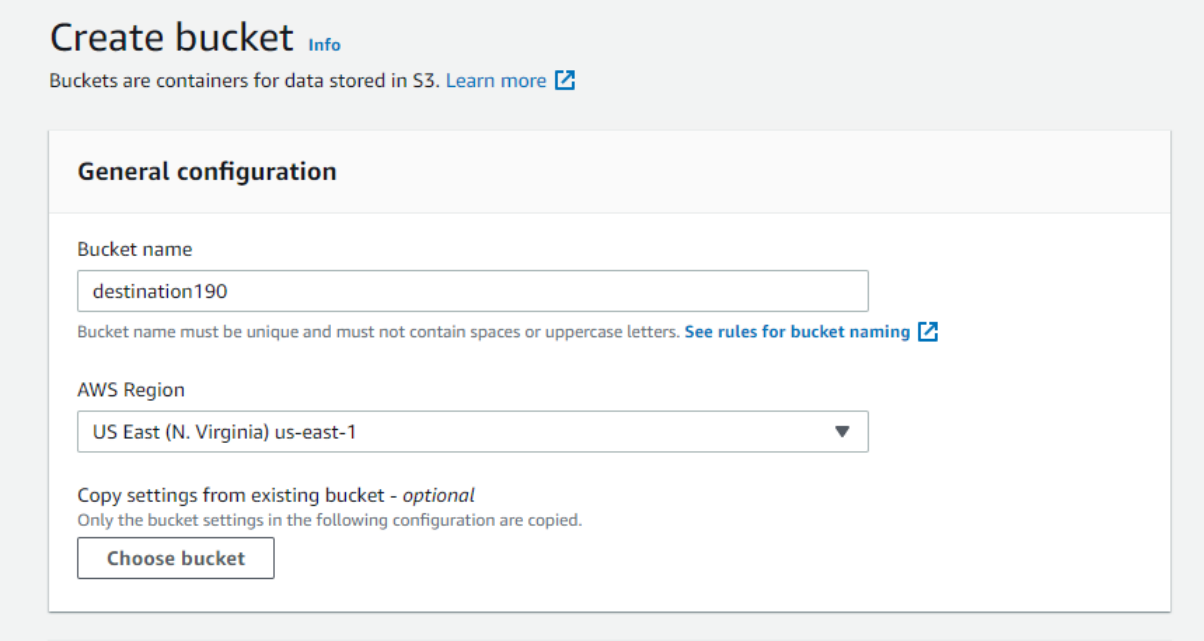
.



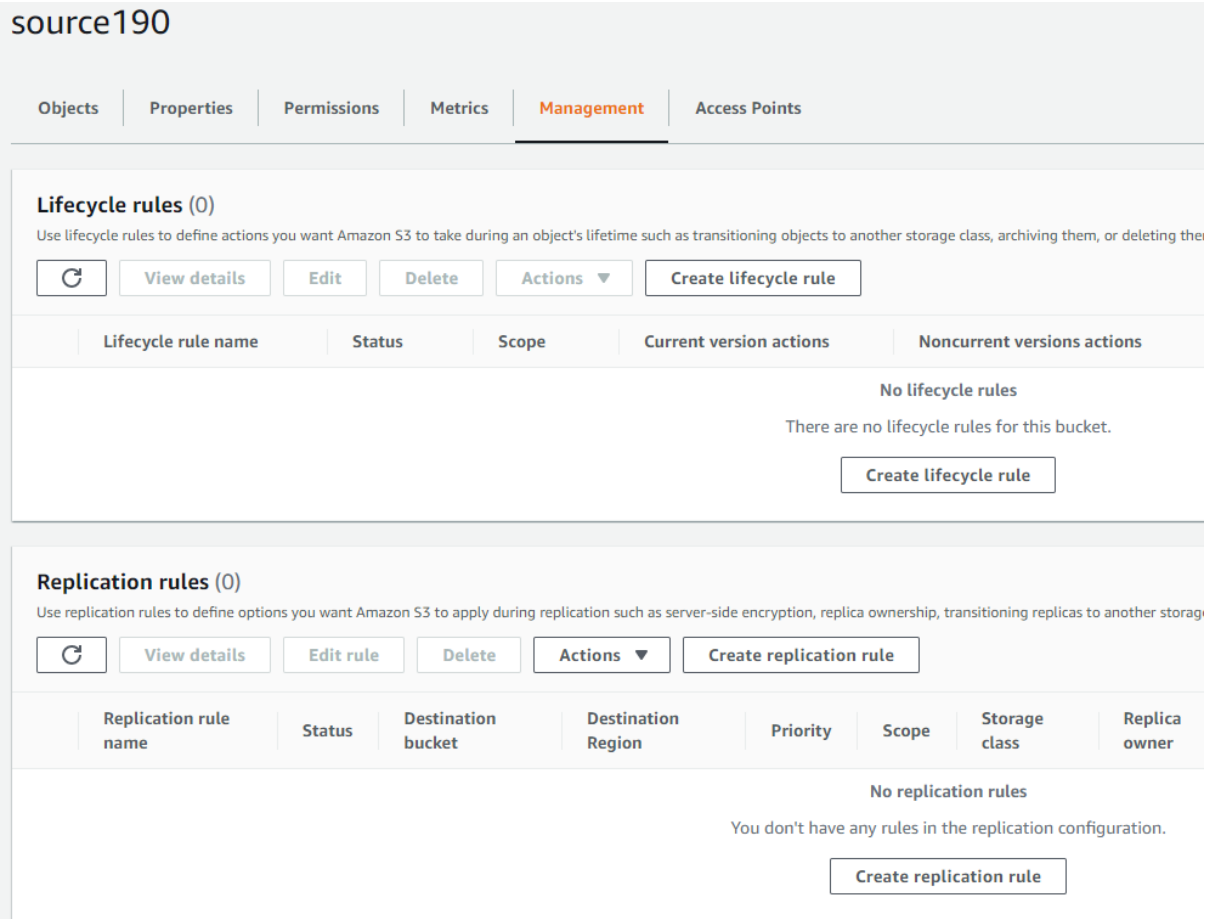
- Now following the same steps create a destination bucket: destination190 with versioning enabled but choose a different region this time.

Chapter 4.0 : AWS S3 service

==========



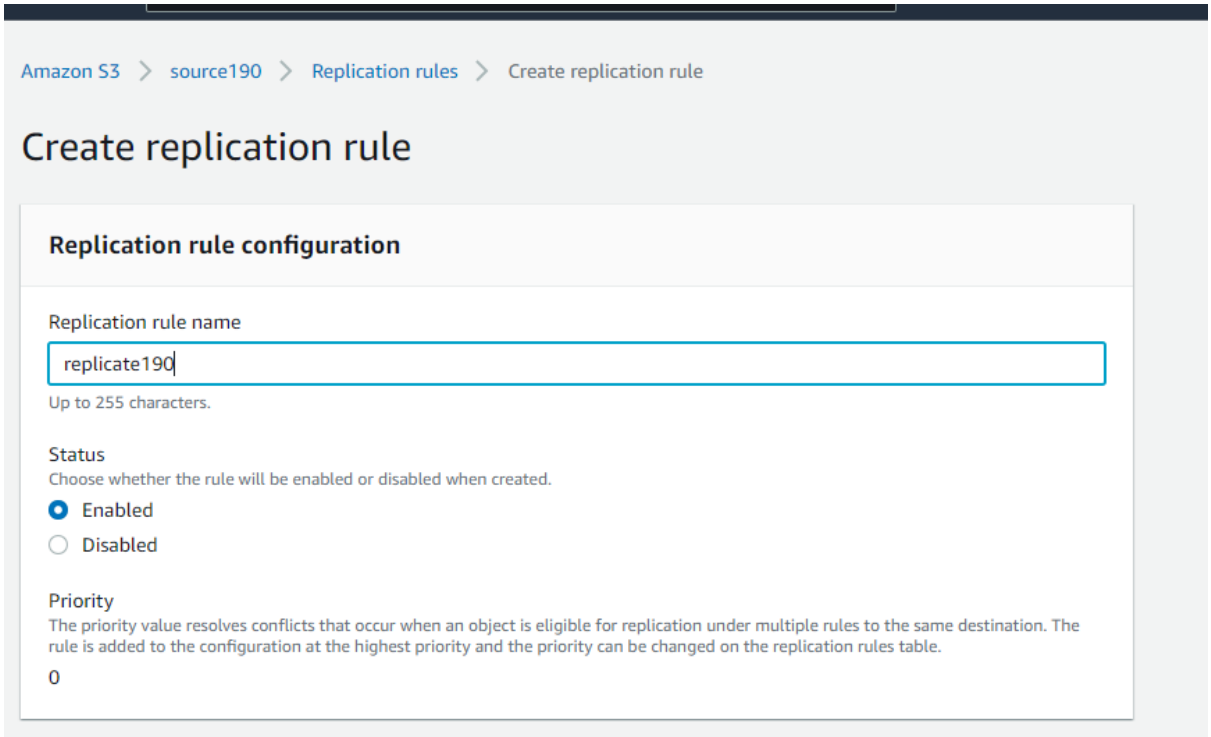
- Now click on your source bucket and head over to the management tab:



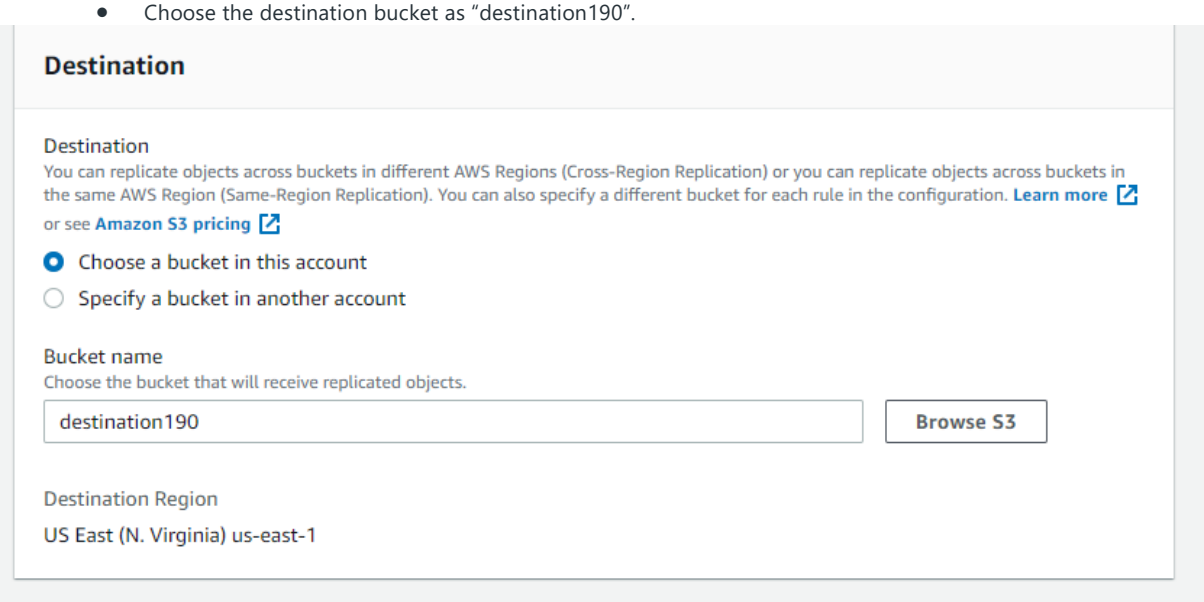
- Now, click on “Create a replication rule” and give your replication rule a name as “ replicate190”

Chapter 4.0 : AWS S3 service

==========



- Choose the destination bucket as “destination190”.



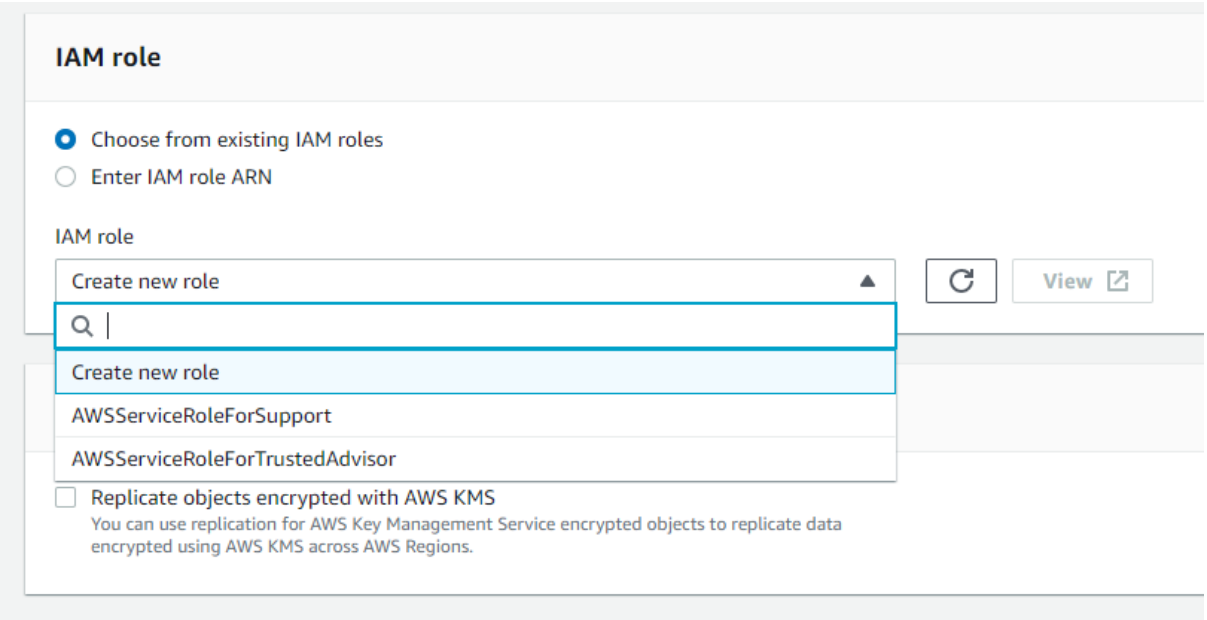
Set destination bucket

Notice that you have an option to choose a destination bucket in another account.

- In order to replicate objects from the source bucket to the destination bucket, you need to create an IAM role. So just create one by clicking on “create a new role”.

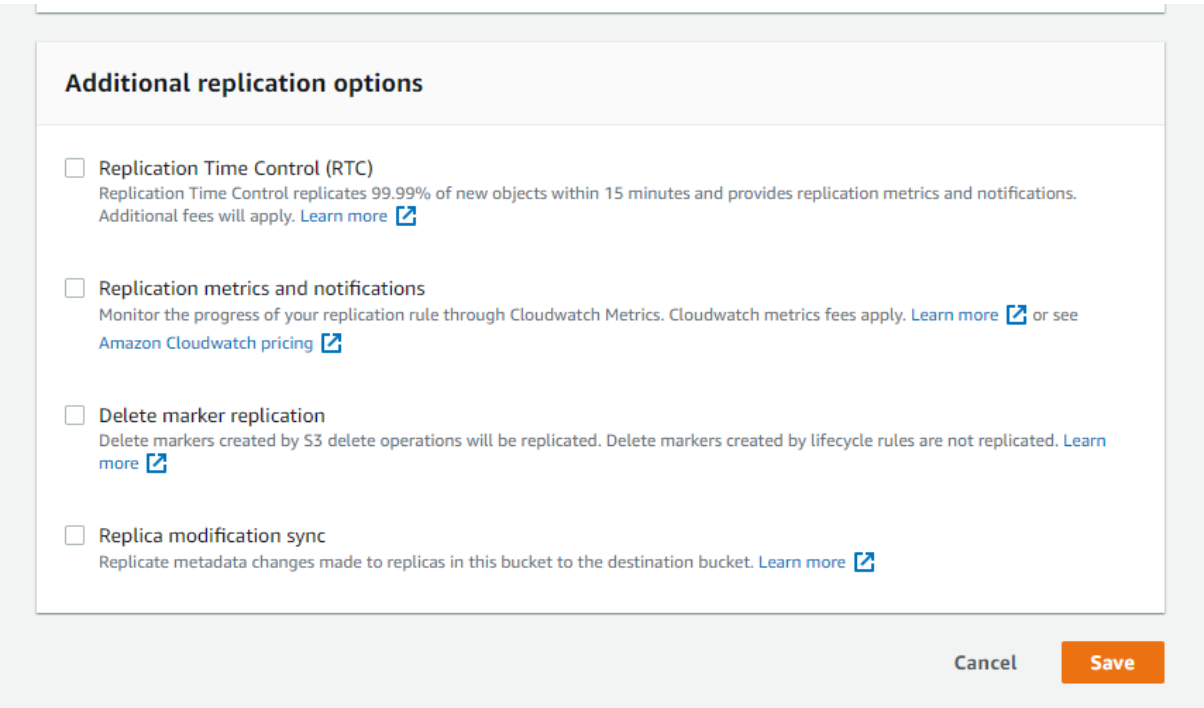
Chapter 4.0 : AWS S3 service

==========



Create IAM role

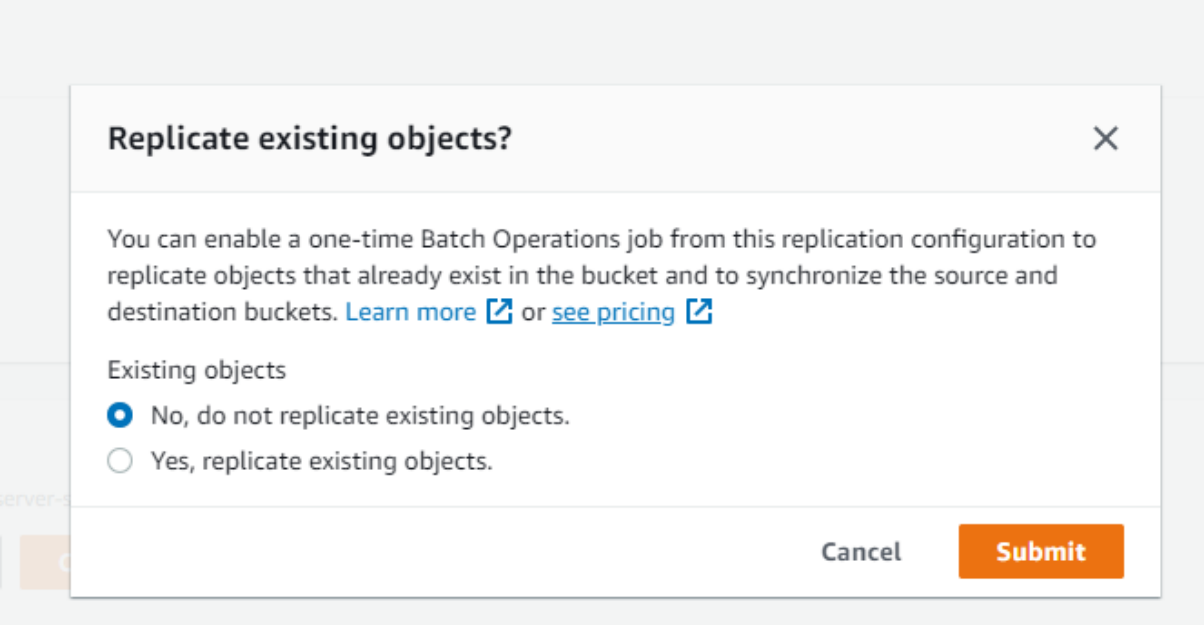
- If you want your S3 objects to be replicated within 15 minutes you need to check the "Replication Time Control (RTC) box. But you will be charged for this. So we will move forward without enabling that for now and click on save.



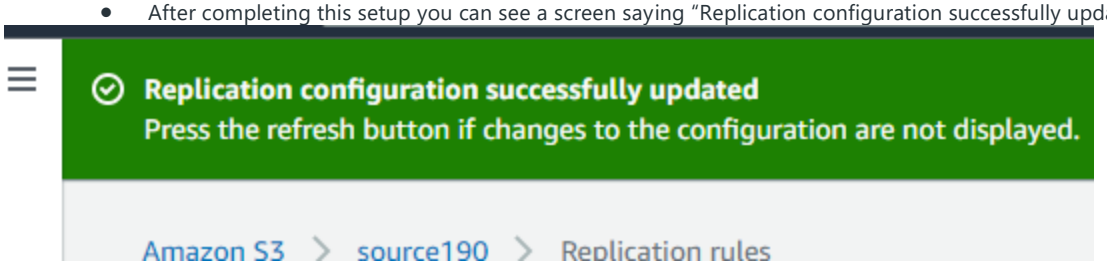
As soon as you click on save, a screen will pop up asking if you want to replicate existing objects in the S3 bucket. But that will incur charges so we will proceed without replicating existing objects and click on submit.

Chapter 4.0 : AWS S3 service

==========



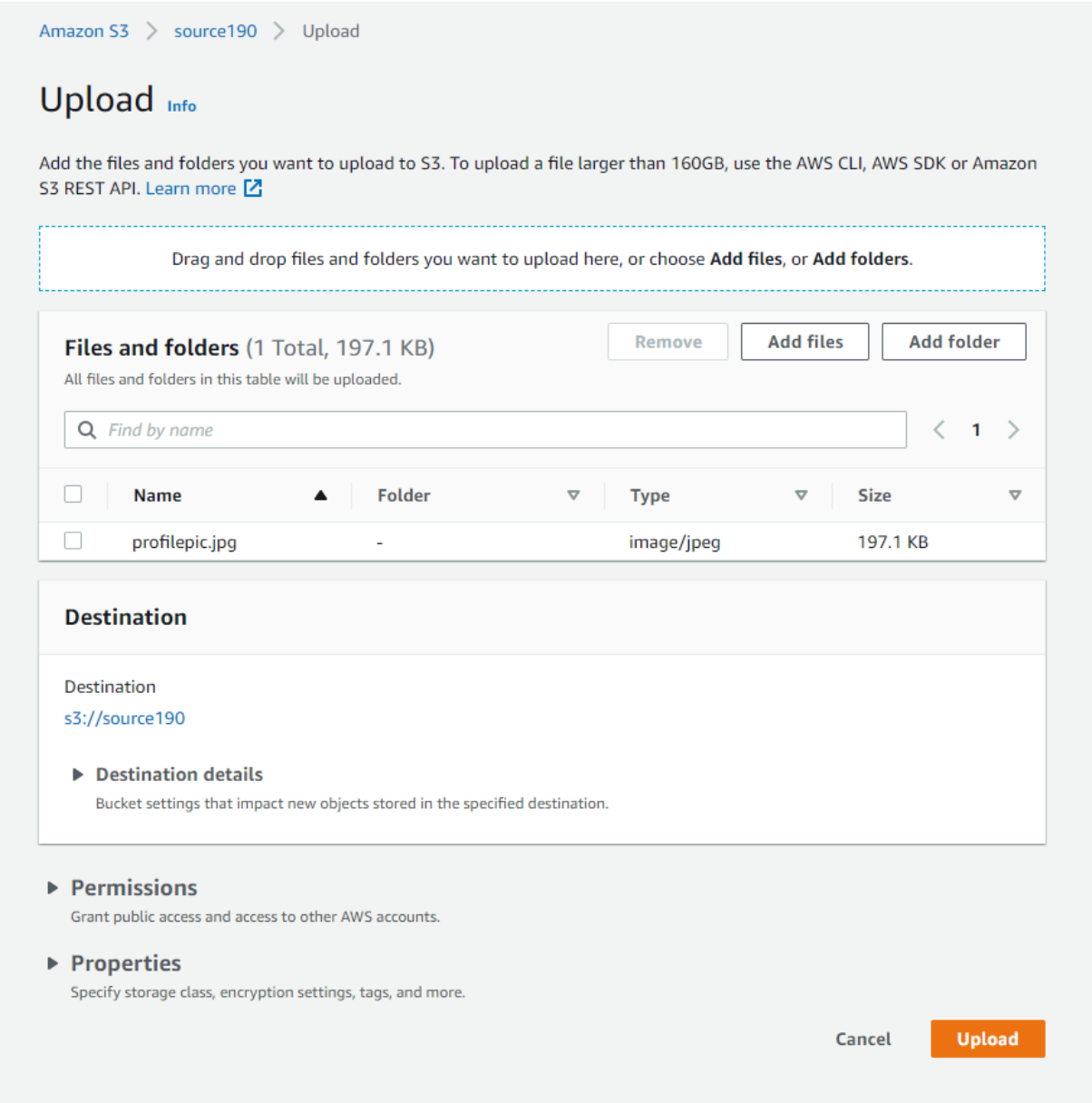
- After completing this setup you can see a screen saying “Replication configuration successfully updated”.



It’s time to test! Now go to the source bucket: source190 and upload a file.

Chapter 4.0 : AWS S3 service

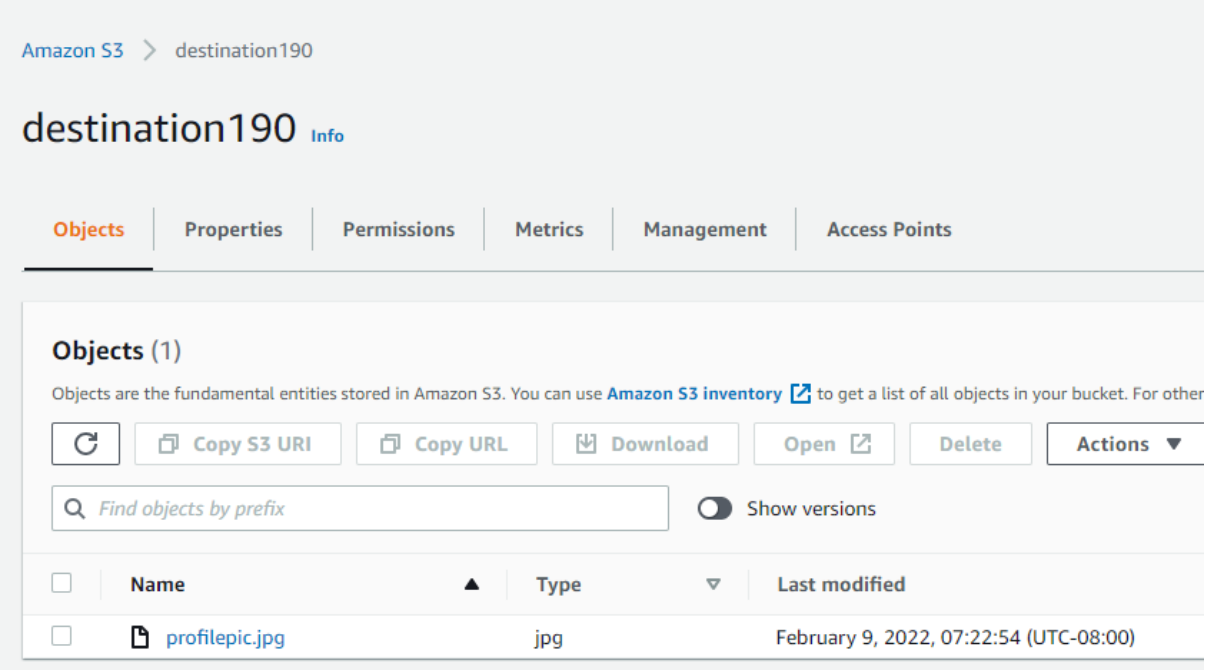
==========



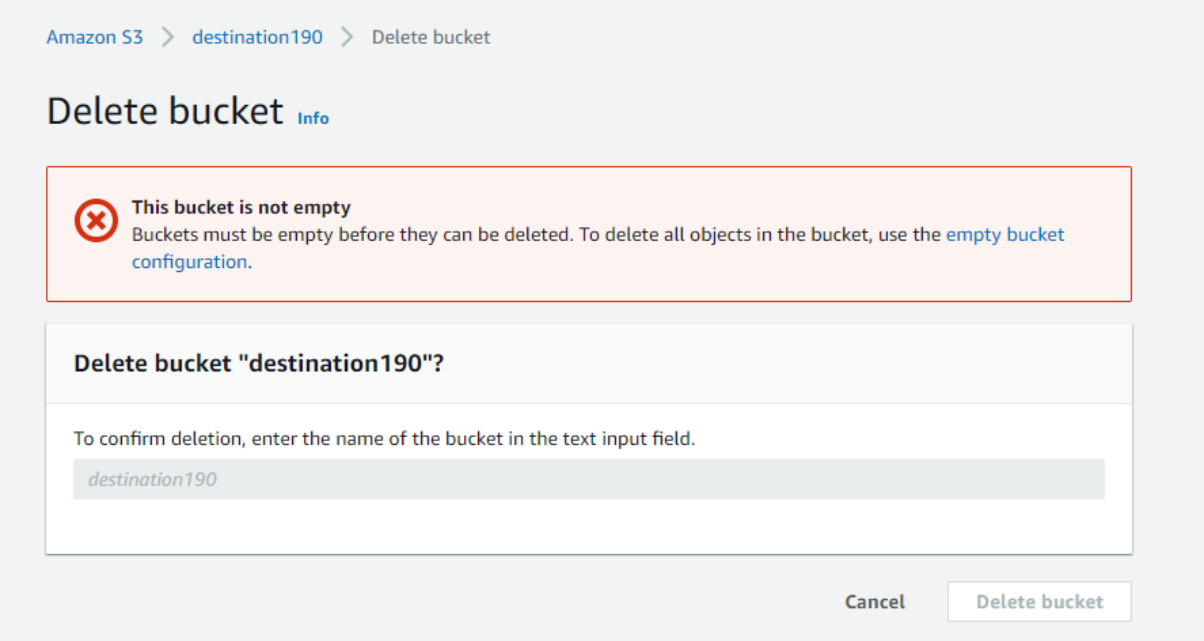
Now head over to our destination bucket: destination190 to check if the uploaded file is replicated to our destination bucket. You can see that our uploaded file is successfully copied to the destination bucket:

Chapter 4.0 : AWS S3 service

==========



Note: Do not forget to empty your buckets and then delete them, if you do not have any further use. Also, you cannot delete a bucket if it is not empty.



Chapter 4.0 : AWS S3 service

==========

Static Website hosting:

Amazon S3 can be used to host static websites, where the content of the website is stored as individual files in an S3 bucket. Here are some important points and notes on S3 static website hosting:

Advantages:

- **Cost-effective:** S3 is a cost-effective solution for hosting static websites, as you only pay for the storage and data transfer costs associated with serving your website.
- **Scalability:** S3 can handle a large number of requests and can scale to accommodate growing traffic to your website.
- **High availability:** S3 provides high availability for your website, ensuring that it is always accessible to your users.

Disadvantages:

- **Limited functionality:** S3 only supports hosting of static websites, and does not support dynamic content or server-side scripting.
- **Performance limitations:** S3 is optimized for serving large files, but may not perform as well for small, frequently-requested objects.

Limitations:

- **No server-side processing:** S3 only supports serving of static files, and does not support server-side processing or dynamic content.
- **No custom domains:** You cannot host a website at a custom domain name with S3. Instead, you must use the S3 website endpoint provided by Amazon.
- **No email services:** S3 does not provide email services for a hosted website.

To achieve S3 static website hosting, follow these steps:

1. **Create an S3 bucket:** Create an S3 bucket to store the files for your website.
2. **Configure the bucket for website hosting:** In the bucket properties, navigate to the "Static website hosting" section and select "Use this bucket to host a website".
3. **Upload your website files:** Upload the files for your website to the S3 bucket.
4. **Configure the website endpoint:** S3 provides a website endpoint for your hosted website. You can access your website using the endpoint provided by Amazon.
5. **Test your website:** Verify that your website is accessible using the S3 website endpoint.

Note: S3 charges for storage and data transfer for serving your website, so it's important to monitor your costs and consider the pricing structure when hosting a website with S3.

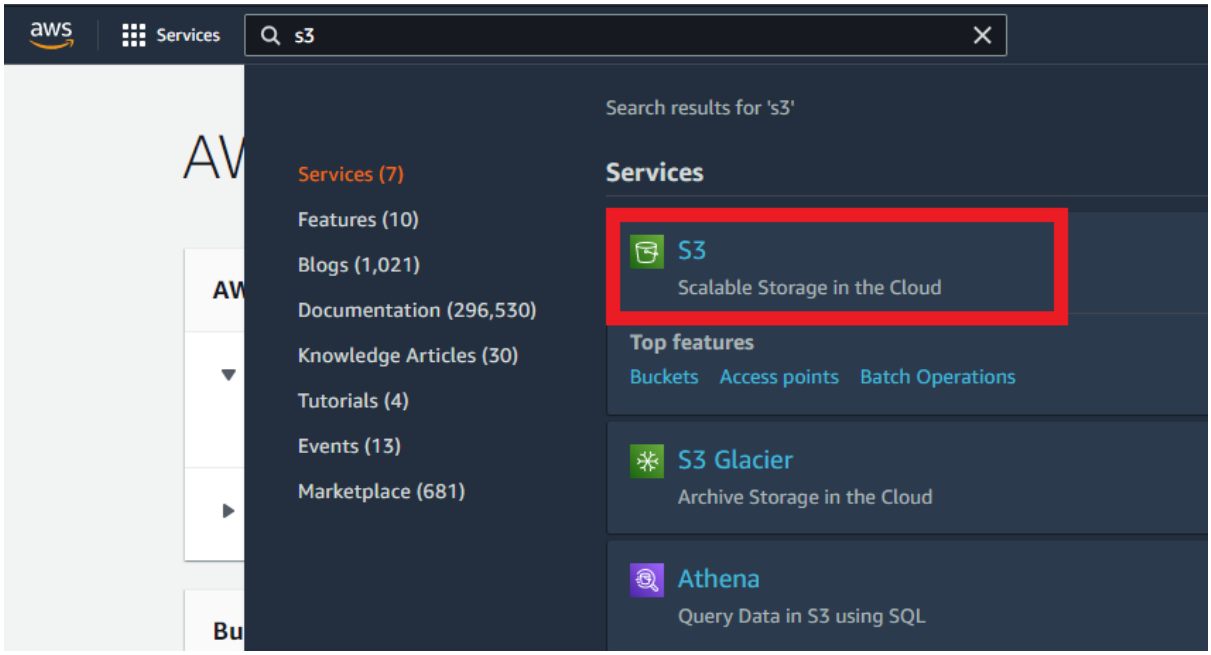
Chapter 4.0 : AWS S3 service

==========

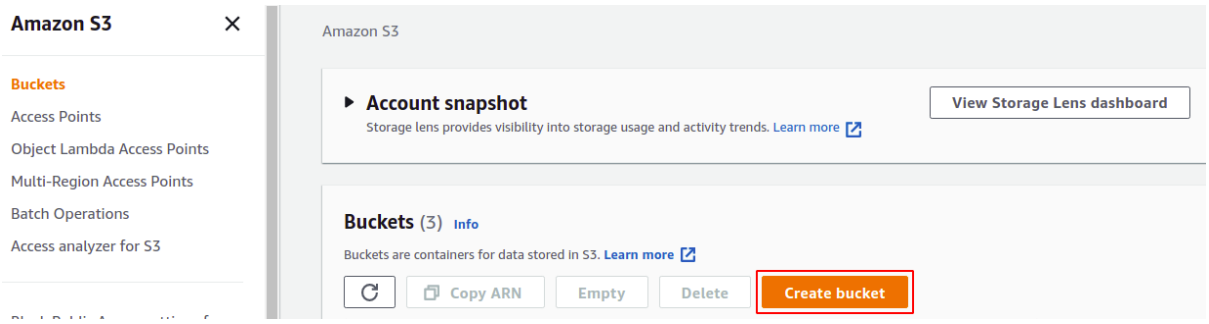
Create an S3 bucket on AWS

The first step to hosting a static website on AWS S3 is to create an S3 bucket in your account. After creating the bucket, we will upload the website contents and files in our bucket. The website content will then be assigned specific permissions to be accessible to the public.

Login to your AWS management console and go to the search bar and search for **S3** there. This will lead you to your S3 dashboard:



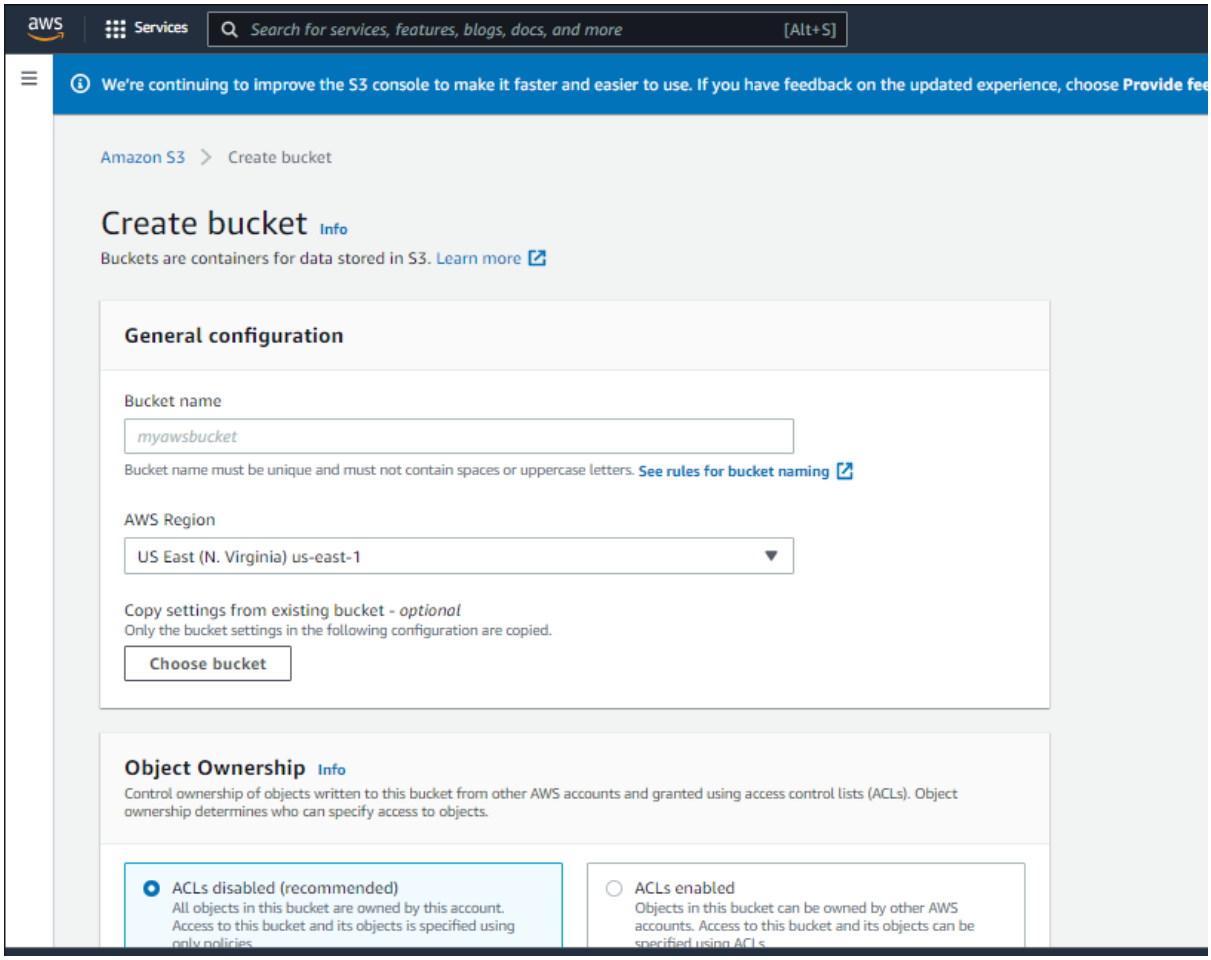
Click on Create Bucket at the right corner of the S3 console:



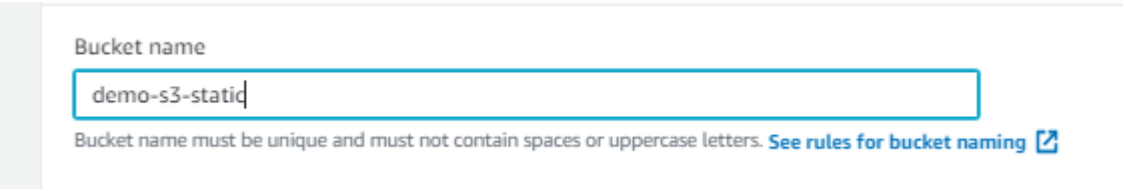
Next, you need to provide your S3 bucket name, the region where you want your bucket to be created, and then configure your bucket's security and privacy setting:

Chapter 4.0 : AWS S3 service

==========



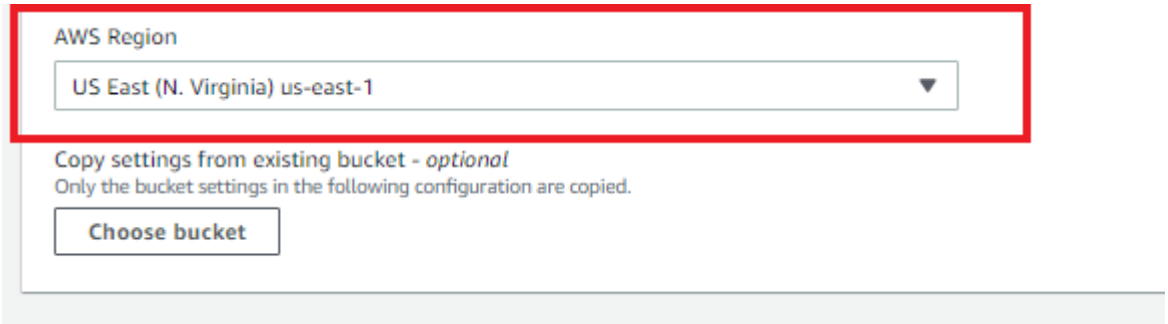
Enter Bucket Name, try to make it look like your domain. The bucket name should be unique for all AWS accounts around the world:



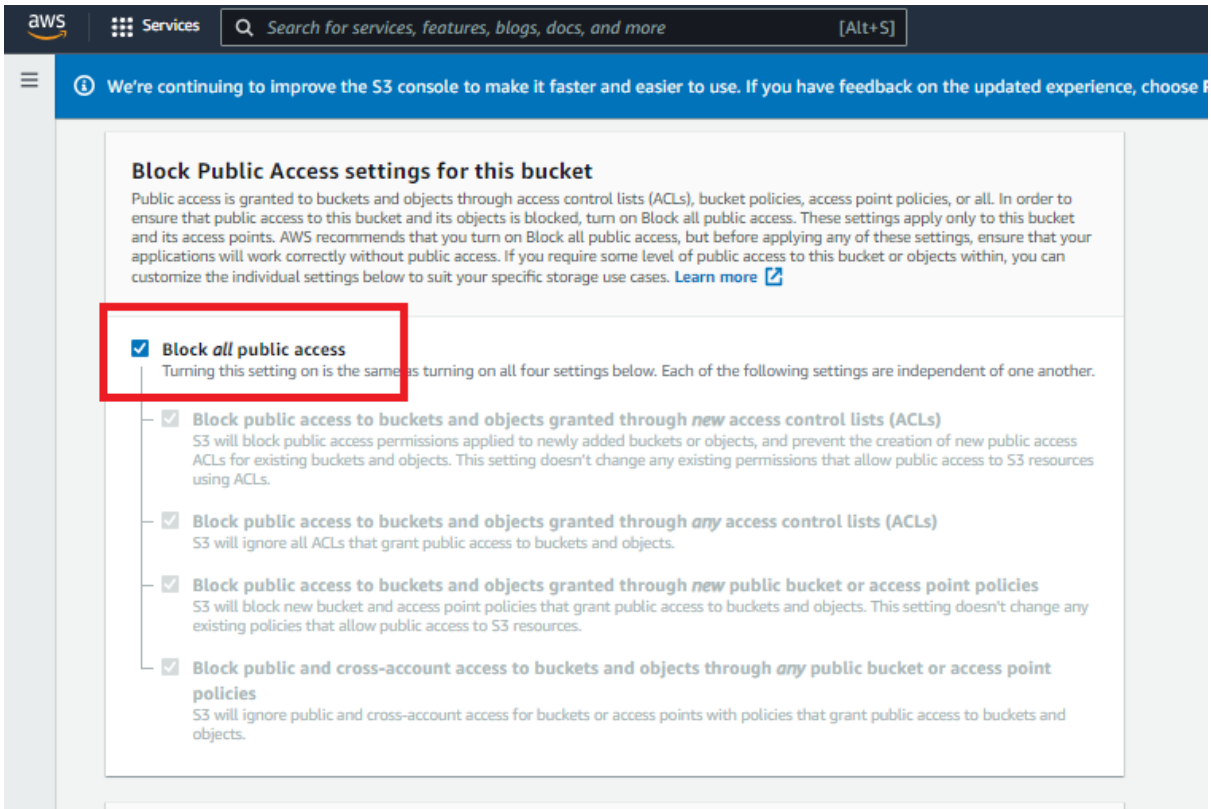
Select the region in which the S3 bucket will be created. Try to select a region near the public that will access the website:

Chapter 4.0 : AWS S3 service

==========



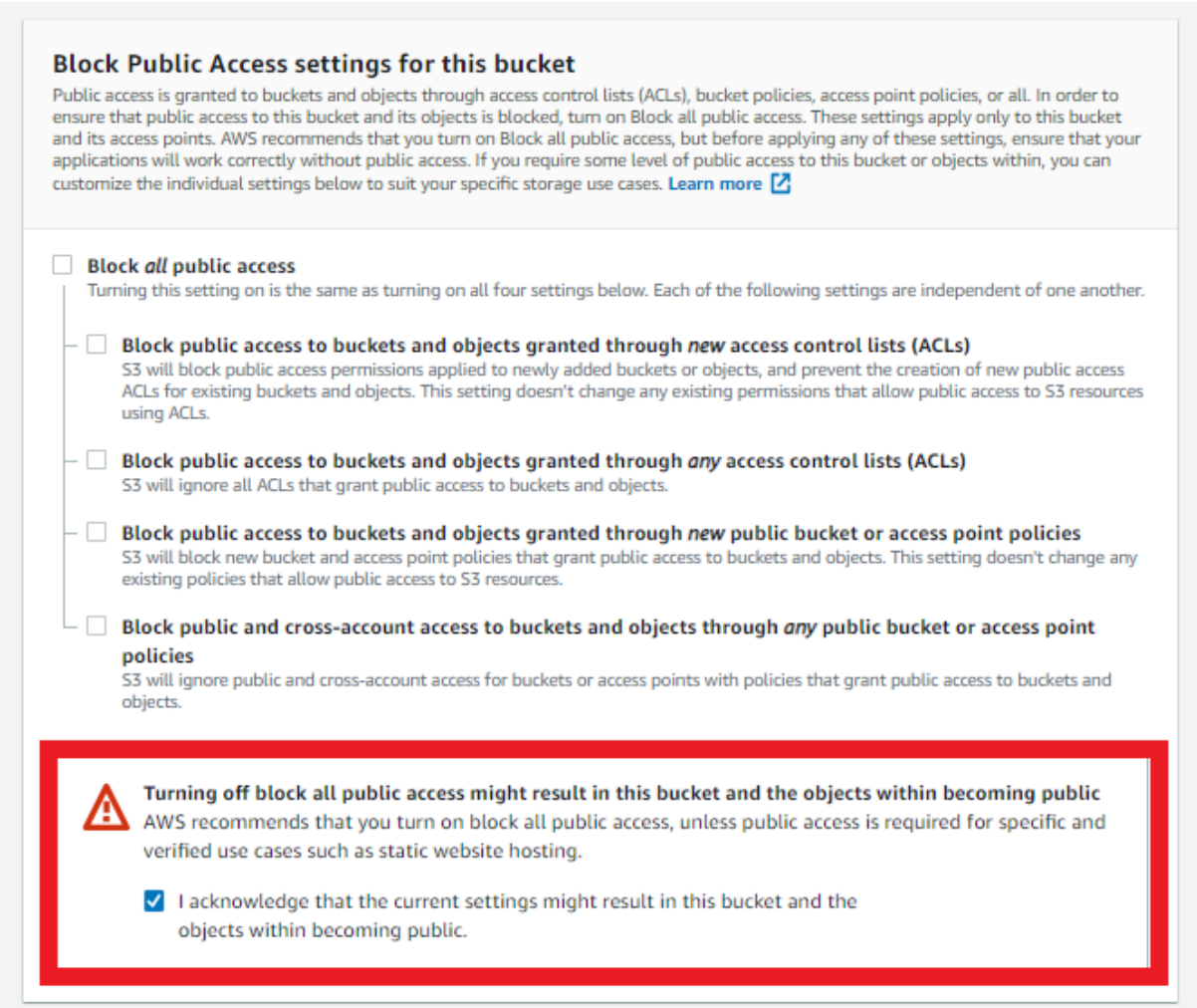
Since we wanted the website to be accessible to the audience, we had to grant the public access to the objects of this S3 bucket. For that, uncheck the Block all public access checkbox in the "Block Public Access setting for this bucket" section:



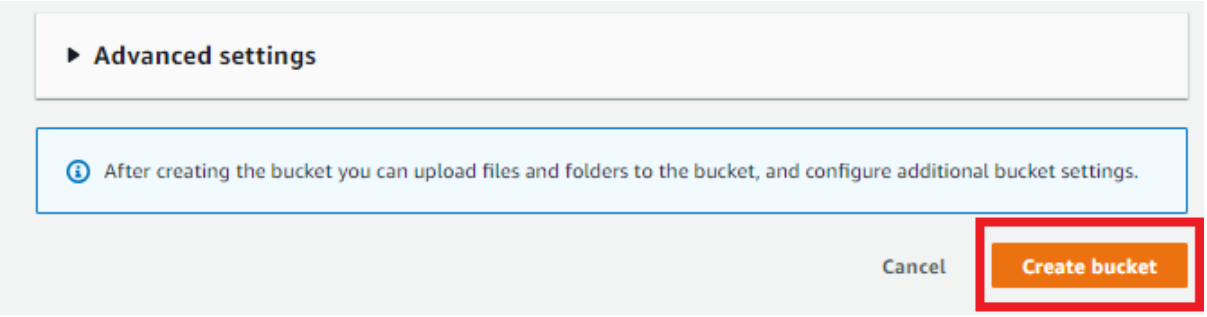
After configuring the public access settings, a section will appear to acknowledge the S3 bucket and its content being made public. Check the box to acknowledge it:

Chapter 4.0 : AWS S3 service

==========



Now, you have finished setting up your bucket, leave other options and settings as it is, and just click on the **Create Bucket** button at the bottom right corner:



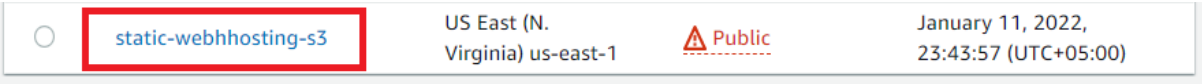
If the bucket name you specified is unique, the S3 bucket will be created. Otherwise, you will get an error, and you have to change the bucket name.

Chapter 4.0 : AWS S3 service

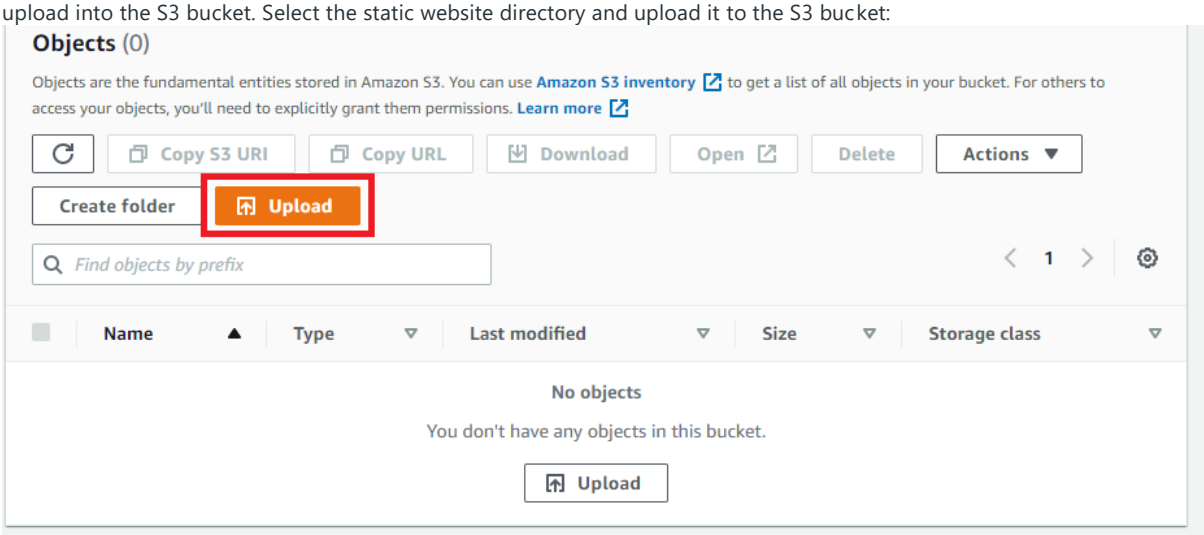
==========

Upload Your Website to the S3 Bucket

After creating the S3 bucket, it is time to upload website content to the S3 bucket. From the S3 console, select the S3 bucket you just created:



Go to the **Objects** section, and then Click on the upload button. Now, browse your system for the directory you want to upload into the S3 bucket. Select the static website directory and upload it to the S3 bucket:



Uploading the static site content may take some time depending upon the size of the folder:

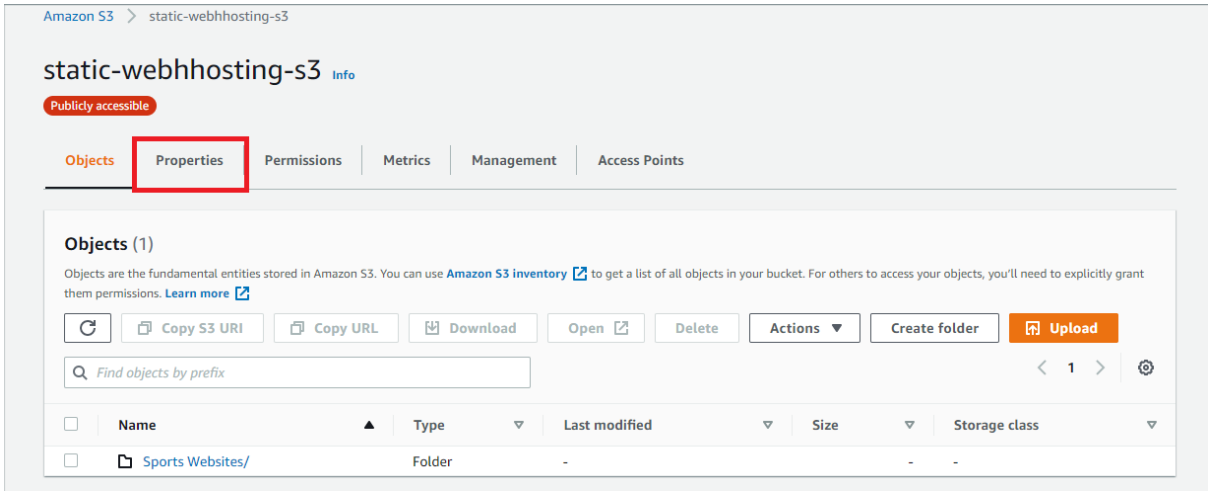
After a successful upload, click close at the right corner. You will be directed back to the object section.

Setting up Static Web Hosting in S3 Bucket

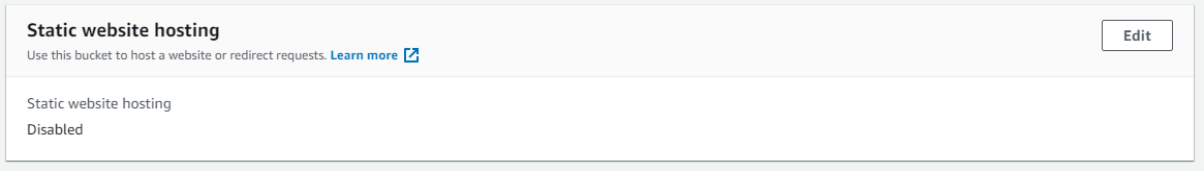
After uploading the static site content, enable hosting on your S3 bucket. In order to allow static website hosting on your S3 bucket, go to the properties tab from the top menu in the S3 bucket:

Chapter 4.0 : AWS S3 service

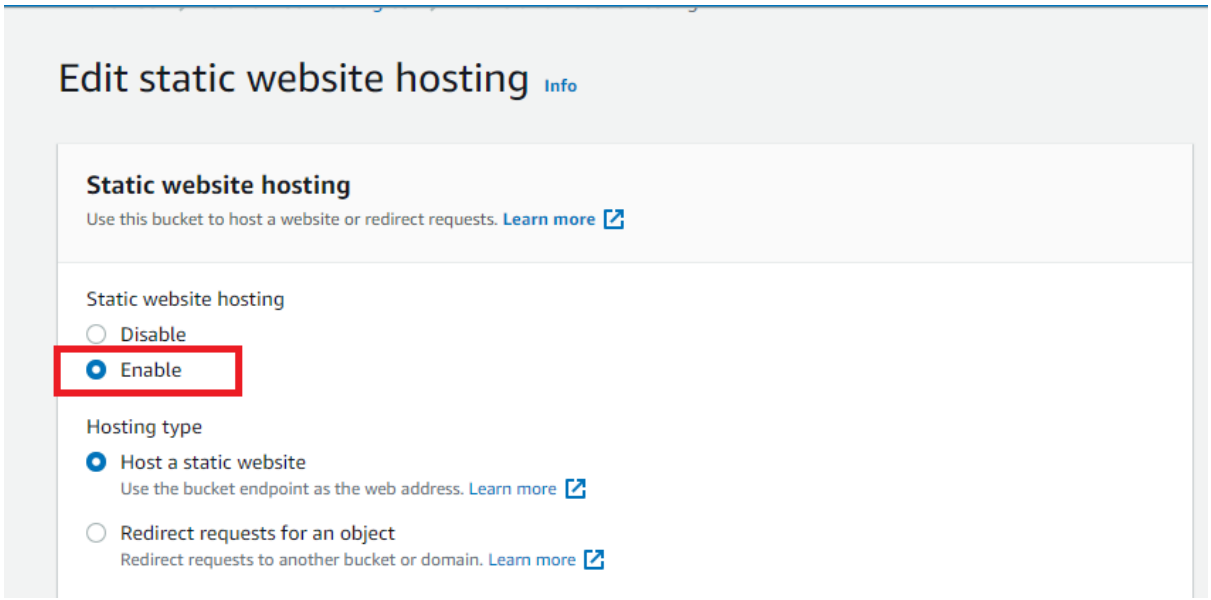
==========



Scroll down in properties tab and look for the Static Website Hosting section:



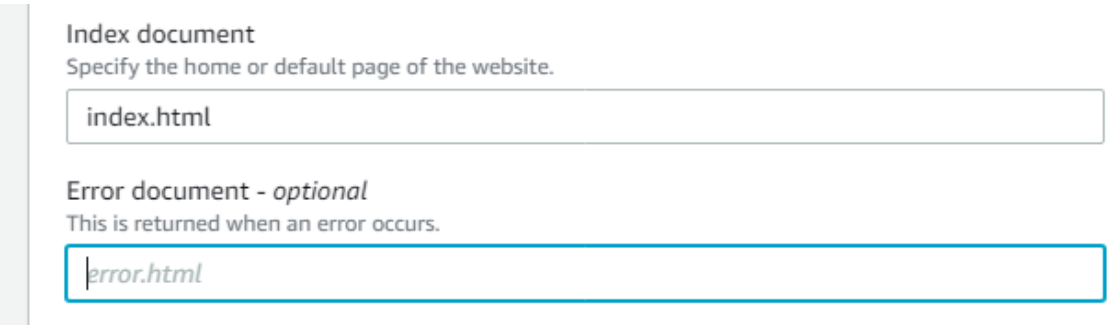
Click on the Edit button in the Static website hosting section and enable the hosting:



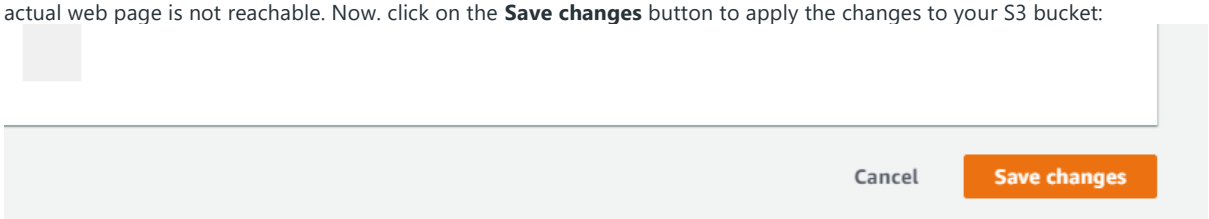
After enabling static website hosting, specify the index file of your project (the opening page of your website or web application). In this case, it is index.html:

Chapter 4.0 : AWS S3 service

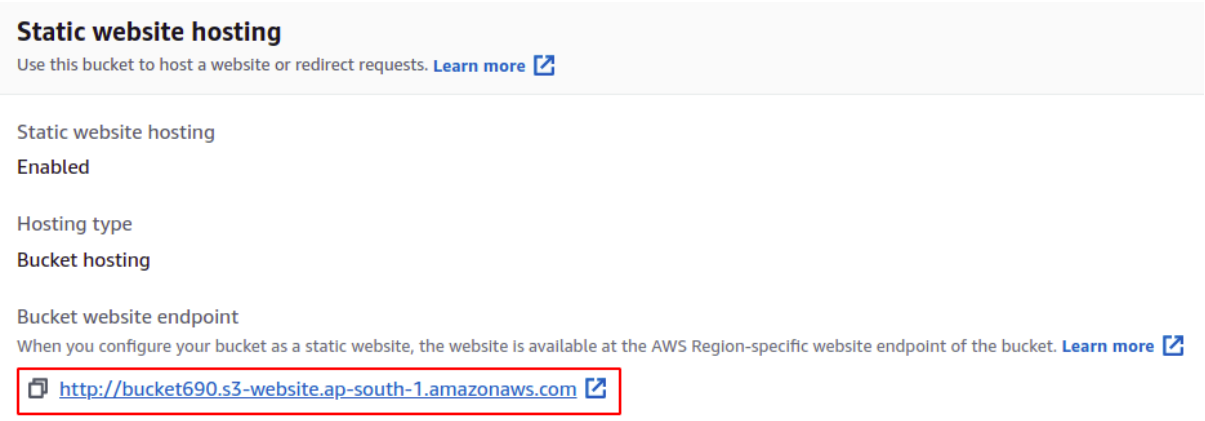
==========



Also, if there is an error file in your project, you must specify it in the error document field. This will appear in case your actual web page is not reachable. Now. click on the **Save changes** button to apply the changes to your S3 bucket:



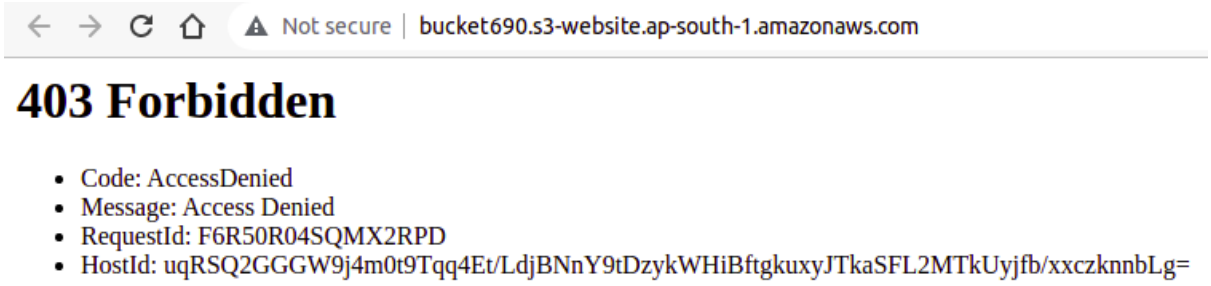
Now, our S3 bucket is hosting the website content uploaded to it and is publicly accessible. In order to access the website, we need a public URL that AWS itself provides. This URL can be seen in the static website hosting section of the S3 bucket:



Go to the URL provided by S3, and the website will not be accessible because we have made the S3 bucket public, but the objects inside the S3 bucket are not public yet:

Chapter 4.0 : AWS S3 service

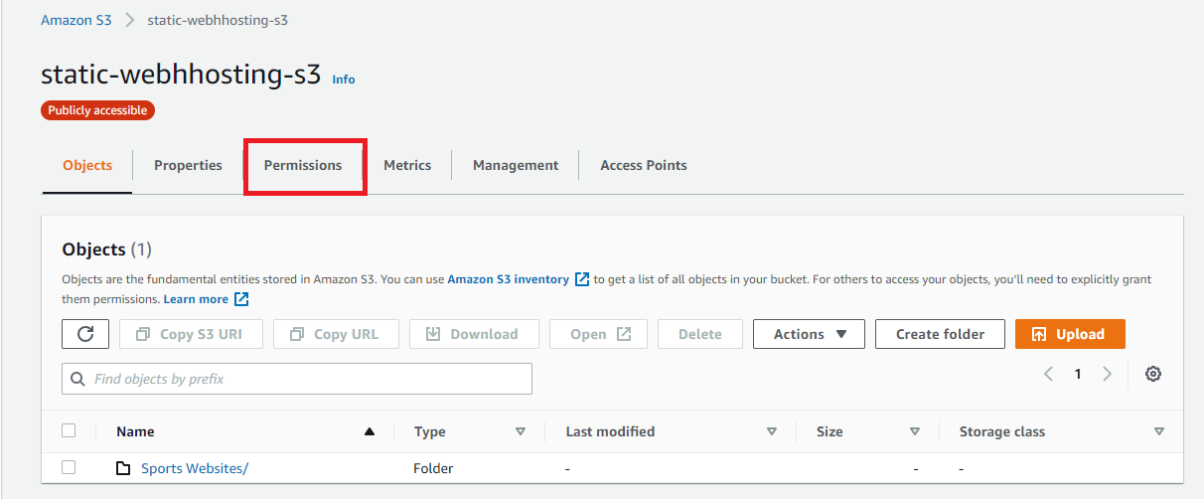
==========



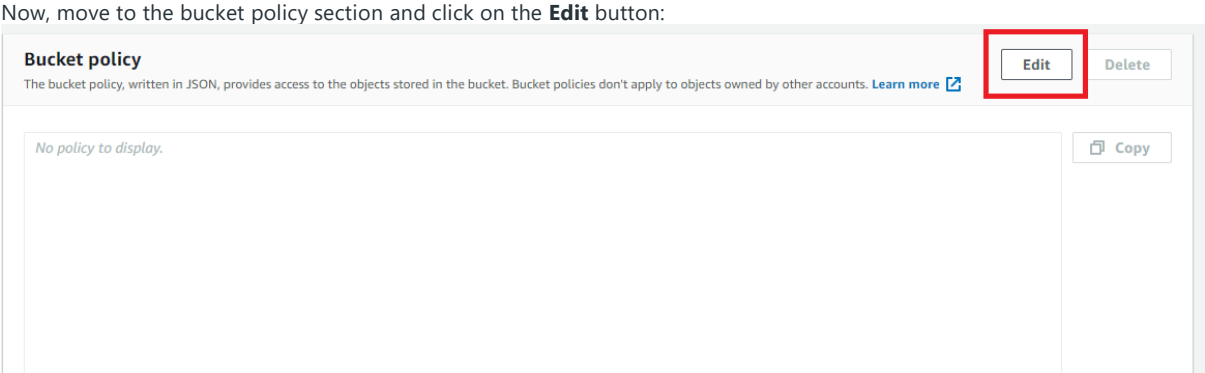
This problem can be solved by using the S3 bucket policies.

Setting up Permissions in S3 Bucket

To make our content accessible publicly, we need to add a bucket policy for which we have to go to the permissions tab of our S3 bucket to make some changes to the permissions of our S3 bucket:



Now, move to the bucket policy section and click on the **Edit** button:



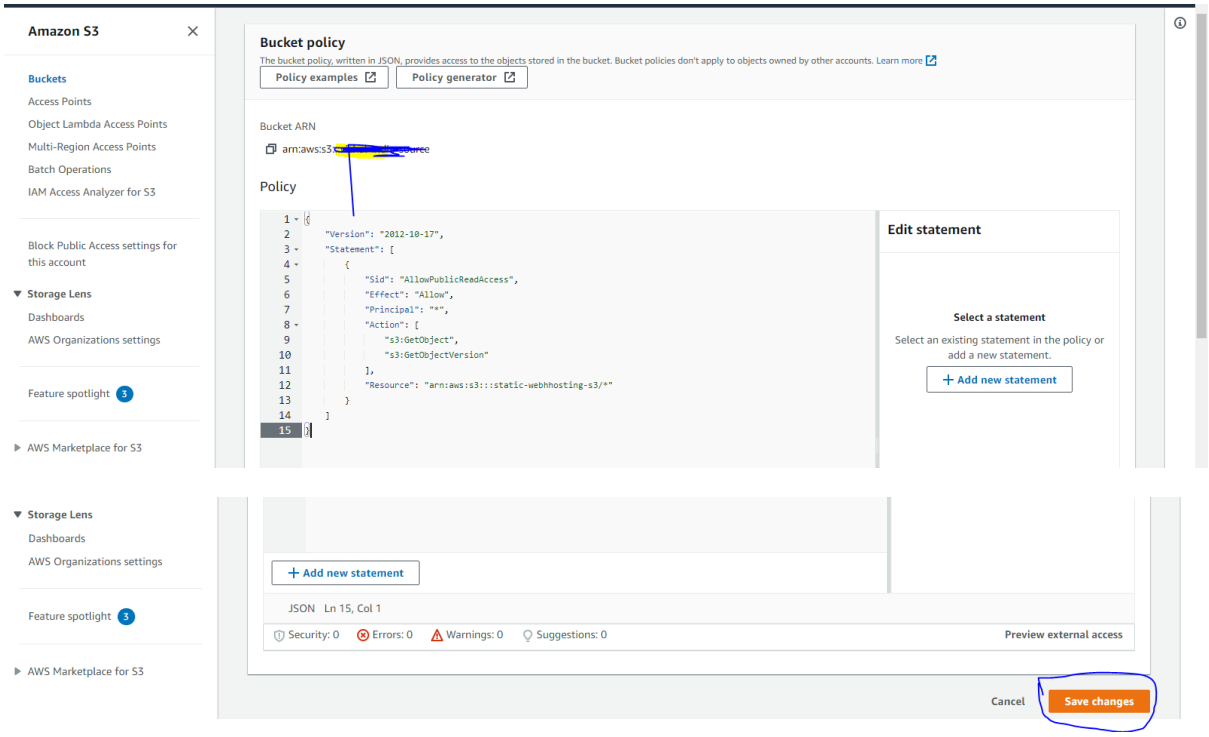
Paste the following JSON in the editor to allow the public to read files from the bucket:

Chapter 4.0 : AWS S3 service

==========

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPublicReadAccess",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::YOUR-S3-BUCKETNAME/*"
        }
    ]
}
```

Make sure to replace "**YOUR-S3-BUCKETNAME**" with your S3 bucket name in the JSON policy.

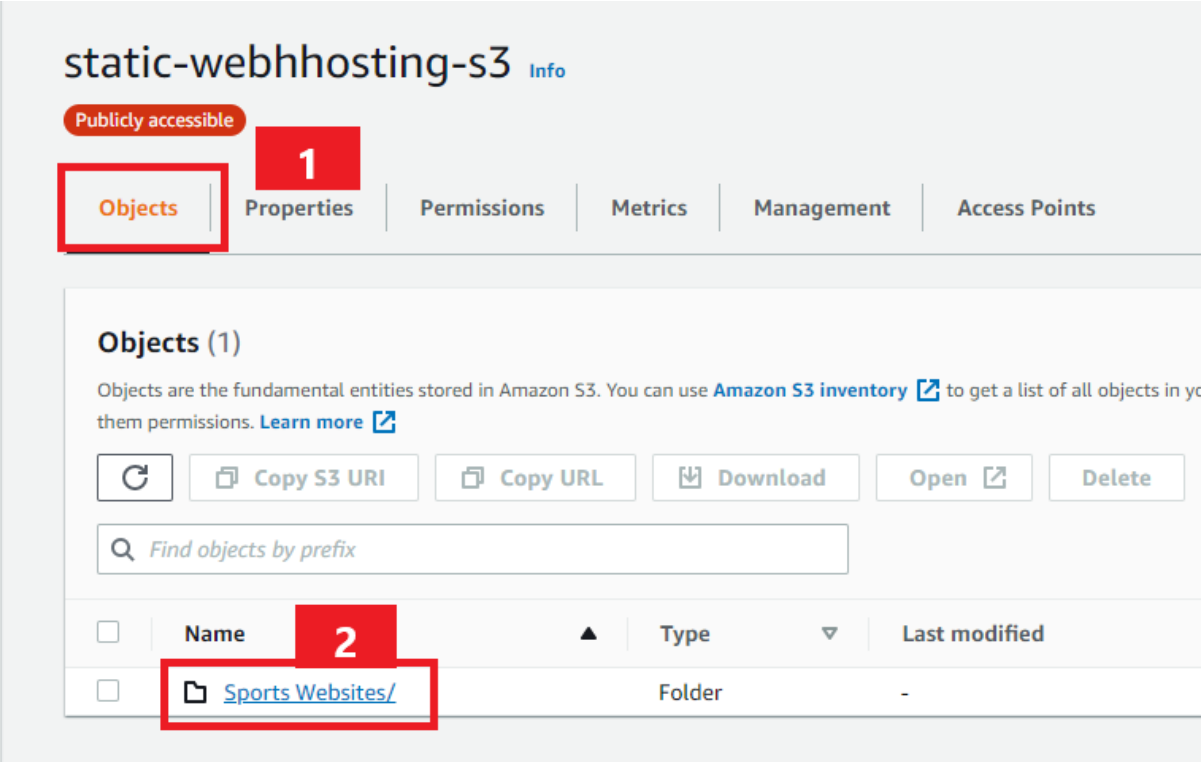


Accessing the Website Through URL

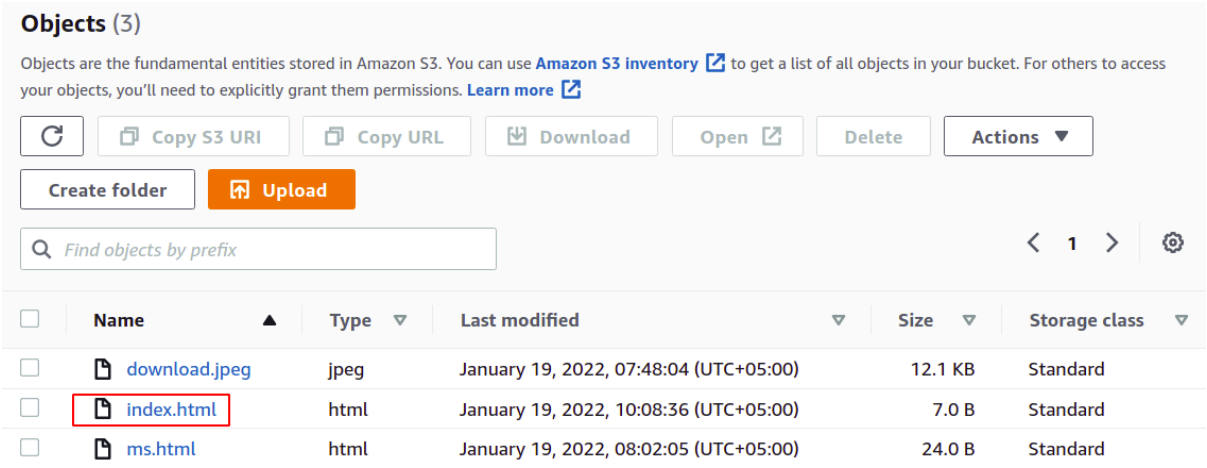
After setting the permissions for the bucket, it's time to access the webpage through the URL. For this, go to the **Objects** tab of the S3 bucket and go to the static site directory:

Chapter 4.0 : AWS S3 service

==========



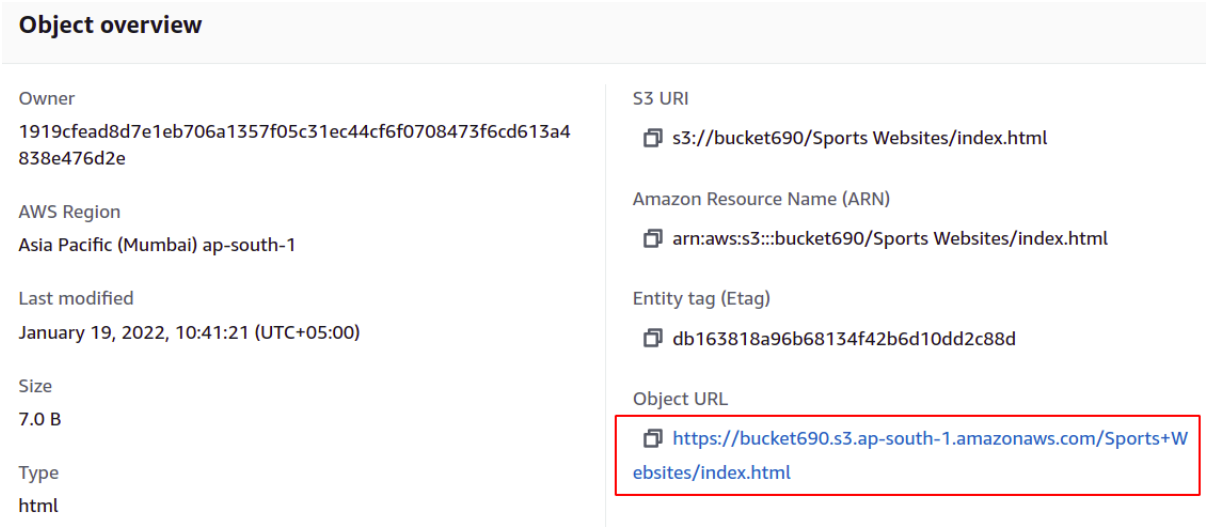
Look for the index.html file in the folder, which you defined as the index document for this project. Click on the index.html file:



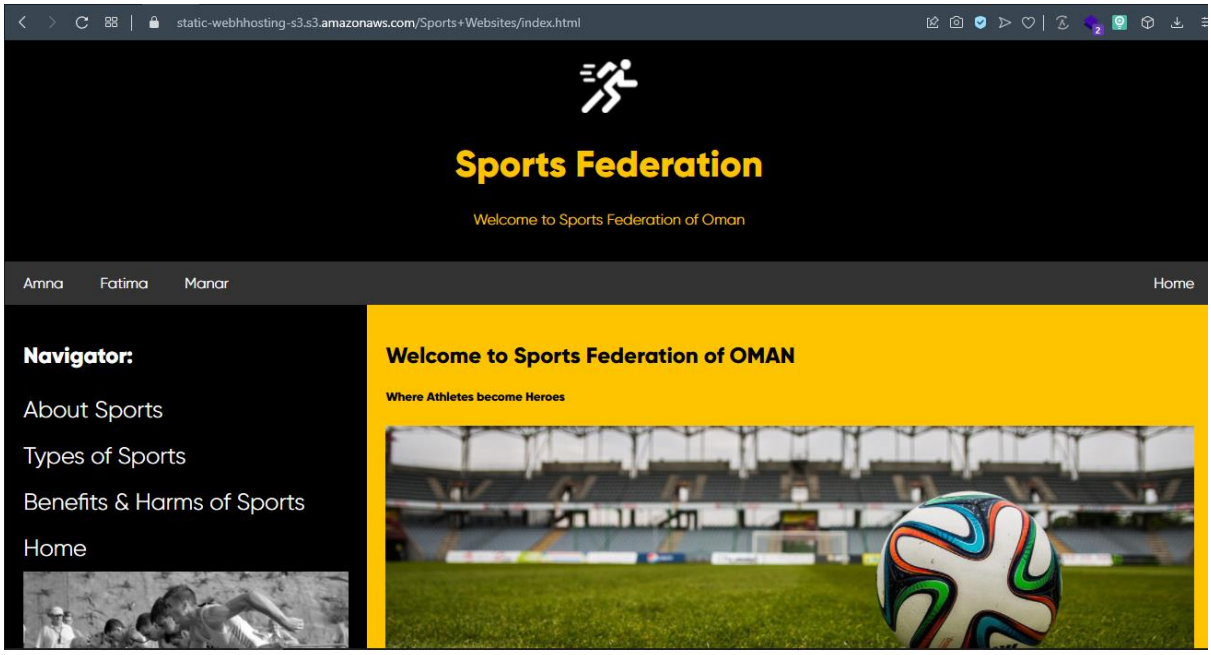
Now, in the object overview section under the properties tab, you can find the URL of the static website:

Chapter 4.0 : AWS S3 service

==========



Go to this URL, and the static website hosted on the AWS S3 bucket will be accessible via browser:



Conclusion

Creating, managing, and hosting websites and webpages and sharing data publicly is very important and crucial as this provides the public face of most brands and organizations. Looking at this perspective, AWS has developed a great idea to publicly provide an easy and simple solution for their users to host content using the S3 bucket. This guide describes simple steps to host your static website using the AWS S3 bucket.

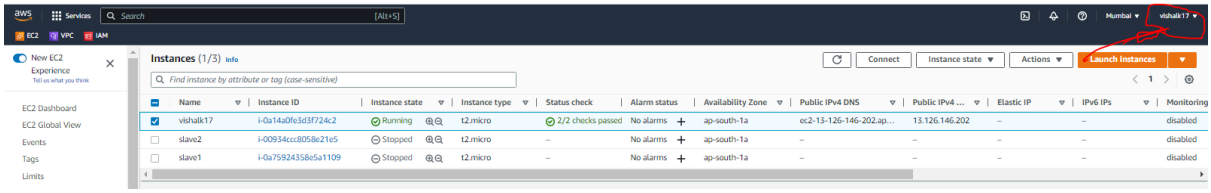
Chapter 4.0 : AWS S3 service

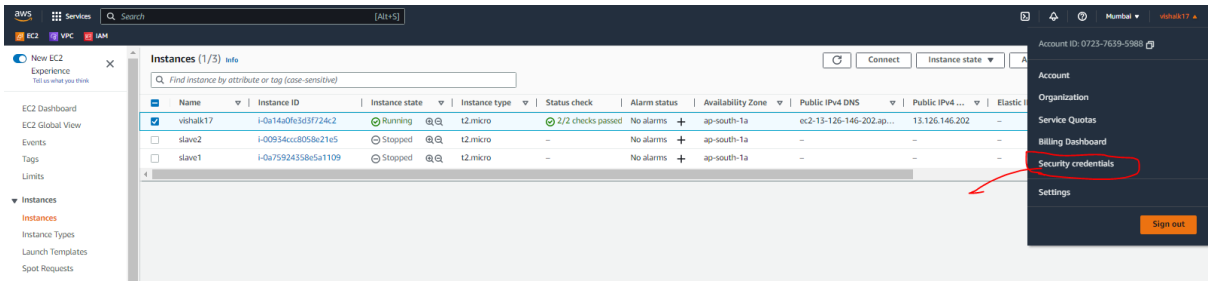
==========

S3- CLI :

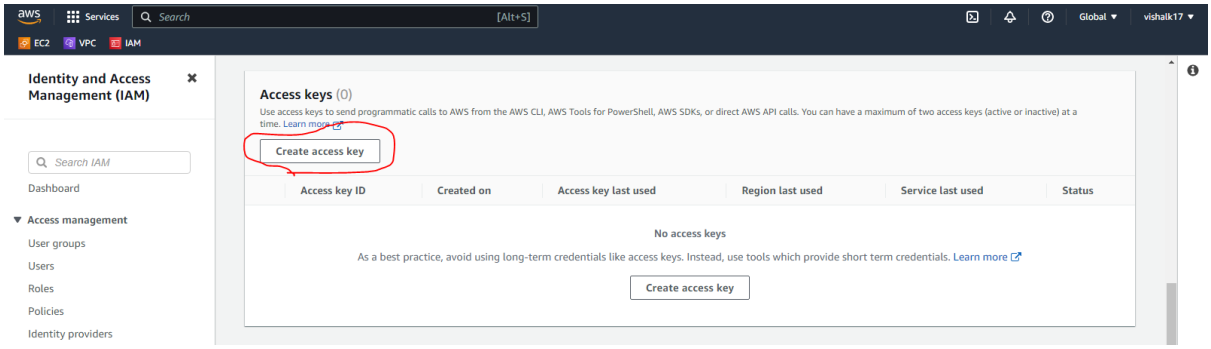
aws access key and Secret access key :

At corner click on user account. Then security credential

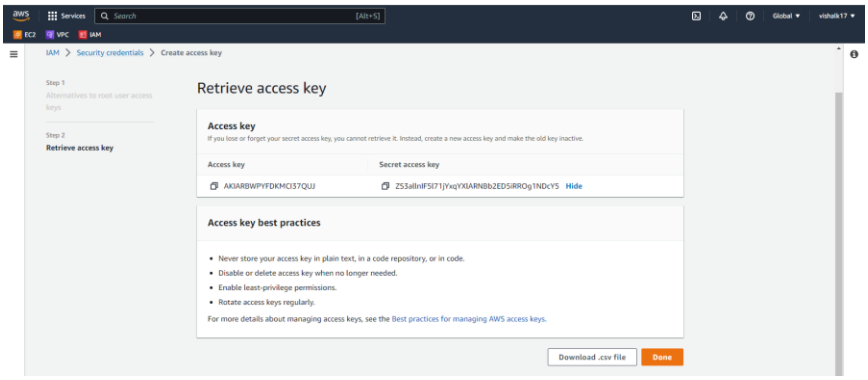




You will be redirected to the I am console >> next under access key >> click on create access key



You will get your access key and Secret key save those and will be require at the time of aws cli



Chapter 4.0 : AWS S3 service

==========

Configure aws cli :

```
2. master
[root@ip-172-31-39-3 ~]#
[root@ip-172-31-39-3 ~]#
[root@ip-172-31-39-3 ~]# aws configure
AWS Access Key ID [None]: AKIARBWPYFDKMCI37QUJ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: table
[root@ip-172-31-39-3 ~]#
[root@ip-172-31-39-3 ~]#
```

- You can choose region and output format (either in table, json, text)

- you could also have multiple profile with different region and output format.

    - you will need to pass –profile argument

```
aws configure :

aws describe-instances // all info about instances
aws ec2 stop-instances --instance-ids i-0a75924358e5a1109 --profile default ..... to stop particular instance


aws s3 ls                                          .. list bucket list
aws s3 ls s3://bucket_name                         ..list contents of bucket_name

aws s3 mb s3://bucket_name                         ..create bucket , mb= make bucket
aws s3 mb s3://bucket_name --region us-east-2      .. make bucket in another region , us-east-2= ohio region

aws s3 rb s3://bucket_name                         .. remove empty bucket
aws s3 rb s3://bucket_name --force                 .. forcely remove & empty bucket

aws s3 cp /path/of/filename s3://bucket_name/      .. to copy file in s3 bucket
aws s3 cp directory_name s3://bucket_name/ --recursive  .. copy all files of a directory in s3 bucket
                                                   .. to upload multiple files u need to move them in folder to upload


aws s3 cp s3://vishu-vishu/docker-compose.yml s3://vish-buck/      .. copy file from one bucket to another
aws s3 cp s3://vishu-vishu/docker-compose.yml /mnt/vish/           .. download file form s3 bucket to local machine

aws s3 mv s3://vishu-vishu/docker-compose.yml s3://amol-manoj/     .. move file from one bucket to another
aws s3 mv docker-compose.yml s3://vishu-vishu/                     .. move file from local machine to bucket & vice versa


aws s3 rm s3://vishu-vishu/docker-compose.yml                      .. remove file from the bucket
aws s3 rm s3://vishu-vishu/test-dir --recursive                    .. remove directory 'test-dir'

aws s3 sync . s3://amol-manoj                                      .. sync current dir with s3 bucket , upload all contain in current dir.
aws s3 sync s3://amol-manoj .                                      .. sync s3 bucket with current dir, download all contain in s3 bucket

**


aws s3 cp s3://amol-manoj/docker-compose.yml s3://amol-manoj/docker-compose.yml --storage-class DEEP_ARCHIVE

                                        -- change storage-class of already existed object in a bucket


aws s3 cp index.html s3://amol-manoj/ --storage-class STANDARD_IA      .. change storage class during upload of a file
```