SIMPLS STORAGE SERVICE (S3)

**Need more Space? Storage?**

* The need for storage is increasing every day, so building and

maintaining your own repositories, therefore, becomes a tedious and

tiresome job because knowing the amount of capacity you may need

in the future is difficult to predict.

* You may either over-utilize it leading to an application failure because

of not having sufficient space or you may end up buying stacks of

storage which will then be under-utilized.

* Keeping all these hassles in mind, Amazon came up with an internet

storage service called AWS S3.

**Type of storage**

* SimpleStorageService-S3
* ElasticFileSyste-EFS
* StorageGateway-SG
* SnoballEdge-SE
* ElasticBeanstalk-EBS
* Glacier
* Snowbal
* Snowmobile
* Amazon S3 (Simple Storage Service) is a scalable, high-speed, lowcost web-based service designed for online backup and archiving of

data and application programs.

* This service allows the subscribers to access the same systems that

Amazon uses to run its own web sites. The subscriber has control

over the accessibility of data, i.e. privately/publicly accessible.

* S3 is object-based storage and its Global
* In S3, we can store all FLAT files
* S3 is safe place to store files
* We can upload and down load / access files from S3
* The files in S3 can’t be executed
* We cannot install OS and DB etc in S3
* We cannot attach S3 to EC2 instance, but we can access S3 from EC2 instance
* S3 is unlimited Storage and Mnium object size 0 bytes, Max object size is 5TB

Means single put, we can upload 0 bytes to **5TB data (Graphical / Console)**

* File can be from 0 bytes to **5 GB (CLI)**
* S3 is support static website hosting
* AWS recommended, if you have > 100MB use MPU
* S3 is cheaper than EC2 instances
* S3 is server less and manged by AWS
* S3 is write once read many times also called WORM Model
* S3 support Static Website Hosting (SWH)(HTML files)
* Secure data using Access control lists & Bucket policy
* File object limit is 5000GB/5TB

Multi-Part Upload (MPU)

* For objects larger than 100 megabytes, customers should consider using the Multipart Upload capability.
* For transferring large data use **snowball**

**Use case for S3**

* Best place to maintain application logs
* Best place to maintain application images/documents/videos (Static Content)

**Benefits**



The largest object that can be uploaded in a single PUT is 5 gigabytes. Low cost and Easy to Use − Using Amazon S3, the user can store a large amount of data at very low charges.

**Secure −** Amazon S3 supports data transfer over SSL and the data gets encrypted automatically once it is uploaded. The user has complete control over their data by configuring bucket policies using AWS IAM.

**Scalable −** Using Amazon S3, there need not be any worry about storage concerns. We can store as much data as we have and access it anytime.

**Durable-** It regularly verifies the integrity of data stored using checksums e.g. if S3 detects there is any corruption in data, it is immediately repaired with the help of replicated data.

**Higher performance −** Amazon S3 is integrated with Amazon CloudFront, that distributes content to the end users with low latency and provides high data transfer speeds without any minimum usage commitments.

**Integrated with AWS services −** Amazon S3 integrated with AWS services include Amazon CloudFront, Amazon CloudWatch, Amazon Kinesis, Amazon RDS, Amazon Route 53, Amazon VPC, AWS Lambda, Amazon EBS, Amazon Dynamo DB, etc.

**Bucket**

* A bucket is a container for objects stored in Amazon S3. Every object is contained in a bucket.
* **For example**, if the object named photos/puppy.jpg is stored in the johnsmith bucket, then it is addressable using the URL

[**http://johnsmith.s3.amazonaws.com/photos/puppy.jpg**](http://johnsmith.s3.amazonaws.com/photos/puppy.jpg)

* Data in S3 is organized in the form of buckets.
* A Bucket is a logical unit of storage in S3.
* A Bucket contains objects which contain the data and metadata.
* Before adding any data in S3 the user has to create a bucket which will be used to store objects
* You can configure buckets so that they are created in different regions.
* You can also configure a bucket so that every time an object is added to it, Amazon
* S3 generates a unique version ID and assigns it to the object.
* Buckets are reginal and bucket creation is mandatory for upload any file
* Buckets names are universal or unique
* Buckets names are public and private
* No nested buckets (i.e., bucket under bucket cannot create)
* Bucket is container of objects
* Object is nothing but file
* Name of the file is called Key
* We can create multiple buckets in different regions
* Maximum number of buckets we can create in S3 is 100 (soft limit)
* By default, buckets are private, if required we can make it public
* Every object will have its own URL

**Example URL**

**key**

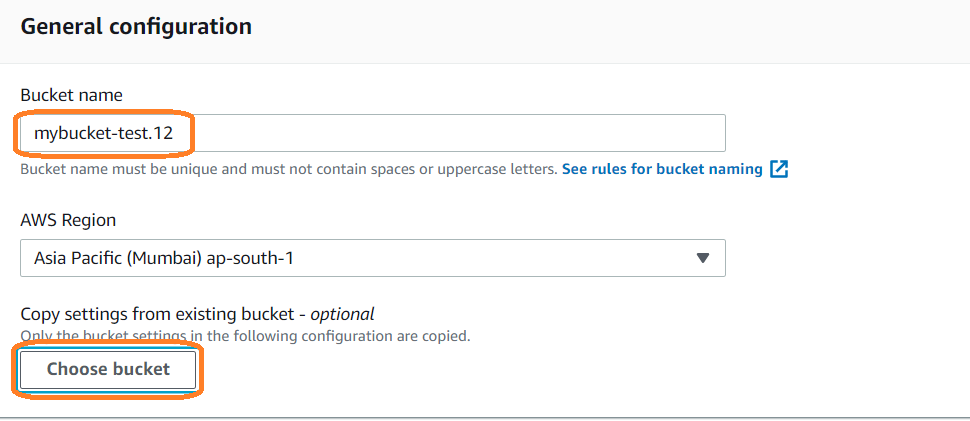
[**http://9ambucket.s3.amazonaws.com/Photos/Puppy.jpg**](http://9ambucket.s3.amazonaws.com/Photos/Puppy.jpg) **suffix**

Bucket Name domain name Pre-fix Object

(Sub-folder)

**Bucket naming rules**

* The bucket name can be between 3 to 63 characters long, and can contain only lower-case characters, numbers, periods(.), and dashes
* Each label in the bucket name must start with a lowercase letter or number
* **The bucket name cannot contain under scores, end with a dash, have consecutive periods, or use dashed adjacent to periods**
* **The bucket name cannot be formatted as an IP address**

****

**Objects**

* Objects are fundamental entities stored in Amazon S3
* Objects consist of object data and metadata.
* The metadata is a set of name-value pairs that describe the object. These include
* some default metadata, such as the date last modified, and standard HTTP

metadata, such as Content-Type. You can also specify custom metadata at the time the object is stored.

* An object is uniquely identified within a bucket by a key (name) and a version ID

**Keys**

* A key is the unique identifier for an object within a bucket. (Simply name of the object.)
* Every object in a bucket has exactly one key. Because the combination of a bucket, key, and
* version ID uniquely identify each object.
* Amazon S3 can be thought of as a basic data map between "bucket + key + version" and the object itself. Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, key, and optionally, a version.
* For example, in the

URL **http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl**, "**doc**" is

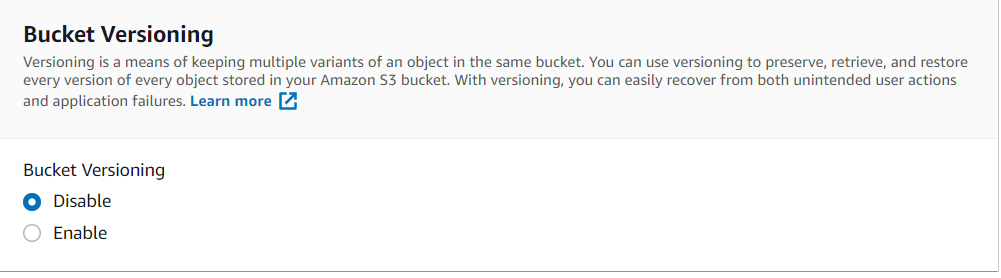
the name of the bucket and "**2006-03-01/AmazonS3.wsdl**" is the key.

**Regions**

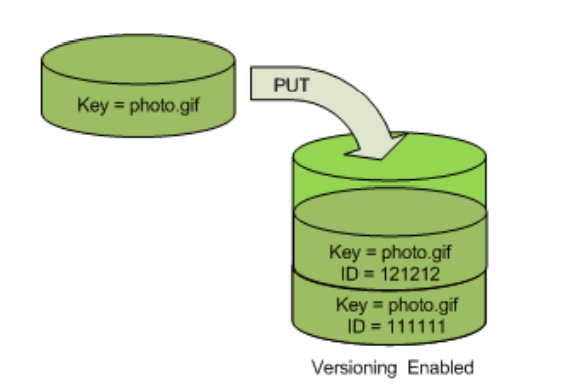
* You can choose the geographical region where Amazon S3 will store the buckets you create

**Bucket Versioning**

* Versioning helps to protect your data against accidental or malicious deletion by keeping multiple versions of each object in the bucket, identified by a unique version ID
* Versioning is a great backup tool
* By default, versioning is not enabled on the bucket
* If you have not enabled versioning, Amazon S3 sets the value of the version ID to null. If you have enabled versioning, Amazon S3 assigns a unique version ID value for the object. When you enable versioning on a bucket, objects already stored in the bucket are unchanged. The version IDs (null), contents, and permissions remain the same.
* Versioning is enabled on bucket level, but applied on object level
* Version ID is always unique
* Version files can be downloaded any time
* If delete the original object, delete marker is applied on the latest version
* If we want the object to be restore, delete the delete marker then object will be restored
* We can’t download the delete marker version, we can only delete it
* Once we have enabled the versioning, we cannot disable it, we can suspend it
* AWS charges for versioning, becarefull while enabled versioning for huge files
* if we upload an object after the versioning is suspended, the latest version will be created as usual But, if we update the original object, versioning files are NOT CREATED
* If we delete the original object, DELETE MARKER will be applied
* If we delete the DELETE MARKER, Object will NOT be Restored
* Old objects what was uploaded wen versioning was enabled, that will be continued as usual even after suspending the versioning
* Old versions we can download and restore / delete
* For disabling Versions also, we can set MFA.

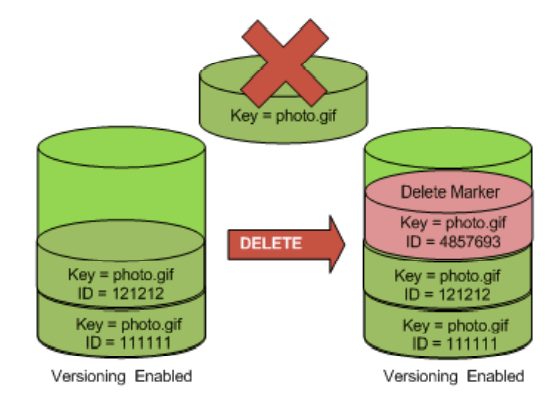


* You PUT an object in a versioning-enabled bucket, the noncurrent version is not overwritten. The following figure shows that when a new version of **photo.gif** is PUT into a bucket that already contains an object with the same name, the original object (**ID = 111111**) remains in the bucket, Amazon S3 generates a new version **ID (121212),** and adds the newer version to the bucket



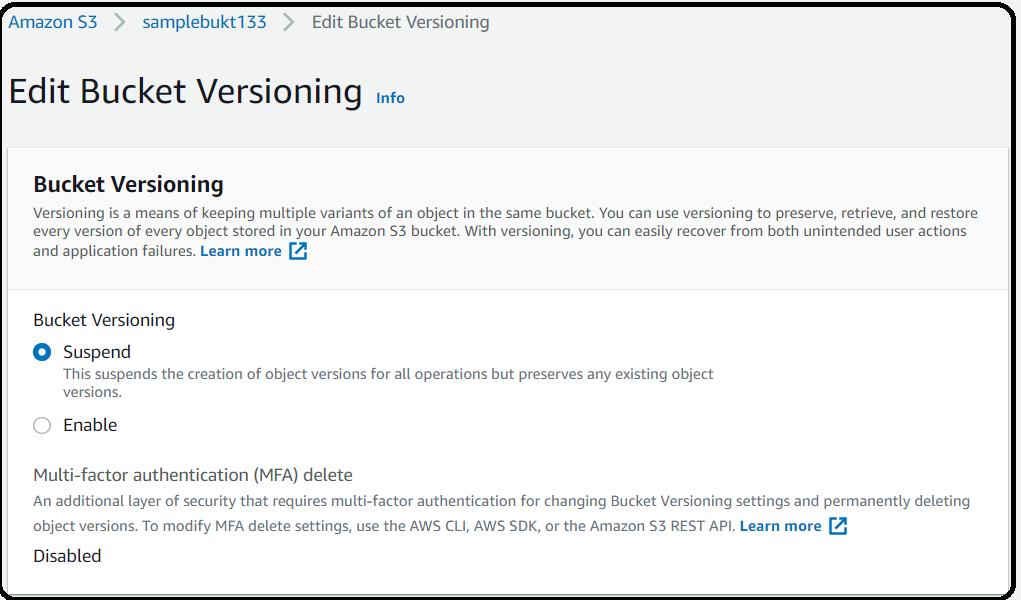
This functionality prevents you from **accidentally overwriting or deleting objects** and affords you the opportunity to retrieve a previous version of an object.

When you DELETE an object, all versions remain in the bucket and Amazon S3 inserts a **delete marker**, as shown in the following figure.

****

**Versioning suspended**

* if you upload the object after the versioning is suspended, the latest version will be created,
* If you delete the original object, delete marker will be applied.
* Once you delete the delete marker, the object **WILL NOT BE RESTORED**
* You cannot even download the delete marker object



**Storage Classes**

* Whenever we are uploading an object that object will store in some physical infrastructure is called Storage Class
* When files are uploaded to the bucket, the user will specify the types of s3 storage class to be used for those specific objects
* While uploading the object in S3 selecting the storage class is mandatory
* Charges is based on object size and storage class
* We cannot change the storage classes manually for this we can use object life cycle rule.

**Standard Frequency Access (FA by default)**

* The files which we are accessing frequently we can maintain in standard storage class. So that our object is highly available and fast retrieval.
* Can be used for cloud applications, dynamic websites, content distribution, gaming applications, and big data analytics
* Data will be maintained in 3 availability zone i.e 3 copies
* No retrieval charges
* Availability (Any Time): 99.9%
* Amazon Guarantee Durability (Long Term): 99. 999999999%
* Minimum Object = 0 bytes

**Standard Infrequency Access (IA)**

* We can maintain files which we are not accessing frequently Cheaper than FA
* Data will be maintained in 3 availability zone i.e 3 copies but retrieval very slow
* Suitable for backups, disaster recovery and lifelong storage of data
* Retrieval Charges apply
* Demand rapid access
* Availability: 99.9%
* Durability: 11 9's
* Minimum Object size: 128KB
* Minimum Duration: 30days

**Reduced Redundancy Storage (RRS)**

* Frequently access but **NOT CRITICAL &**Re producible data we can maintain in RSS
* Data will be **maintained only in one Availability** Zone (only one copy)
* No Retrieval Charges
* AWS does not recommend to use this storage class
* Cheaper than others
* Availability: 99.99%
* Durability: 99.99%

**One Zone IA**

* Infrequently access but NOT CRITICAL
* Retrieval Charges apply
* Availability: 99.5%
* Durability: 11 9'S of objects in a single AZ; data lost when AZ is destroyed
* Minimum Object size: 128KB
* Minimum duration=30 days
* Low latency and high throughput performance
* Low cost compared to IA (by 20%)

**Intelligent Tiering**

* Automatically moves objects between two access tiers based on changing access pattern
* If customer don knows FA or IA then he/her can use Intelligent Tiering
* Unknown access patterns
* Availability: 99.9%
* Durability: 11 9's
* Minimum Duration: 30 days

**S3 Glacier**

* Glacier is an extremely low-cost storage service that provides durable, secure, and flexible storage for data archiving and online backup. Storage class offers secure, durable and extremely low-cost cloud storage for

data that does not require real time access, such as archives and long-term backups

* We can maintain long term & archived data in glacier for very less price
* **Archives:** In Glacier, data is stored in archives. An archive can contain up to **40 TB** of data, and you can have unlimited number of archives
* But charges are applicable when we read the data. It takes more time if you want retrieve
* Suitable for archiving data where data access is infrequent
* Vault-lock feature provides a long-term data storage
* provides the lowest cost availability
* Glacier contain container called **Vault**
* Vault: Vaults are containers for archives. Each AWS account can have up to 1000 vaults.
* Archives are stored in” **Vaults**”
* In Glacier, data is stored in archives. An archive can contain up to **40 TB** of data, and you can have unlimited number of archives
* Retrieval charges apply

**Vault Policies and Vault Lock**

* Vault is a collection of archives
* Each Vault has:
* ONE vault access policy
* ONE vault lock policy
* Vault Policies are written in JSON
* Vault Access Policy is similar to bucket policy (restrict user / account permissions)
* Vault Lock Policy is a policy you lock, for regulatory and compliance requirements.
* The policy is immutable, it can never be changed (that’s why it’s call LOCK)
* Example 1: forbid deleting an archive if less than 1 year old
* Example 2: implement WORM policy (write once read many

**S3 Glacier Deep Archive** To get data need to wait for 12 hours

**Glacier has retrieval option**

* Expedited: 1 to 5 minutes
* Standard: 3 to 5 hours
* Bulk: 5 to 12 hours
* Availability: 99.99%
* Durability: 11 9's
* Minimum duration: 90 days

**Life Cycle Management**

* In Life Cycle management, Amazon S3 applies a set of rules that define actions to a group of objects.
* By using Life Cycle management, we can automate the storage tiers in s3 buckets. We can move objects from one storage class/tier to another storage class/tier based on our business requirement
* Life Cycle Management is created on bucket level and applied on object level
* we can archive them or deleting them after a specified period of time
* Minimum 30 days is mandatory for moving files from one storage class to another storage class
* By using this we can achieve cost saving

**There are 2 features**

* **Transition** Moving object from one storage class (SFA) to another storage class (Glacier) is called **Transition**
* **Expiration** For example, LCM rule 30 days in SFA and move next to Glacier (60 days) after those 365 days delete after certain time (**Expiration**)
* Transition is applicable for current version and previous version also

**Logging**

* By enabling logs, we can track requests on your Amazon S3 bucket. Logging is off by default. You can enable it from bucket properties.
* Every log will contain the below information
* Requestor account and IP address
* Bucket Name
* Request time
* Action (GET, PUT, LIST etc)
* Response status or error code.

**S3 Access Logs**

* For Audit purpose, you may want to log all access to S3 buckets
* Any requests made to S3, from any account, authorized or denied, will be logged into another S3 bucket
* That data can be analysed using data **analysis tools**
* We can create separate bucket for logs
* All logs are in bucket log, if we want object level, we need to integrate cloud trail
* Logs file not readable format and difficult to read and analyse the logs

**Amazon Athena**

* Serverless service to perform analytics directly against S3
* Uses SQL language to query the files
* Has a JDBC/ODBC drivers
* Charged per query and amount of data scanned
* Supports CSV, JSON, ORC, Avro and Parquet (Athena uses Presto Query
* Engine)
* If we have requirement like analyse the logs, we have a separate service in AWS i.e **Athena**
* Athena will analyse the logs directly from S3 bucket
* It will give a chance to create table, make you analyse the logs

**For example:**

|  |  |  |  |
| --- | --- | --- | --- |
| IP Address | Source | Destination | Object |
|  |  |  |  |
|  |  |  |  |

All the logs will be stored in above format by using sql statement we can retrieve the data and analyse.

**Cross Region Replication ( CRR )**

* With cross region replication Amazon S3 allows you to asynchronously replicate all new objects in the
* source bucket in one AWS region to a target bucket in another region.
* Versioning must be enabled on both the source and destination buckets.
* Regions must be unique
* Files in an existing bucket are not replicated automatically, All subsequent/Future updated files will be replicated automatically.
* You cannot replicate to multiple buckets or user daisy chaining (at this time).
* Delete markers are not replicated
* Deleting individual versions or delete markers will not be replicated.
* Cross-region replication is used to reduce the latency required to access objects in Amazon S3 by placing objects close to a set of users or to meet requirements to store backup data at a certain distance from the original source data.
* Amazon s3 must have permission to replicate objects from that source bucket to the destination bucket on your behalf.
* You can replicate buckets to another account bucket as well
* Nothing but “duplication”
* Duplication of data in different region
* CRR is not enabled by default
* CRR is enable on bucket level
* **CRR can be configured across accounts**
* Objects will be replicated only after creating CRR rule and existing objects are not replicated. Copy it manually
* We can replicate S3 bucket another region using replication rule
* It supports replication (In Same Account and diff account)

1. Same-Region Replication (SRR)
2. Cross-Region Replication (CRR)

**Same-Region Replication**

* Replication of same bucket one region to another region.

**Amazon S3 Transfer acceleration**

* Amazon S3 Transfer acceleration enables fast, easy and secure transfers of files over long distances between your client and an s3 bucket.
* Transfer acceleration takes advantage of Amazon CloudFront’s globally distributed edge locations.
* As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path. Charges apply
* By using the Amazon S3 Transfer Acceleration Speed Comparison tool we can compare the accelerated and non-accelerated upload speeds across s3 regions
* The speed Comparison tool uses multipart uploads to transfer a file from your browser to various S3 regions with and without using Transfer acceleration.
* Once enable it you cannot disable it only suspend it

**Amazon S3 Events**

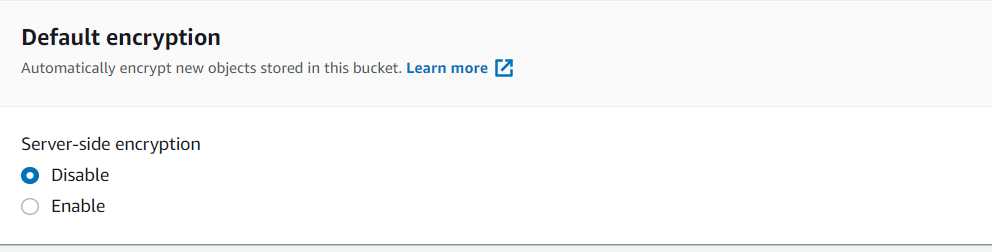
* Amazon S3 Event notifications can be sent in response to the actions taken on objects uploaded or stored in Amazon S3. The Amazon S3 notification feature enables you to receive notifications when certain events happen in your buckets.
* Notification messages can be sent through either Amazon SNS or SQS or directly to AWS Lambda functions.

**Encryption**

* Encryption for security
* **BY Default**, all newly created buckets are PRIVATE, you can setup access control to your buckets using
* **Bucket Policies**
* **Access Control lists**
* S3 Bucket can be configured to create access logs which log all requests made to the bucket. This can be sent to another bucket and even another bucket in another account
* Encryption is done in two ways

**In-Transit:** We can use SSL API endpoints; this ensure that all data send to and from Amazon S3 is encrypted while in transit using the HTTPS protocol.

**Data at Rest:** Component which data is sitting idle i.e EBS, EFS



**AWS S3 has 3 Types of encryptions**

* **Server-Side Encryption** 
  + - Should be handle by AWS
    - SSE – S3 (Default Managed by AWS)
    - SSE – KMS (AWS Key Management Service) if we want own key
    - SSE – C (Customer Provided Key)
* **Client-Side Encryption**

We cannot encrypt the data on the client before sending it to Amazon S3. We have to take care about the encryption and Decryption process.

* + - Should be handle by Customer
    - In-Transit Encryption: **HTTPS**

**ACM (Amazon Certificate Manager)**

* Is where we can generate HTTPS certificates
* We need to have a domain name

**Encryption algorithm (AES – 256) Advanced Encryption Standard is S3 is using**

**S3 Data Consistency Model – 2 Types**

* Read after write consistency for PUTS of NEW OBJECTS (immediately)
* Eventual consistency for OVERWRITES of PUTS and DELETES
* Changes will be reflected slowly

**Amazon S3 Pre-Signed URL**

* Can generate pre-signed URL using SDK or CLI
* Valid for a default of 3600 seconds, can change timeout with –expires-in time by seconds argument.
* Instead of giving permission for entire bucket we can assign pre-signed URL to download/upload.
* **Examples**

Allows only logged-in users to download a premium video on your s3 bucket

Allows temporarily a user to upload a file to a precise location in our bucket

* If we enable the bucket public everyone can see, I want to enable bucket for only few peoples in this case we can this concept called **Pre-Signed URL**
* Endpoint is valid for certain period of time
* **Examples**

Allows only logged-in users to download a premium video on your s3 bucket

Allows temporarily a user to upload a file to a precise location in our bucket

Example: **Premier show**

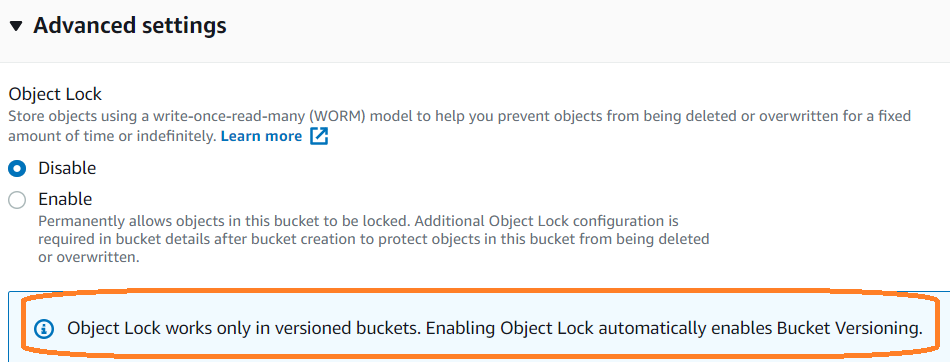
****

**Amazon S3 Inventory & Requester pays**

* Amazon S3 Inventory is one of the tools Amazon S3 provides to help manage your storage. Amazon S3 inventory provides a comma-separated values (CSV) flat file output of your objects and their corresponding metadata on a daily or weekly basis for an s3 bucket or a shared prefix.
* **Amazon S3 Requestor Pays**
* Generally, bucket owners pay for all Amazon S3 storage and data transfer costs associated with their bucket. If you enable Requester pays on the bucket, instead of bucket owner requested user will pay.
* Anonymous access to that bucket is not allowed, if we want to enable the requester pays on bucket

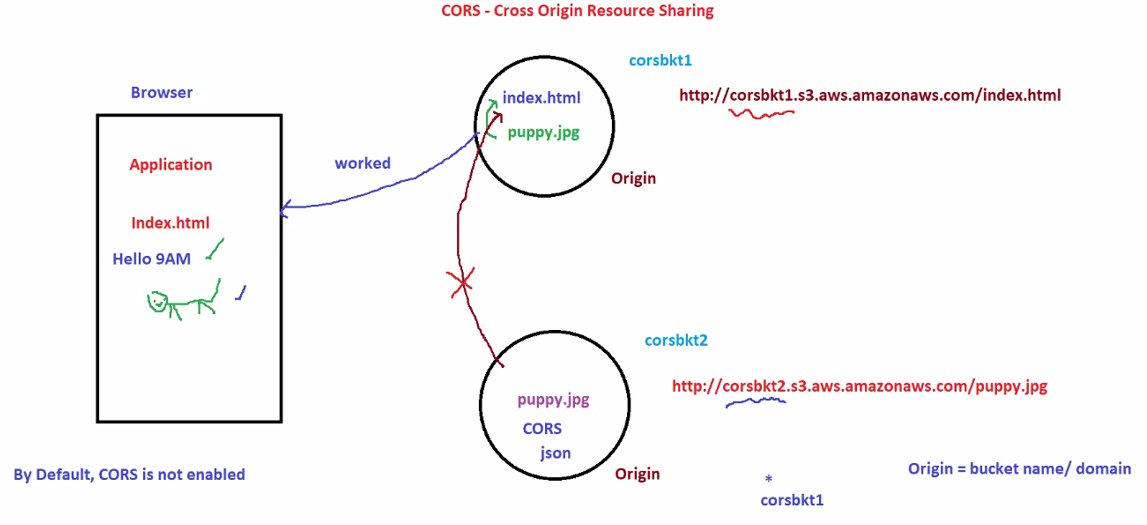
**Object Lock**

* Amazon S3 Object Lock enables you to store objects using a "Write Once Read Many" (WORM) model. Using S3 Object Lock, you can prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely.
* S3 Object Lock provides two ways to manage object retention: **retention periods and legal holds.**
* A retention period specifies a fixed period of time during which an object remains locked. During this period, your object will be **WORM-protected** and can't be overwritten or deleted.
* A legal hold provides the same protection as a retention period, but has no expiration date. Instead, a legal hold remains in place until you explicitly remove it.
* Object Lock works only in versioned buckets. Enabling Object Lock automatically enables Bucket Versioning
* If we don’t want object should be editable or readable, we can lock it permanently or certain period of time



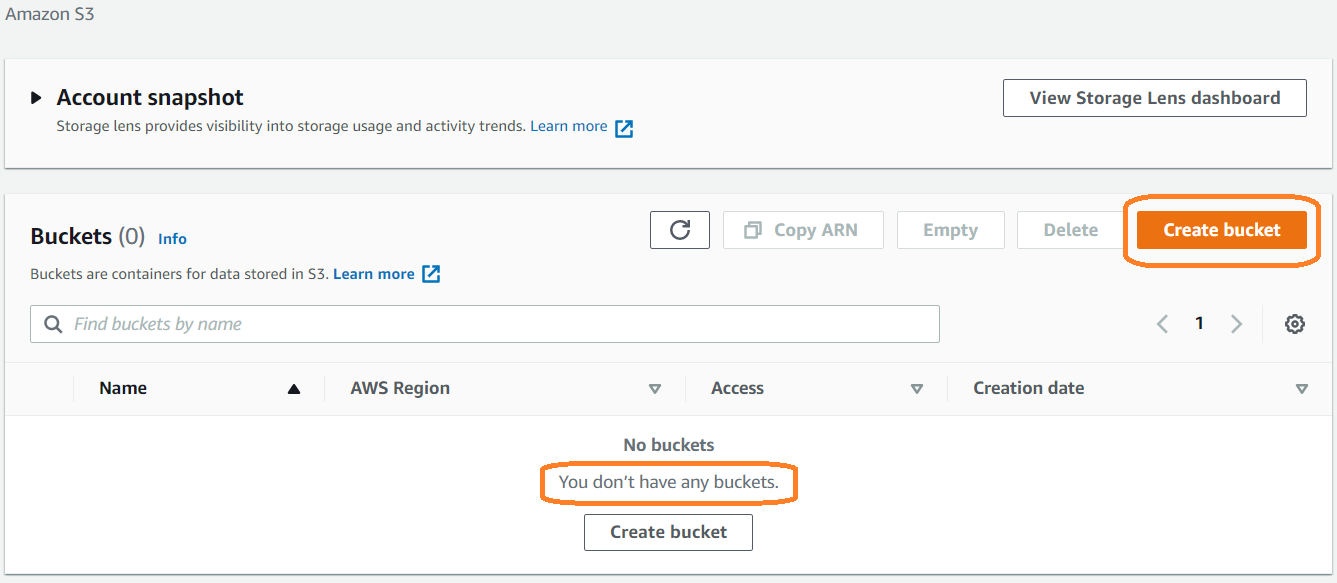
**CORS (Cross Origin Resource Sharing)**

* By default, CORS not enable
* CORS is json object
* In order to access same object in different bucket we need to enable CORS
* Other bucket should enable CORS

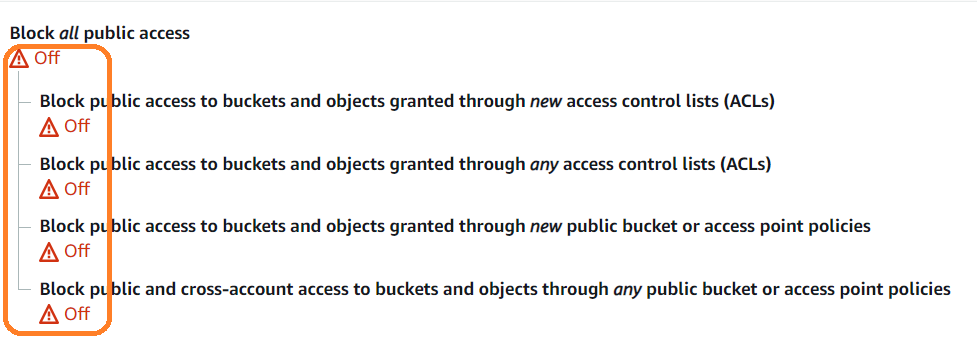


**Amazon S3 dashboard**

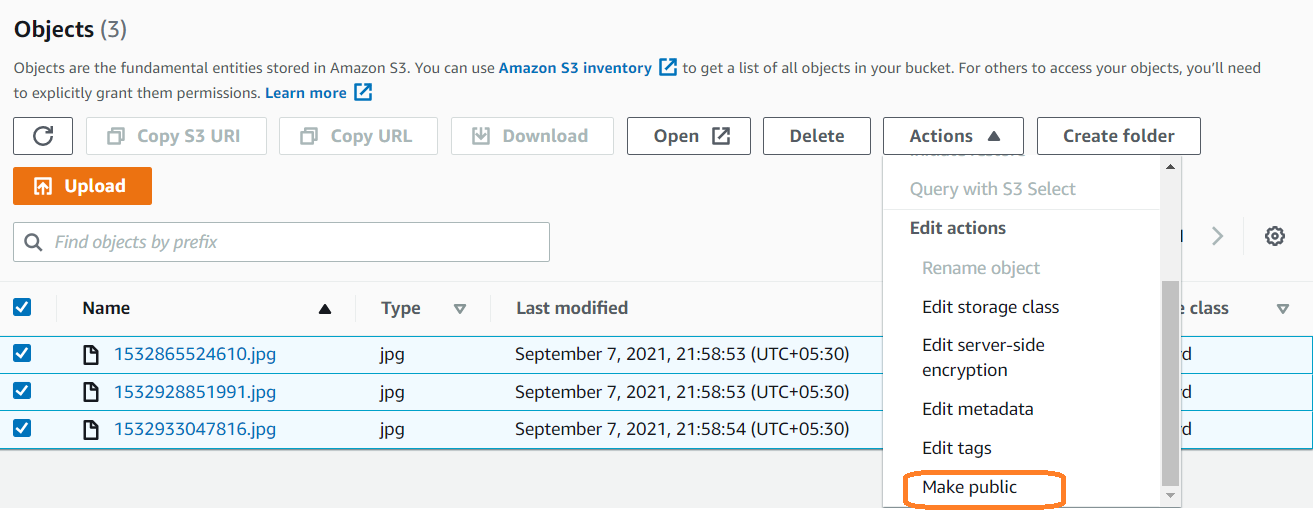
**Services 🡪 Storage 🡪 S3**



**Amazon S3 --> <Bucket-name> --> Permissions**



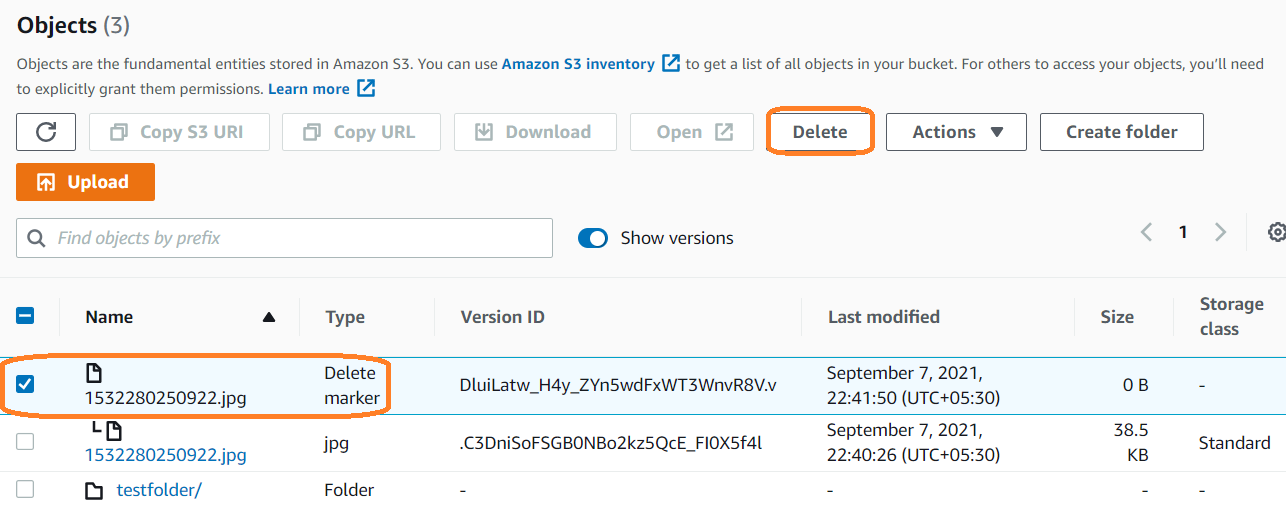
**Select all objects and make public**



Access from object end point URL

Empty folders cannot be restored even though we have enabled versioning

**Delete marker**



**Enable Server Access log**

1.Amazon S3 🡪 <bucket-name> 🡪 Edit server access logging

2.We need to create 2 buckets bucket-1 and bucket-2

3.while enable server log we need give target bucket as bucket-2

4.do some operation like upload and delete operations in bucket-1

5.verify after 1 hours expected logs are generated in buket-2 or not.

6.once logs generated delete all objects and buckets

**Note: Logs update will take minimum 1 hour**

