S3

* S3 is one of the first services that has been produced by aws.
* S3 stands for Simple Storage Service.
* S3 provides developers and IT teams with secure, durable, highly scalable object storage.
* It is easy to use with a simple web services interface to store and retrieve any amount of data from anywhere on the web.

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.

If you create a bucket, URL look like:

AWS S3

* If you upload a file to S3 bucket, then you will receive an HTTP 200 code means that the uploading of a file is successful.

Advantages of Amazon S3



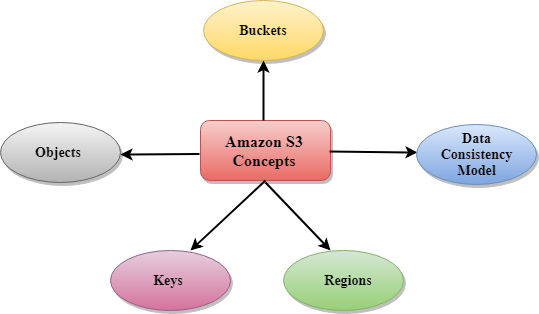
* **Create Buckets:** Firstly, we create a bucket and provide a name to the bucket. Buckets are the containers in S3 that stores the data. Buckets must have a unique name to generate a unique DNS address.
* **Storing data in buckets:** Bucket can be used to store an infinite amount of data. You can upload the files as much you want into an Amazon S3 bucket, i.e., there is no maximum limit to store the files. Each object can contain upto 5 TB of data. Each object can be stored and retrieved by using a unique developer assigned-key.
* **Download data:** You can also download your data from a bucket and can also give permission to others to download the same data. You can download the data at any time whenever you want.
* **Permissions:** You can also grant or deny access to others who want to download or upload the data from your Amazon S3 bucket. Authentication mechanism keeps the data secure from unauthorized access.
* **Standard interfaces:** S3 is used with the standard interfaces REST and SOAP interfaces which are designed in such a way that they can work with any development toolkit.
* **Security:** Amazon S3 offers security features by protecting unauthorized users from accessing your data.

S3 is a simple key-value store

**S3 is object-based. Objects consist of the following:**

* **Key:** It is simply the name of the object. For example, hello.txt, spreadsheet.xlsx, etc. You can use the key to retrieve the object.
* **Value:** It is simply the data which is made up of a sequence of bytes. It is actually a data inside the file.
* **Version ID:** Version ID uniquely identifies the object. It is a string generated by S3 when you add an object to the S3 bucket.
* **Metadata:** It is the data about data that you are storing. A set of a name-value pair with which you can store the information regarding an object. Metadata can be assigned to the objects in Amazon S3 bucket.
* **Subresources:** Subresource mechanism is used to store object-specific information.
* **Access control information:** You can put the permissions individually on your files.

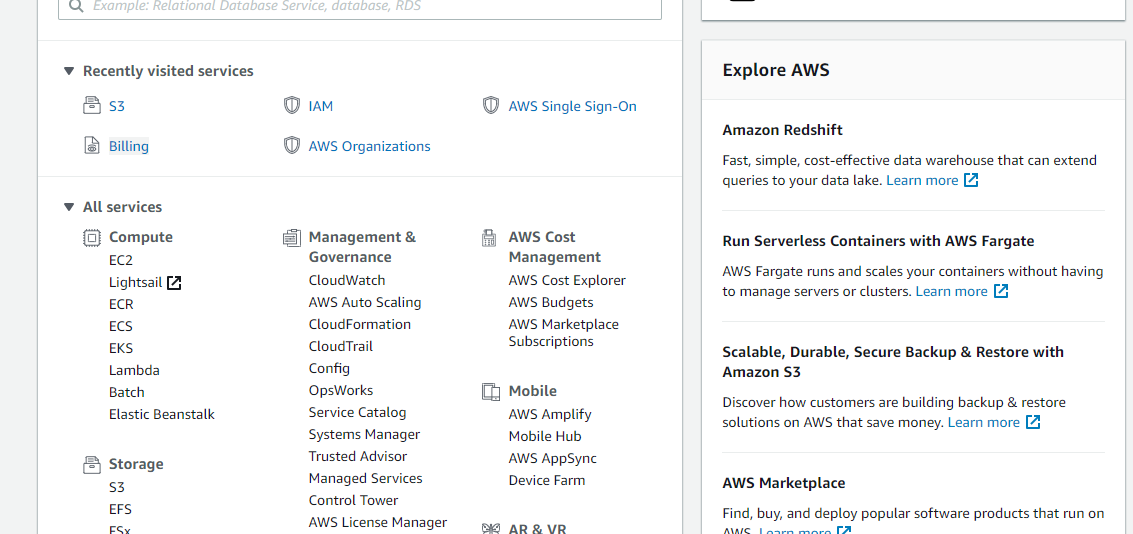
Amazon S3 Concepts



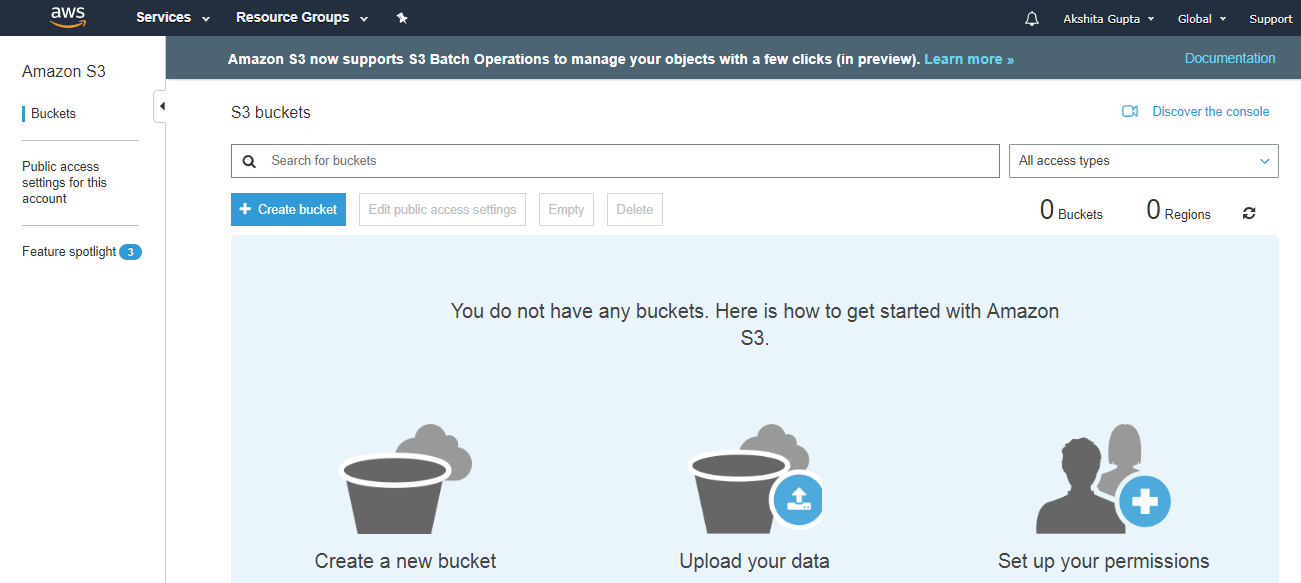
* Buckets
* Objects
* Keys
* Regions
* Data Consistency Model
* **Buckets**
  + A bucket is a container used for storing the objects.
  + Every object is incorporated in a bucket.
  + For example, if the object named photos/tree.jpg is stored in the tree image bucket, then it can be addressed by using the URL http://treeimage.s3.amazonaws.com/photos/tree.jpg.
  + A bucket has no limit to the amount of objects that it can store. No bucket can exist inside of other buckets.
  + S3 performance remains the same regardless of how many buckets have been created.
  + The AWS user that creates a bucket owns it, and no other AWS user cannot own it. Therefore, we can say that the ownership of a bucket is not transferrable.
  + The AWS account that creates a bucket can delete a bucket, but no other AWS user can delete the bucket.
* **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.
  + An object consists of some default metadata such as date last modified, and standard HTTP metadata, such as Content type. Custom metadata can also be specified at the time of storing an object.
  + It is uniquely identified within a bucket by key and version ID.
* **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.
  + Objects will not leave the region unless you explicitly transfer the objects to another region.
* **Data Consistency Model**  
  Amazon S3 replicates the data to multiple servers to achieve high availability.  
  Two types of model:
  + **Read-after-write consistency for PUTS of new objects.**
    - For a PUT request, S3 stores the data across multiple servers to achieve high availability.
    - A process stores an object to S3 and will be immediately available to read the object.
    - A process stores a new object to S3, it will immediately list the keys within the bucket.
    - It does not take time for propagation, the changes are reflected immediately.
  + **Eventual consistency for overwrite PUTS and DELETES**
    - For PUTS and DELETES to objects, the changes are reflected eventually, and they are not available immediately.
    - If the process replaces an existing object with the new object, you try to read it immediately. Until the change is fully propagated, the S3 might return prior data.
    - If the process deletes an existing object, immediately try to read it. Until the change is fully propagated, the S3 might return the deleted data.
    - If the process deletes an existing object, immediately list all the keys within the bucket. Until the change is fully propagated, the S3 might return the list of the deleted key.

Creating an S3 Bucket

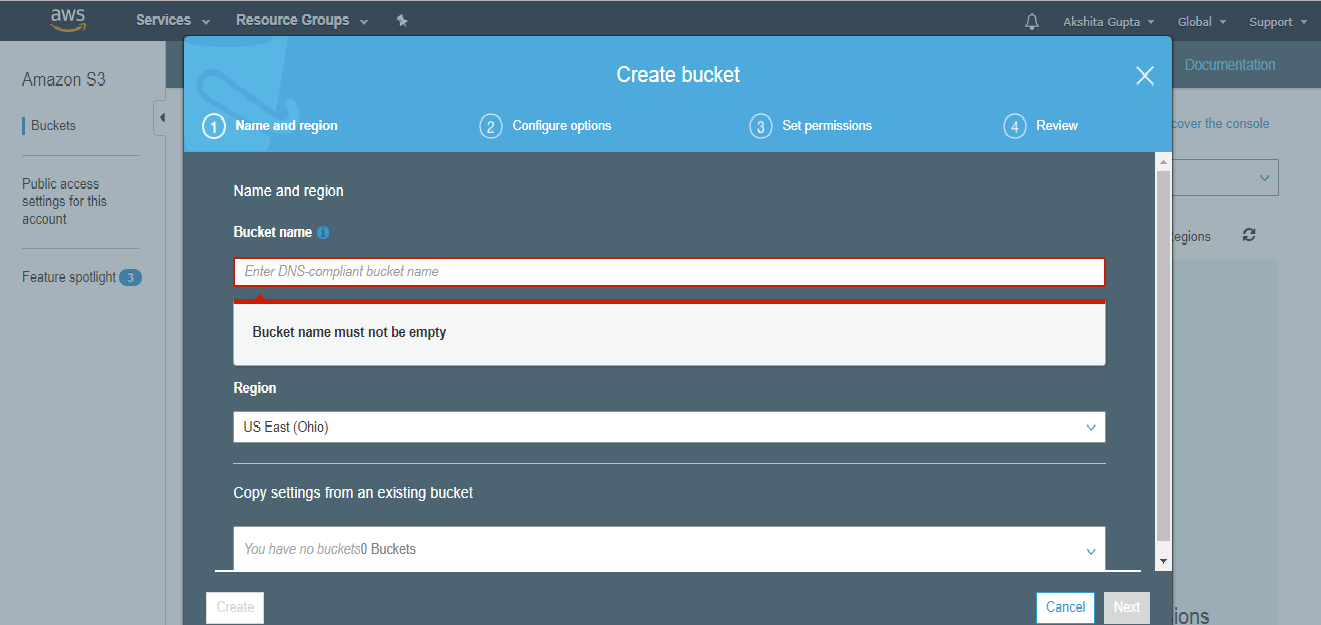
* Sign in to the AWS Management console. After sign in, the screen appears is shown below:



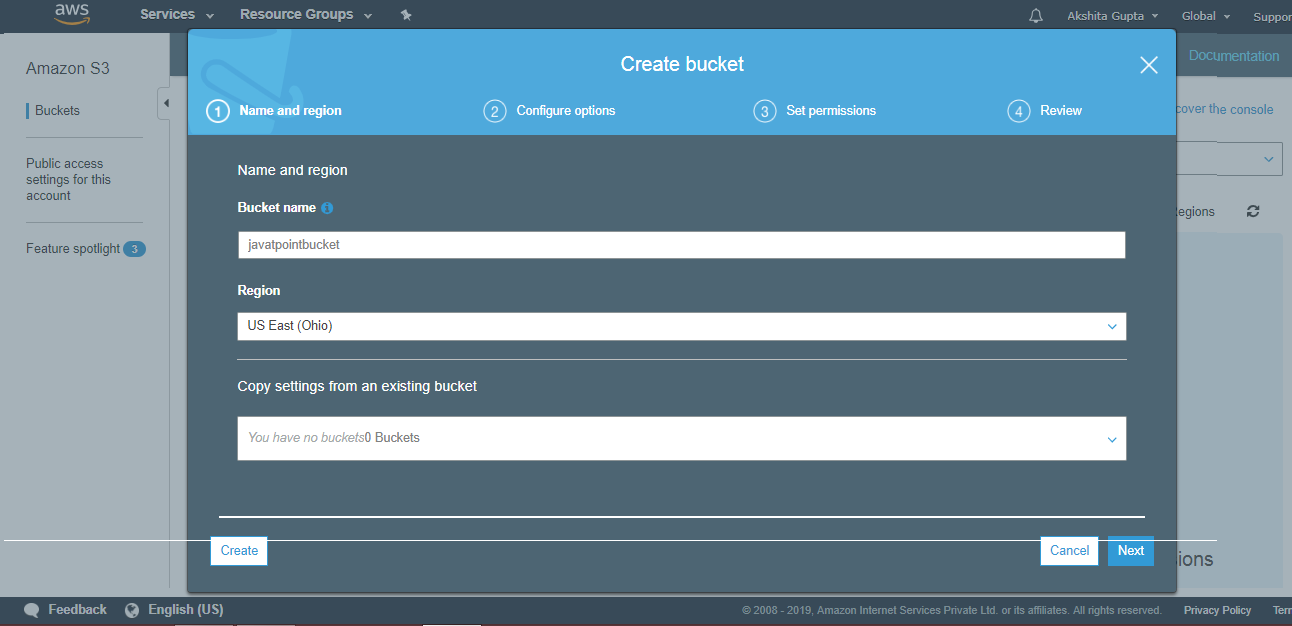
* Move to the S3 services. After clicking on S3, the screen appears is shown below:



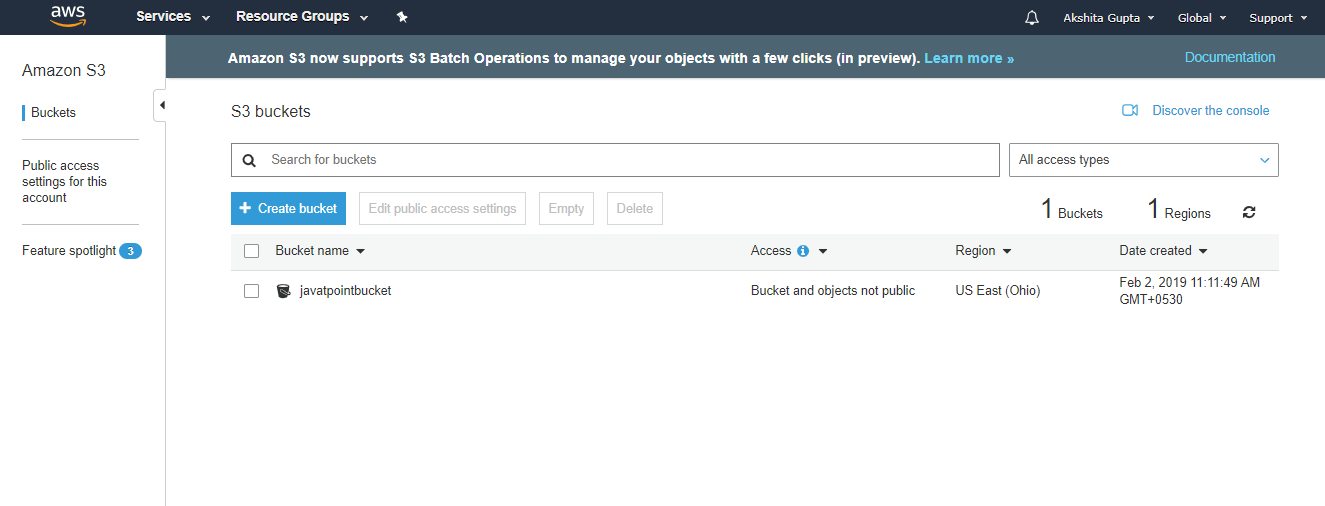
* To create an S3 bucket, click on the "Create bucket". On clicking the "Create bucket" button, the screen appears is shown below:



* Enter the bucket name which should look like DNS address, and it should be resolvable. A bucket is like a folder that stores the objects. A bucket name should be unique. A bucket name should start with the lowercase letter, must not contain any invalid characters. It should be 3 to 63 characters long.

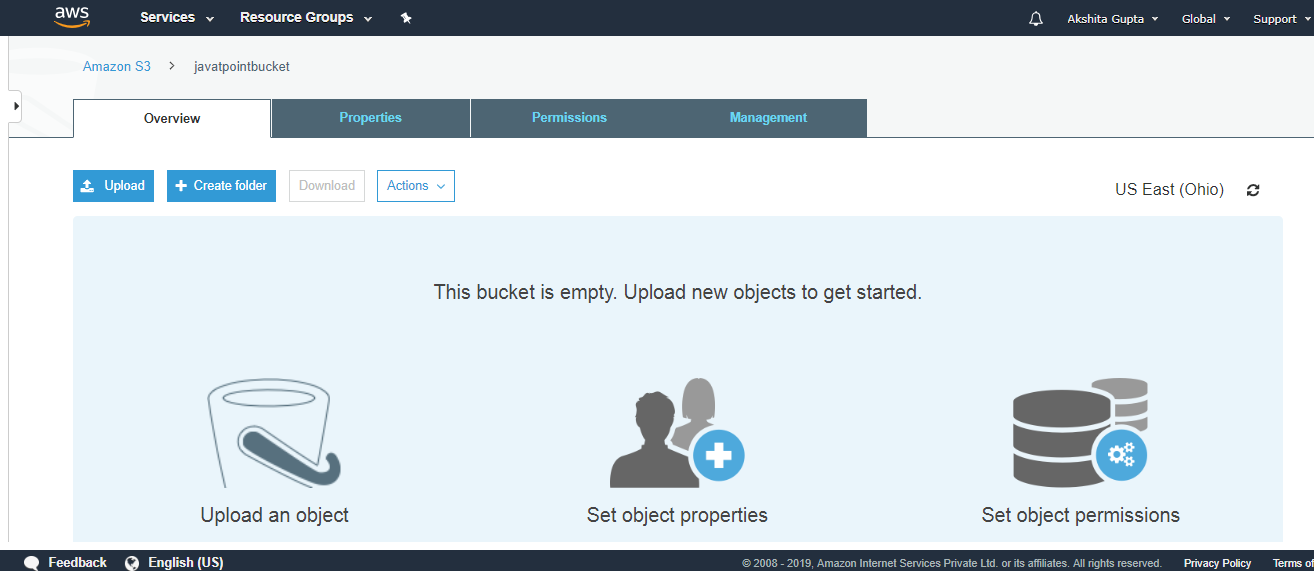


* Click on the "Create" button. Now, the bucket is created.

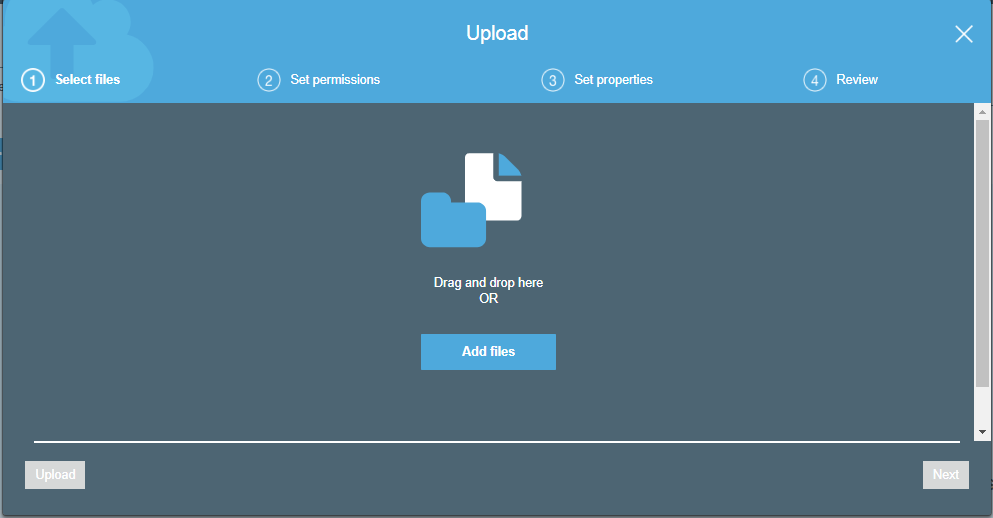


We have seen from the above screen that bucket and its objects are not public as by default, all the objects are private.

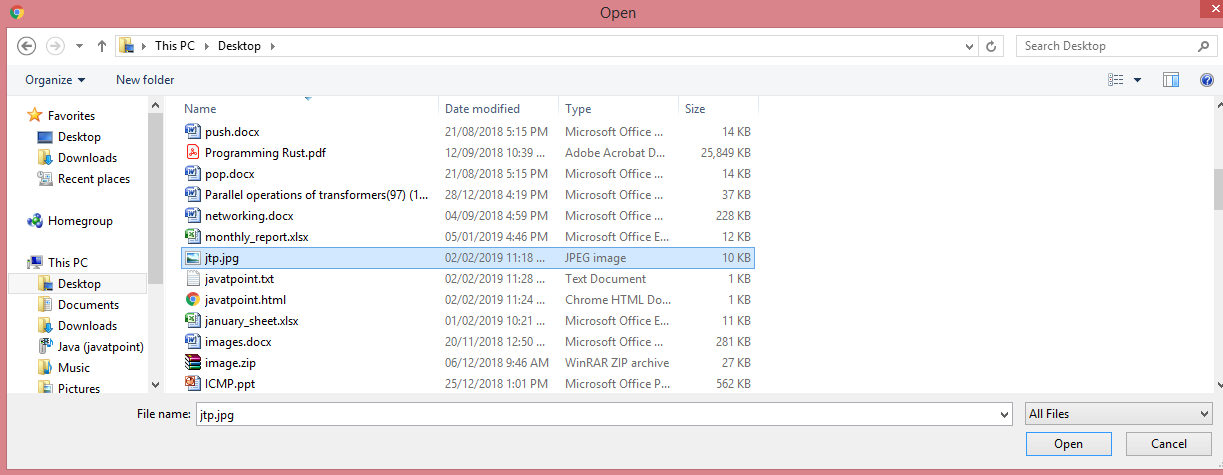
* Now, click on the "javatpointbucket" to upload a file in this bucket. On clicking, the screen appears is shown below:



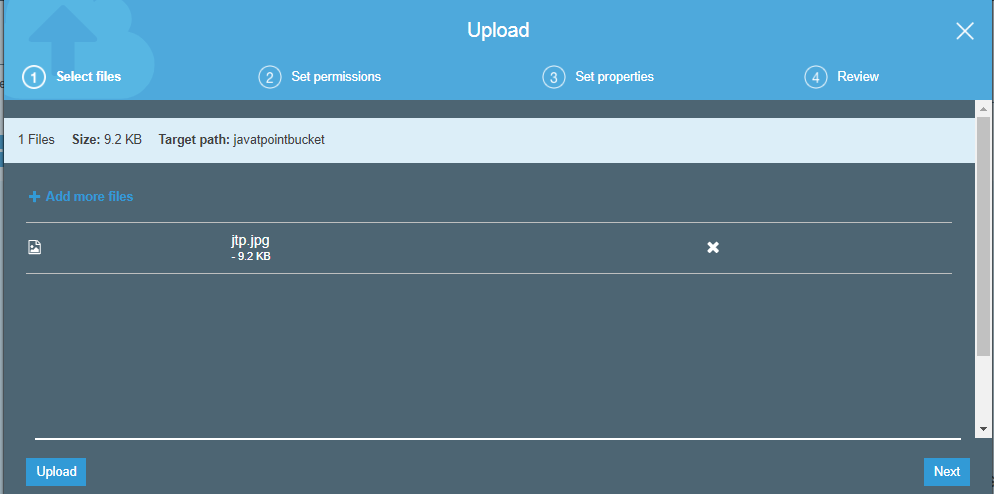
* Click on the "Upload" button to add the files to your bucket.



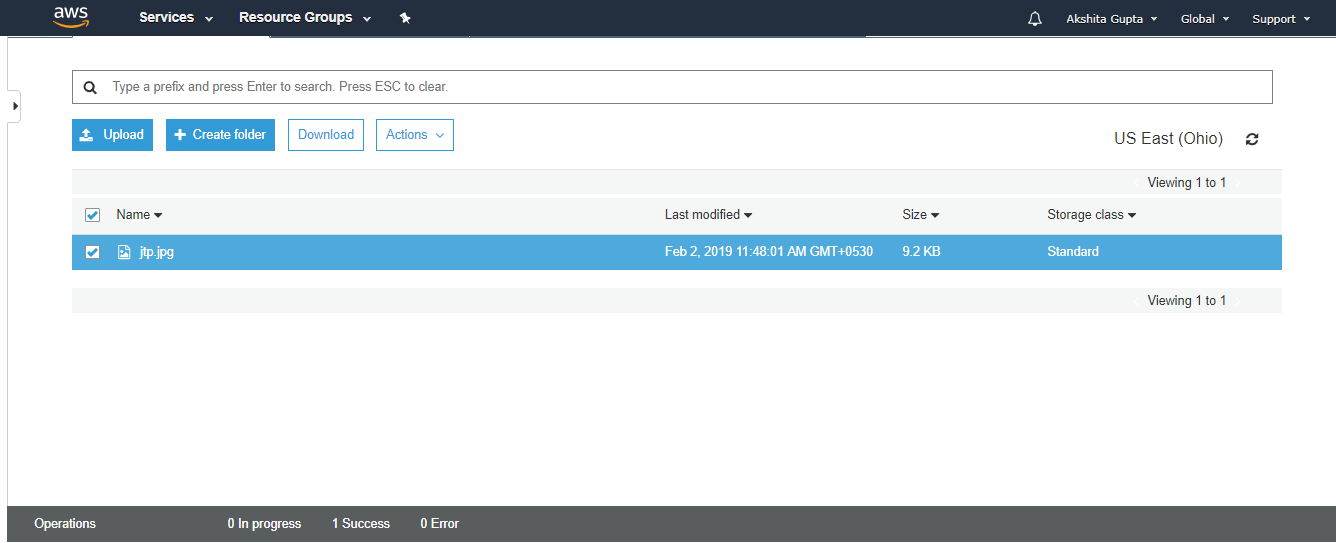
* Click on the "Add files" button.



* Add the **jtp.jpg** file.

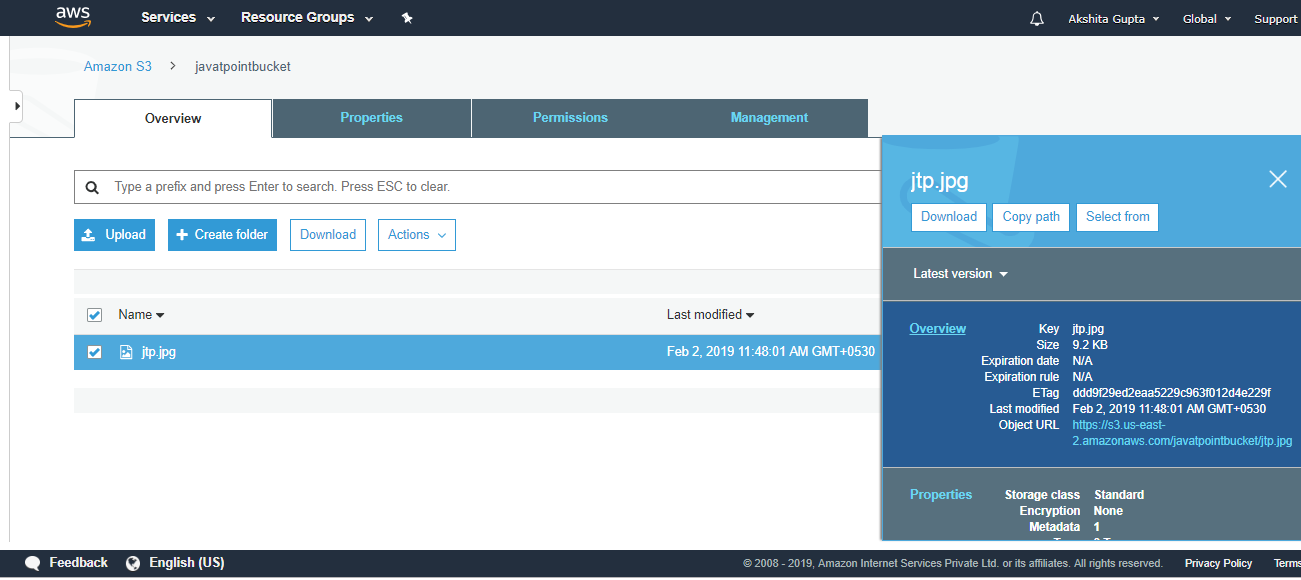


* Click on the "upload" button.

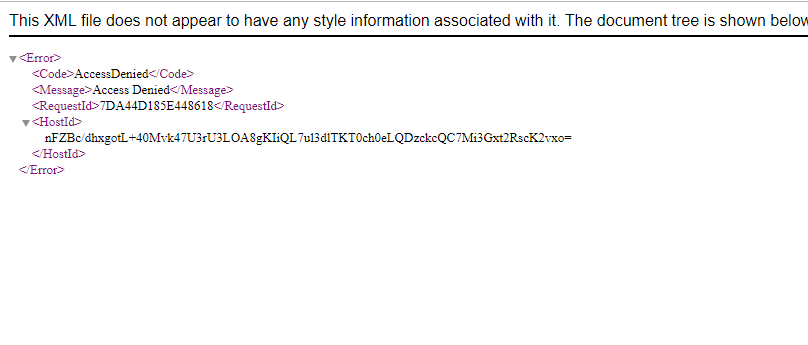


From the above screen, we observe that the "jtp.jpg" has been successfully uploaded to the bucket "javatpoint".

* Move to the properties of the object **"jtp.jpg"** and click on the object URL to run the file appearing on the right side of the screen

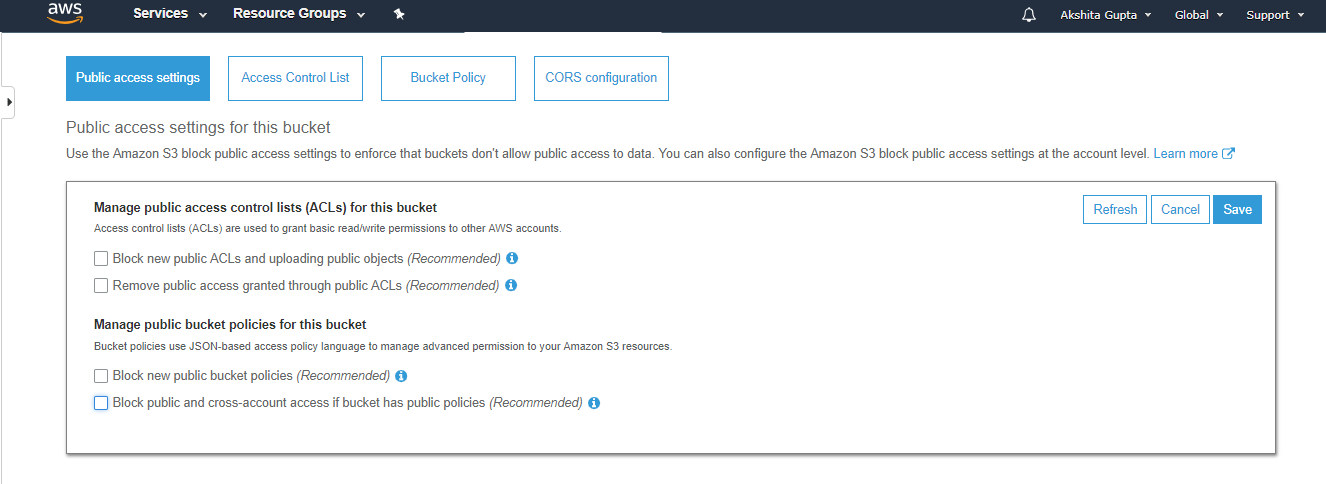


* On clicking the object URL, the screen appears is shown below:

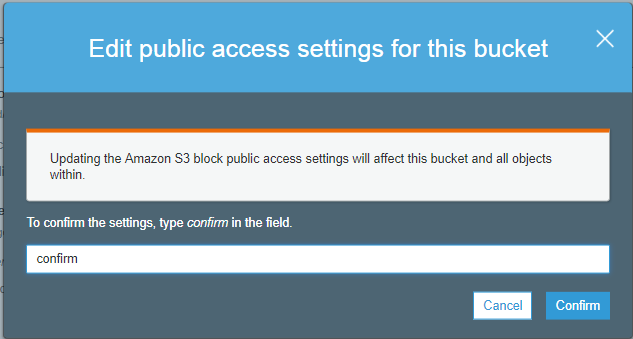


From the above screen, we observe that we are not allowed to access the objects of the bucket.

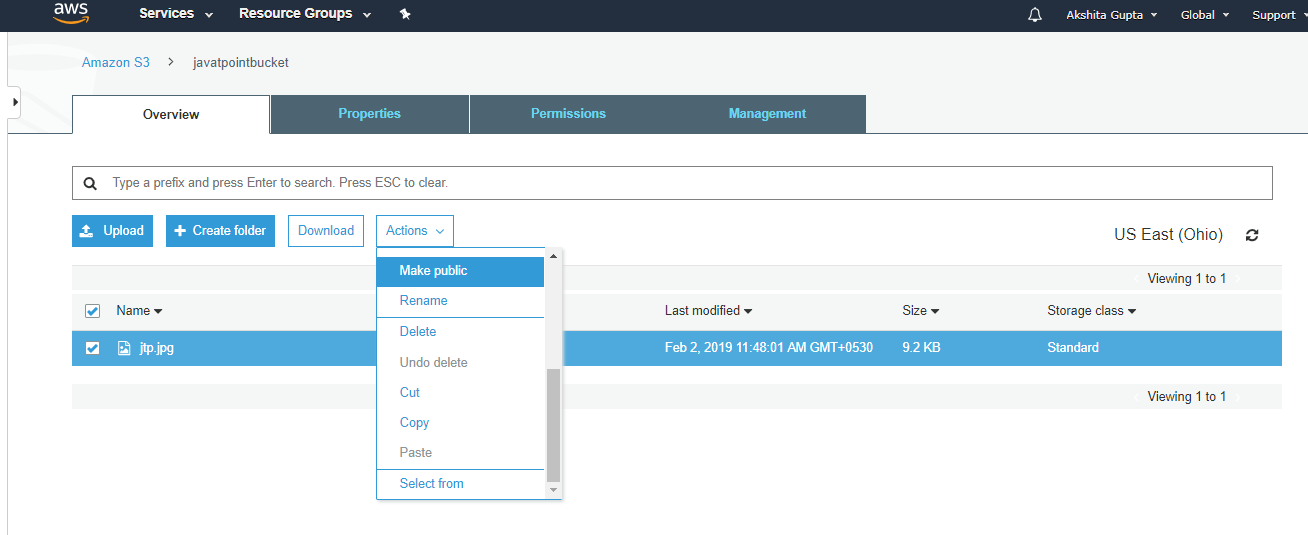
* To overcome from the above problems, we need to set the permissions of a bucket, i.e., "javatpointbucket" and unchecked all of them.



* Save these permissions.
* Enter "confirm" in a textbox, then click on the "confirm" button.



* Click on the **"Actions"** dropdown and then click on the "Make public".



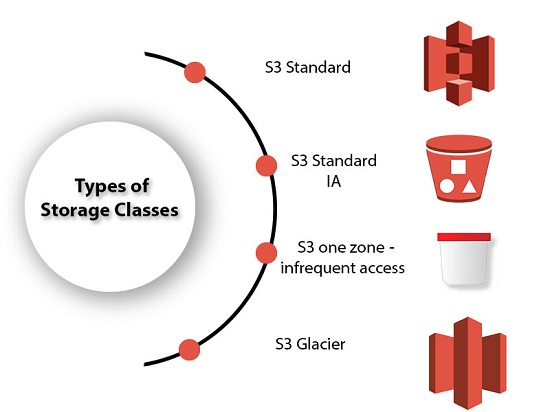
* Now, click on the Object URL of an object to run the file.



Important points to remember

* Buckets are a universal namespace, i.e., the bucket names must be unique.
* If uploading of an object to S3 bucket is successful, we receive a HTTP 200 code.
* S3, S3-IA, S3 Reduced Redundancy Storage are the storage classes.
* Encryption is of two types, i.e., Client Side Encryption and Server Side Encryption
* Access to the buckets can be controlled by using either ACL (Access Control List) or bucket policies.
* By default buckets are private and all the objects stored in a bucket are also private.

AWS Storage Classes



* S3 storage classes are used to assist the concurrent loss of data in one or two facilities.
* S3 storage classes maintain the integrity of the data using checksums.
* S3 provides lifecycle management for the automatic migration of objects for cost savings.

**S3 contains four types of storage classes:**

* S3 Standard
* S3 Standard IA
* S3 one zone-infrequent access
* S3 Glacier

S3 Standard

* Standard storage class stores the data redundantly across multiple devices in multiple facilities.
* It is designed to sustain the loss of 2 facilities concurrently.
* Standard is a default storage class if none of the storage class is specified during upload.
* It provides low latency and high throughput performance.
* It designed for 99.99% availability and 99.999999999% durability

S3 Standard IA

* IA stands for infrequently accessed.
* Standard IA storage class is used when data is accessed less frequently but requires rapid access when needed.
* It has a lower fee than S3, but you will be charged for a retrieval fee.
* It is designed to sustain the loss of 2 facilities concurrently.
* It is mainly used for larger objects greater than 128 KB kept for atleast 30 days.
* It provides low latency and high throughput performance.
* It designed for 99.99% availability and 99.999999999% durability

S3 one zone-infrequent access

* S3 one zone-infrequent access storage class is used when data is accessed less frequently but requires rapid access when needed.
* It stores the data in a single availability zone while other storage classes store the data in a minimum of three availability zones. Due to this reason, its cost is 20% less than Standard IA storage class.
* It is an optimal choice for the less frequently accessed data but does not require the availability of Standard or Standard IA storage class.
* It is a good choice for storing the backup data.
* It is cost-effective storage which is replicated from other AWS region using S3 Cross Region replication.
* It has the same durability, high performance, and low latency, with a low storage price and low retrieval fee.
* It designed for 99.5% availability and 99.999999999% durability of objects in a single availability zone.
* It provides lifecycle management for the automatic migration of objects to other S3 storage classes.
* The data can be lost at the time of the destruction of an availability zone as it stores the data in a single availability zone.

S3 Glacier

* S3 Glacier storage class is the cheapest storage class, but it can be used for archive only.
* You can store any amount of data at a lower cost than other storage classes.
* S3 Glacier provides three types of models:
  + **Expedited:** In this model, data is stored for a few minutes, and it has a very higher fee.
  + **Standard:** The retrieval time of the standard model is 3 to 5 hours.
  + **Bulk:** The retrieval time of the bulk model is 5 to 12 hours.
* You can upload the objects directly to the S3 Glacier.
* It is designed for 99.999999999% durability of objects across multiple availability zones.

**Performance across the Storage classes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **S3 Standard** | **S3 Standard IA** | **S3 One Zone-IA** | **S3 Glacier** |
| **Designed for durability** | 99.99999999% | 99.99999999% | 99.99999999% | 99.99999999% |
| **Designed for availability** | 99.99% | 99.9% | 99.5% | N/A |
| **Availability SLA** | 99.9% | 99% | 99% | N/A |
| **Availability zones** | >=3 | >=3 | 1 | >=3 |
| **Minimum capacity charge per object** | N/A | 128KB | 128KB | 40KB |
| **Minimum storage duration charge** | N/A | 30 days | 30 days | 90 days |
| **Retrieval fee** | N/A | per GB retrieved | per GB retrieved | per GB retrieved |
| **First byte latency** | milliseconds | milliseconds | milliseconds | Select minutes or hours |
| **Storage type** | Object | Object | Object | Object |
| **Lifecycle transitions** | Yes | Yes | Yes | Yes |

# Versioning

**Versioning** is a means of keeping the multiple forms of an object in the same S3 bucket. Versioning can be used to retrieve, preserve and restore every version of an object in S3 bucket.

For example, bucket consists of two objects with the same key but with different version ID's such as photo.jpg (version ID is 11) and photo.jpg (version ID is 12).

Versioning-enabled buckets allow you to recover the objects from the deletion or overwrite. It serves two purposes:

* If you delete an object, instead of deleting the object permanently, it creates a delete marker which becomes a current version of an object.
* If you overwrite an object, it creates a new version of the object and also restores the previous version of the object.

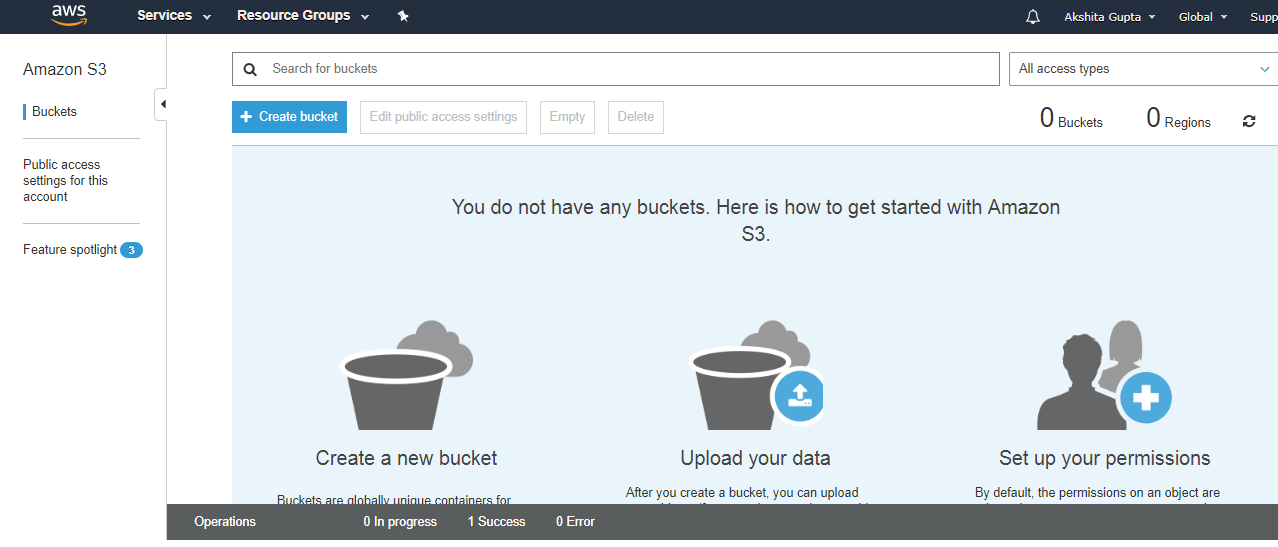
#### **Note: Once you enable the versioning of a bucket, then it cannot be disabled. You can suspend the versioning.**

Versioning state can be applied to all the objects in a bucket. Once the versioning state is enabled, all the objects in a bucket will remain versioned, and they are provided with the unique version ID. **Following are the important points:**

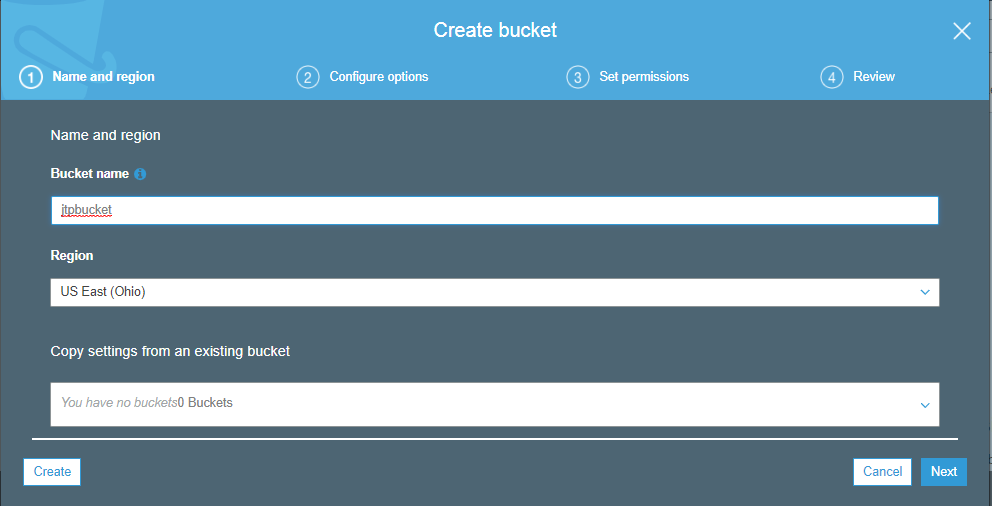
* If the versioning state is not enabled, then the version ID of the objects is set to null. When the versioning is not enabled, existing objects are not changed or are not affected.
* The bucket owner can suspend the versioning to stop the object versions. When you suspend the versioning, existing objects are not affected.

**Let's understand the concept of versioning through an example.**

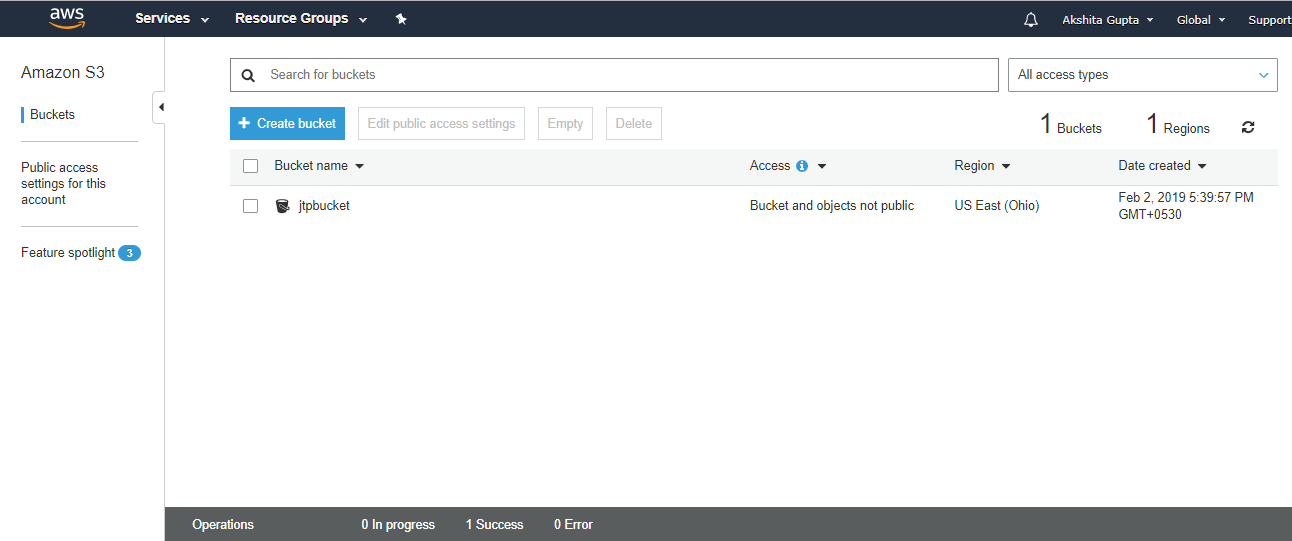
* Sign in to the AWS Management Console.
* Move to the S3 services.
* No, click on the **"Create bucket"** to create a new bucket.



* Enter the bucket name which must be unique.

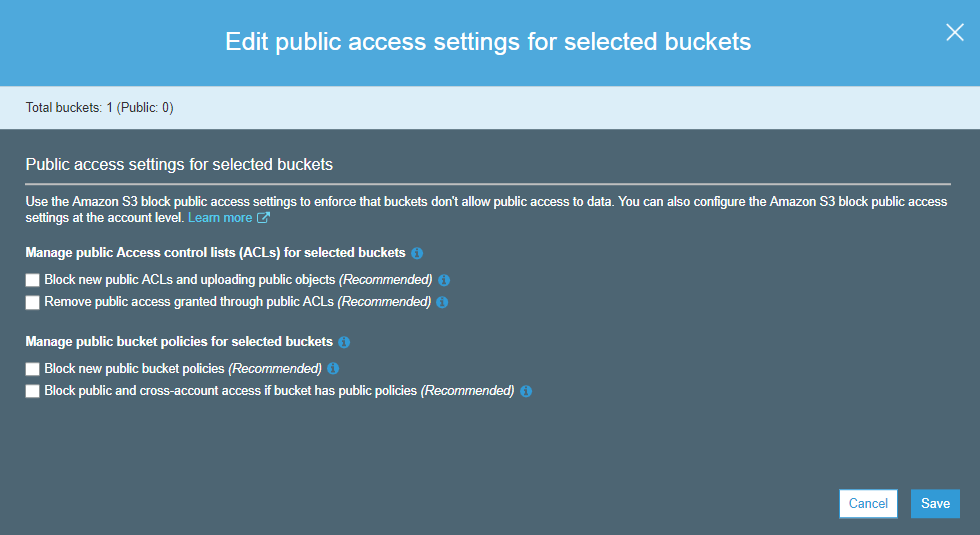


* Click on the **"create"** button.

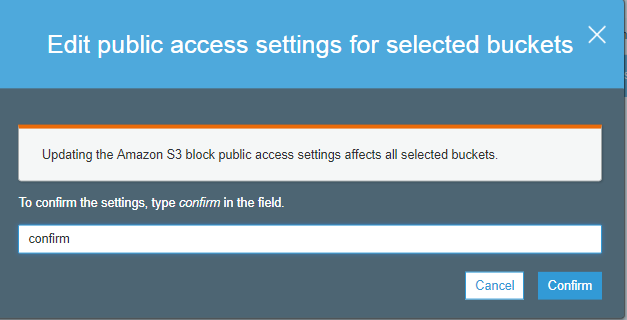


In the above screen, we observe that the bucket **"jtpbucket"** is created with the default settings, i.e., bucket and objects are not public.

* Now, I want to see some objects in a bucket; we need to make a bucket public. Move to the "**Edit public access settings**", uncheck all the settings, and then save the settings.

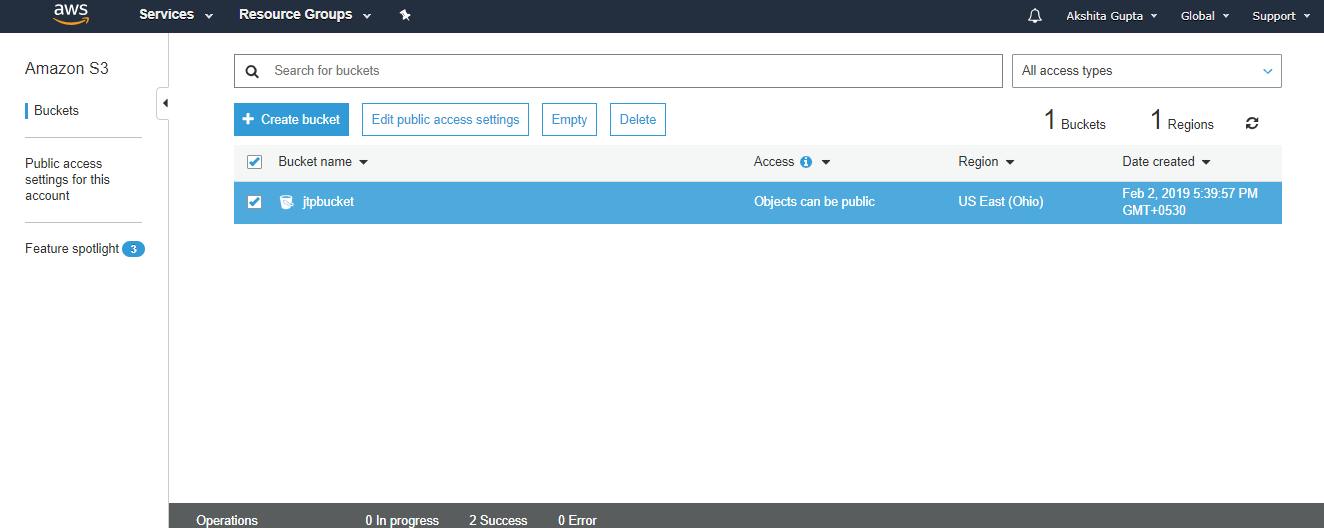


* After saving the settings, the screen appears is shown below:



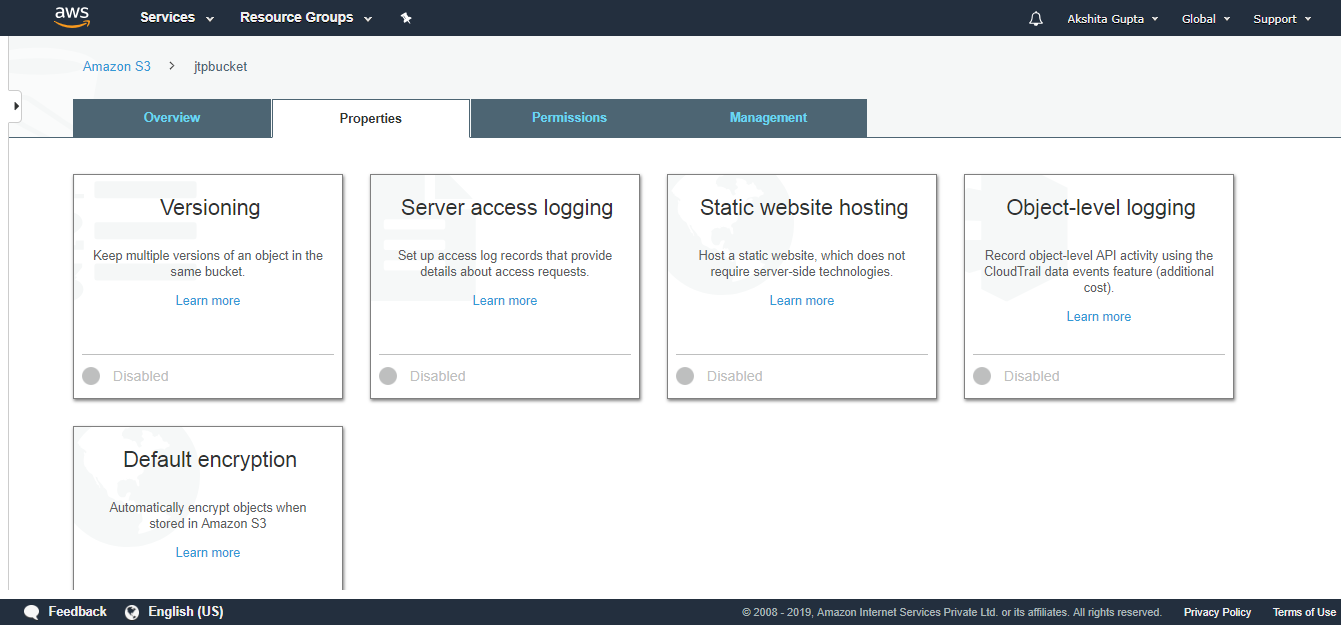
Type the "confirm" in a textbox to confirm the settings. Click on the **"confirm"** button.

* When the settings are confirmed, the screen appears as shown below:

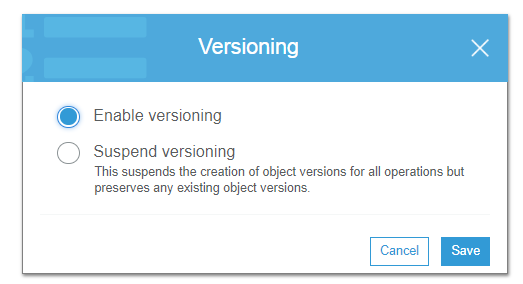


The above screen shows that the objects in a bucket have become public.

* Now, we add the versioning to our bucket. Move to the properties of a bucket, i.e., **jtpbucket** and click on the versioning.

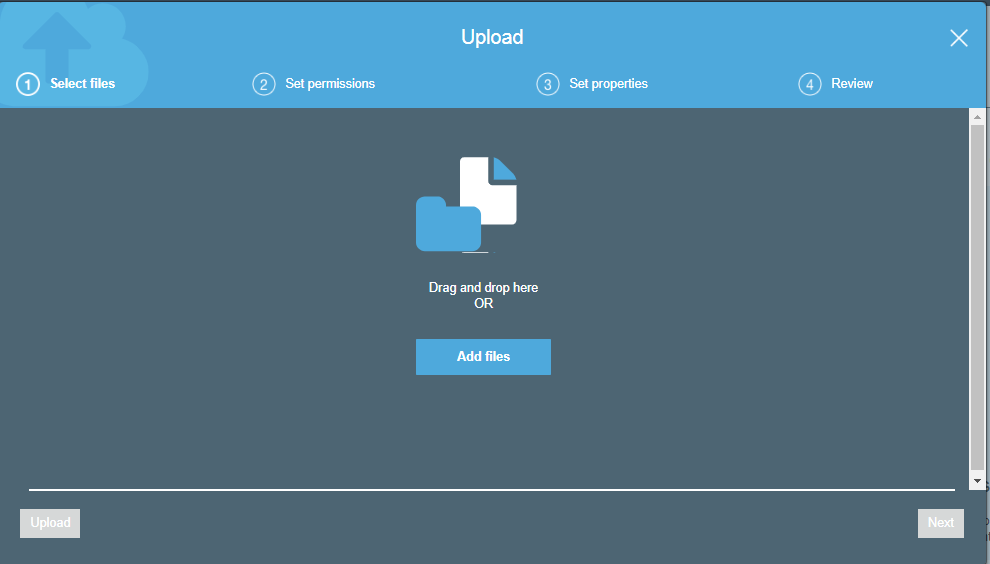


* On clicking on the versioning, the screen appears as shown below:

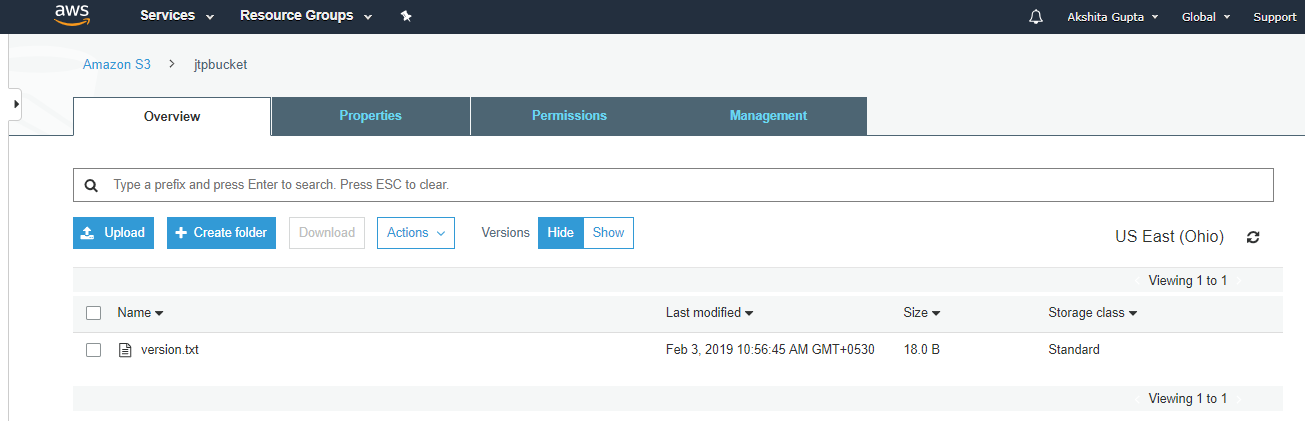


We can either enable or suspend the versioning. Suppose we enable the versioning and save this setting, this adds the versioning to our bucket.

* We now upload the files to our bucket.

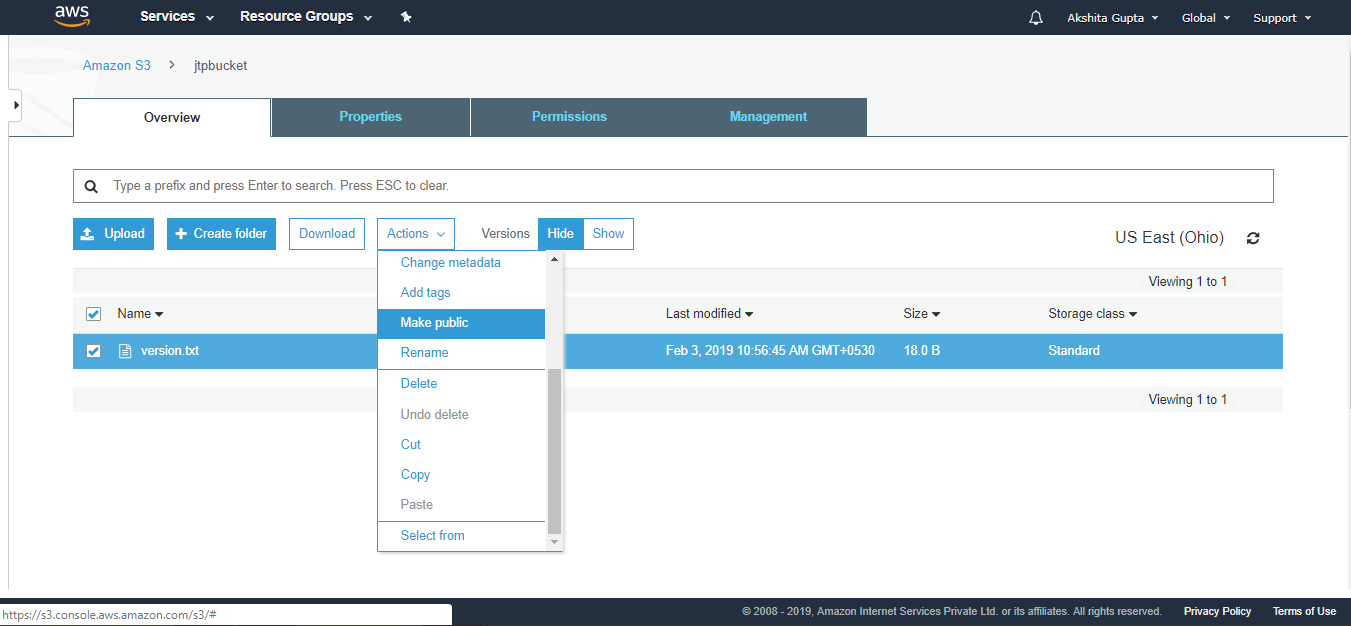


* Now, we click on the **"Add files"** to add the files in our bucket. When a file is uploaded, the screen appears as shown below:



In the above screen, we observe that version.txt file is uploaded.

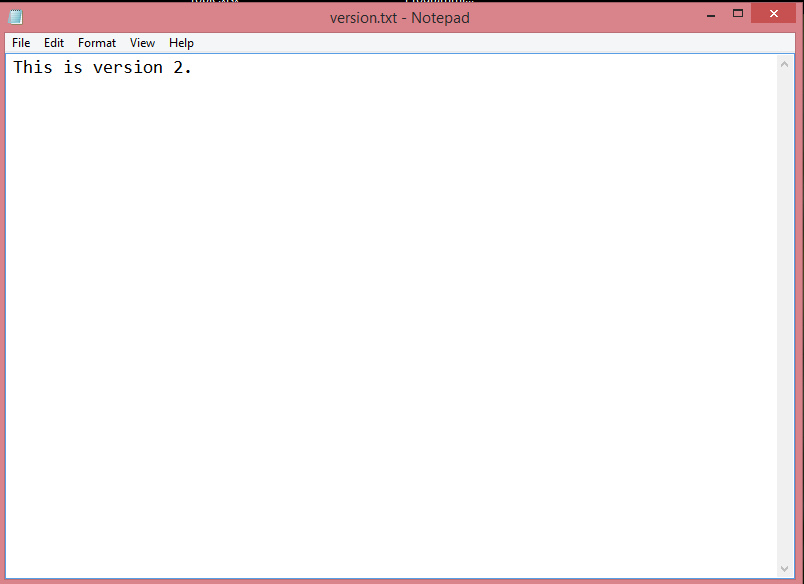
* To run the **version.txt** file, we have to make it public from the **Actions** dropdown menu.



* When a file becomes public, we can run the file by clicking on its object URL. On clicking on the object URL, the screen appears as shown below:

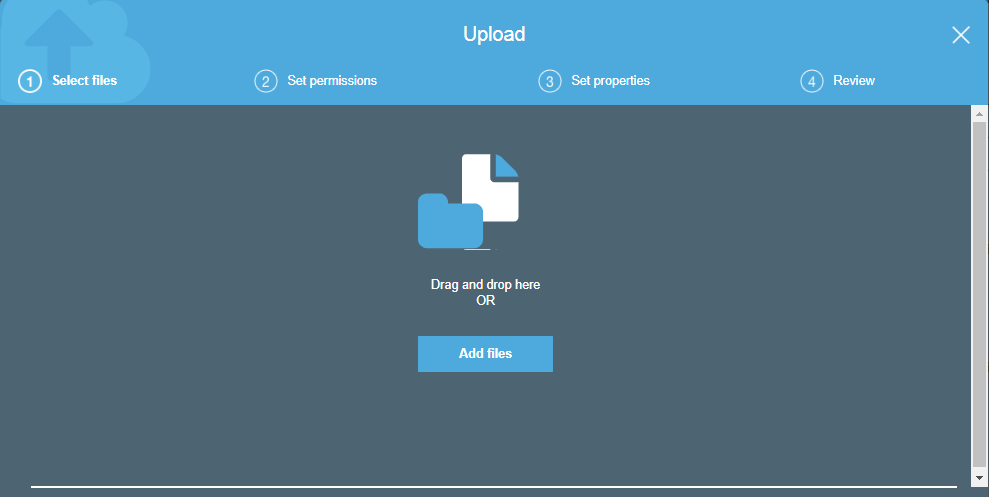
AWS Versioning

* Now, we create the second version of the file. Suppose I change the content of the file and re-upload it, then it becomes the second version of the file.

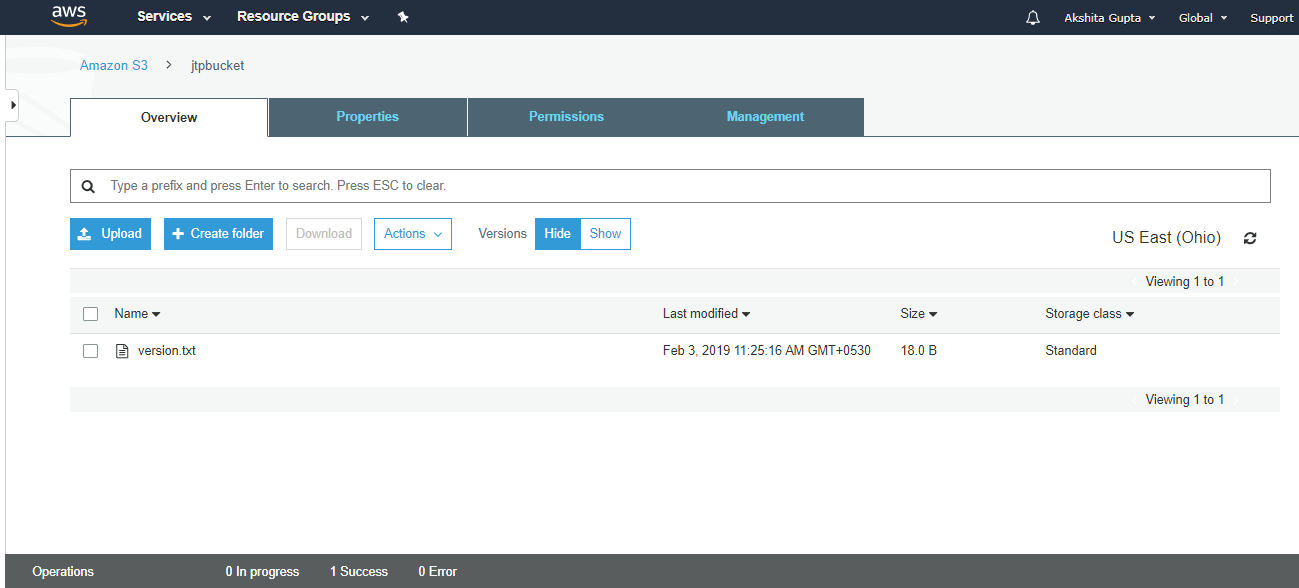


In the above screen, we change the content from "version 1" to "version 2" and then save the file.

* Now, we upload the above file to our bucket.

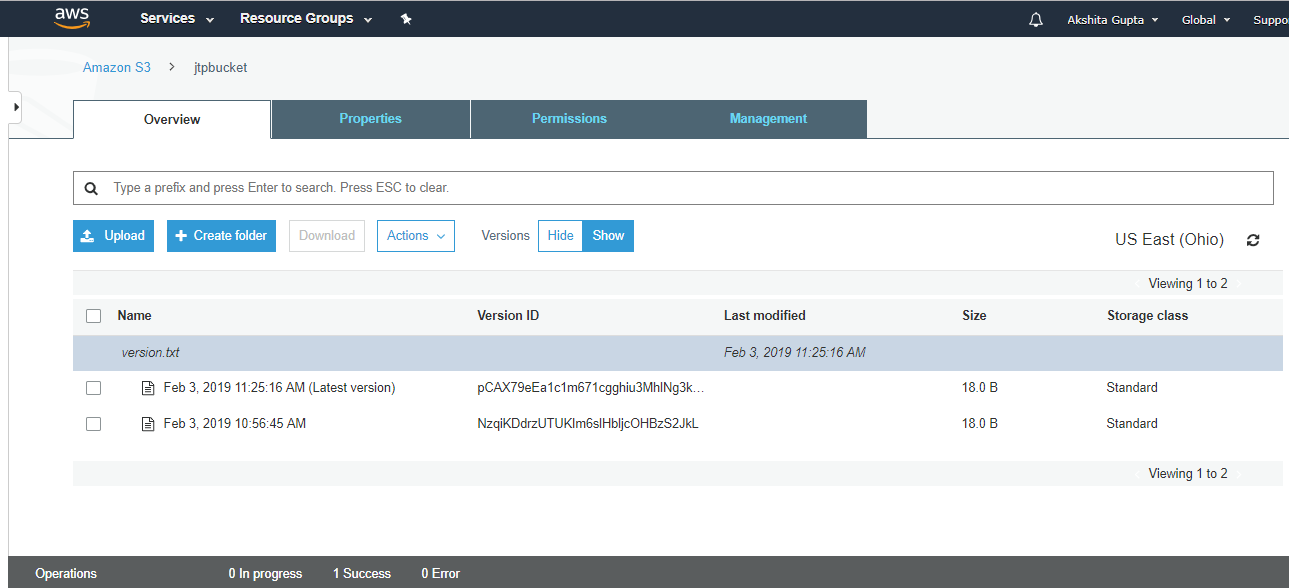


* After uploading the file, two versions of a file are created.



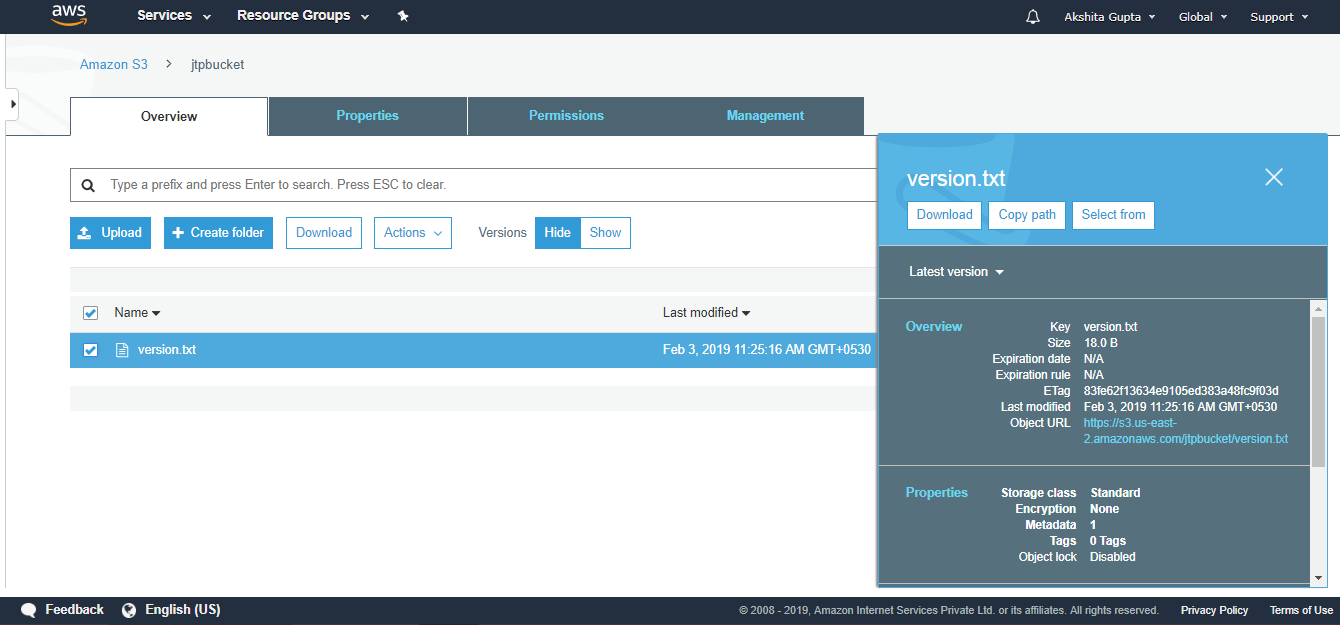
From the above screen, we observe that either we can hide or show the versions.

* When we click on the "show", we can see all the versions of a file.

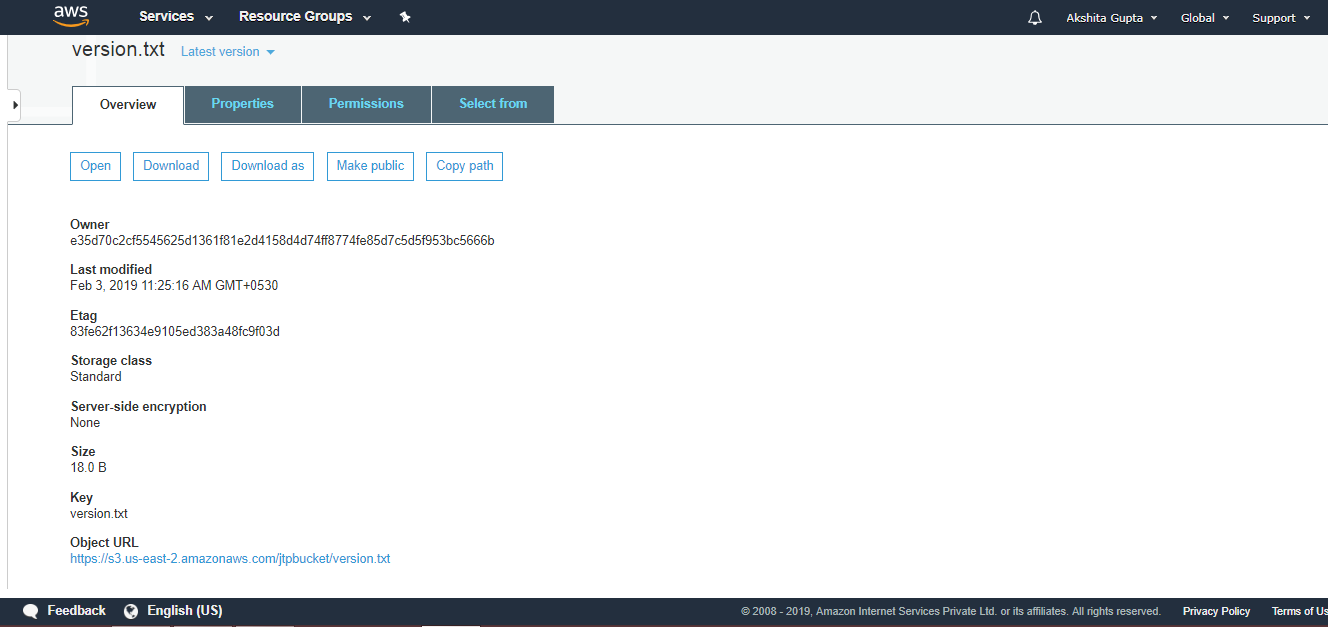


From the above screen, we can see both the versions of a file and currently uploaded file become the latest version. Both the files are of same size, i.e., 18.0 B and storage class, i.e., Standard.

* To run the **version.txt** file, we have to make it public from the **Actions** dropdown menu.
* Now, move to the properties of a file and click on the object URL.



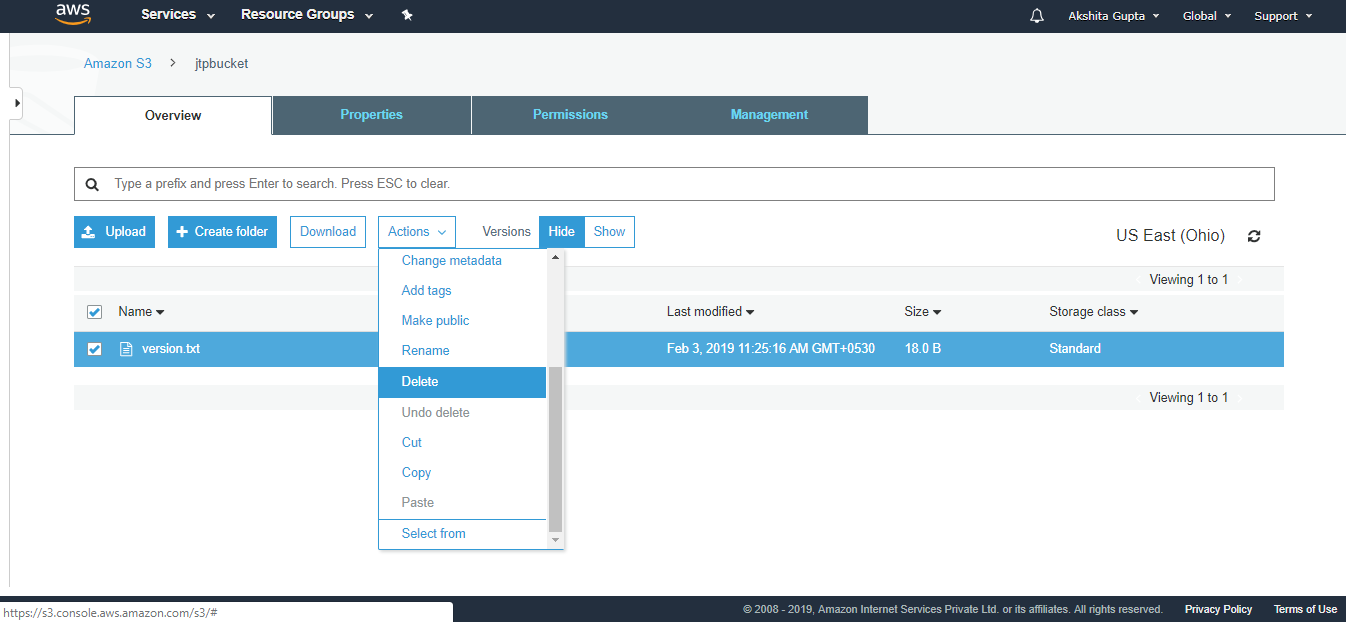
Click on the Object URL.



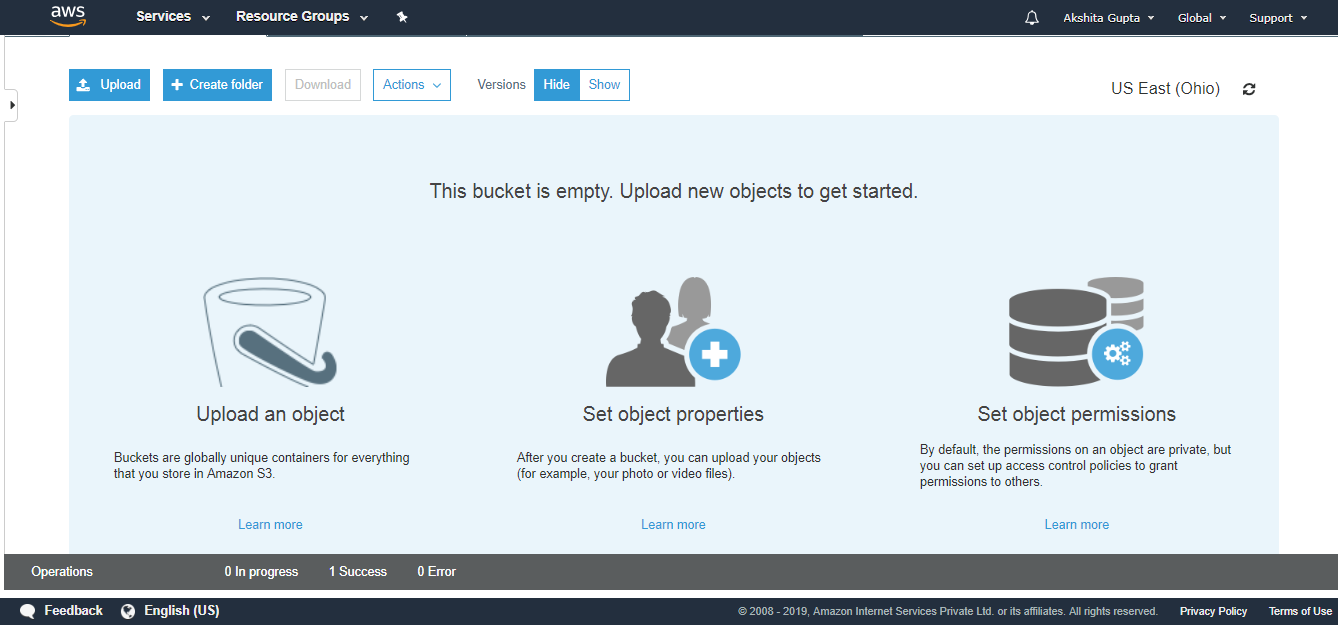
* On clicking on the Object URL, we can see the output, i.e., the content of the currently uploaded file.

AWS Versioning

* Now, we delete an object. Move to the Actions dropdown menu and click on the Delete.

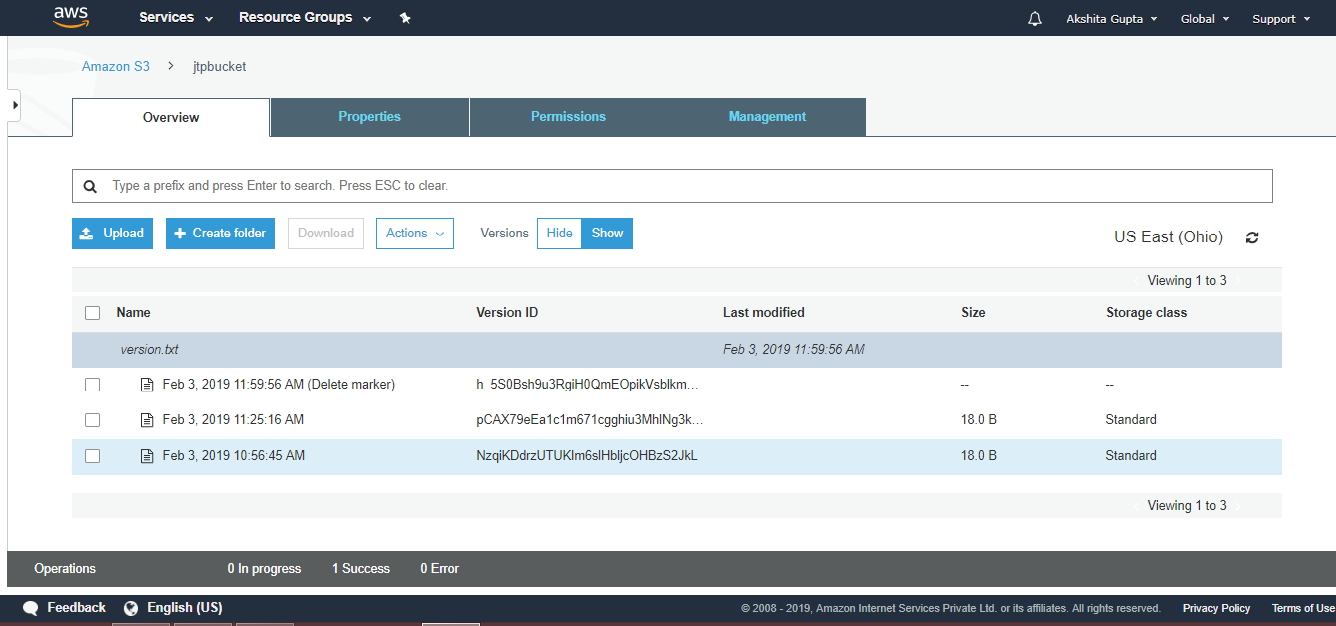


* On deleting the object, the screen appears as shown below:



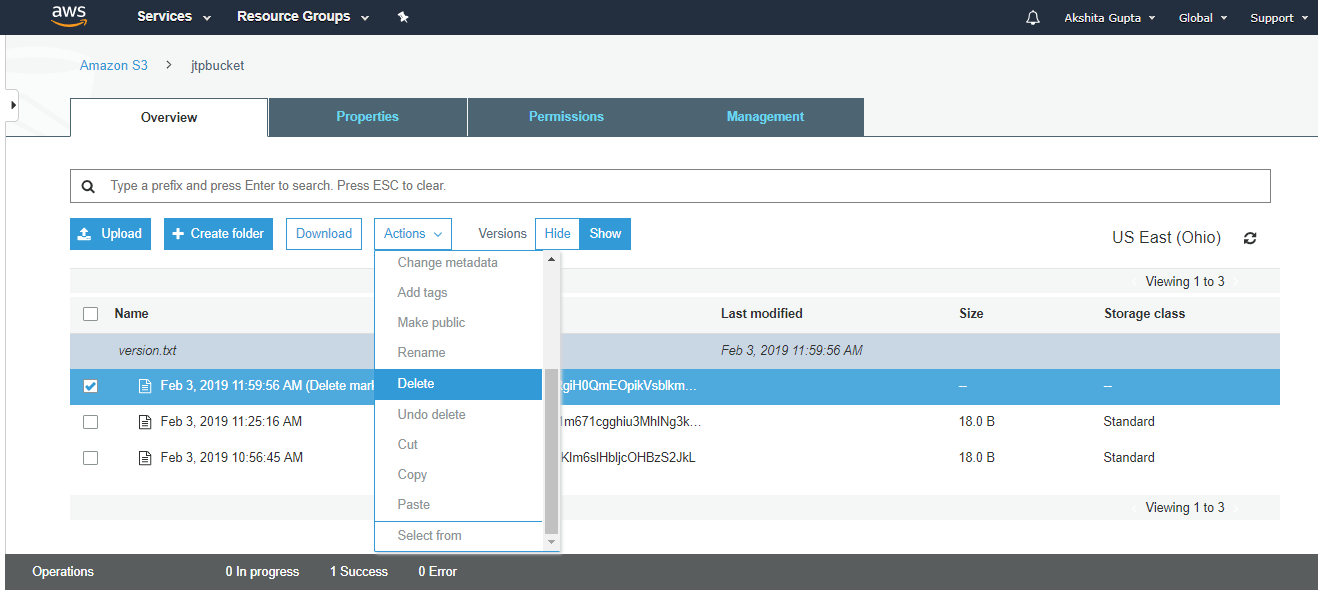
We observe that the bucket becomes empty.

* However, when we click on the **Show** Version, we can see all the versions of a file, i.e., Delete marker and other two versions of a file.

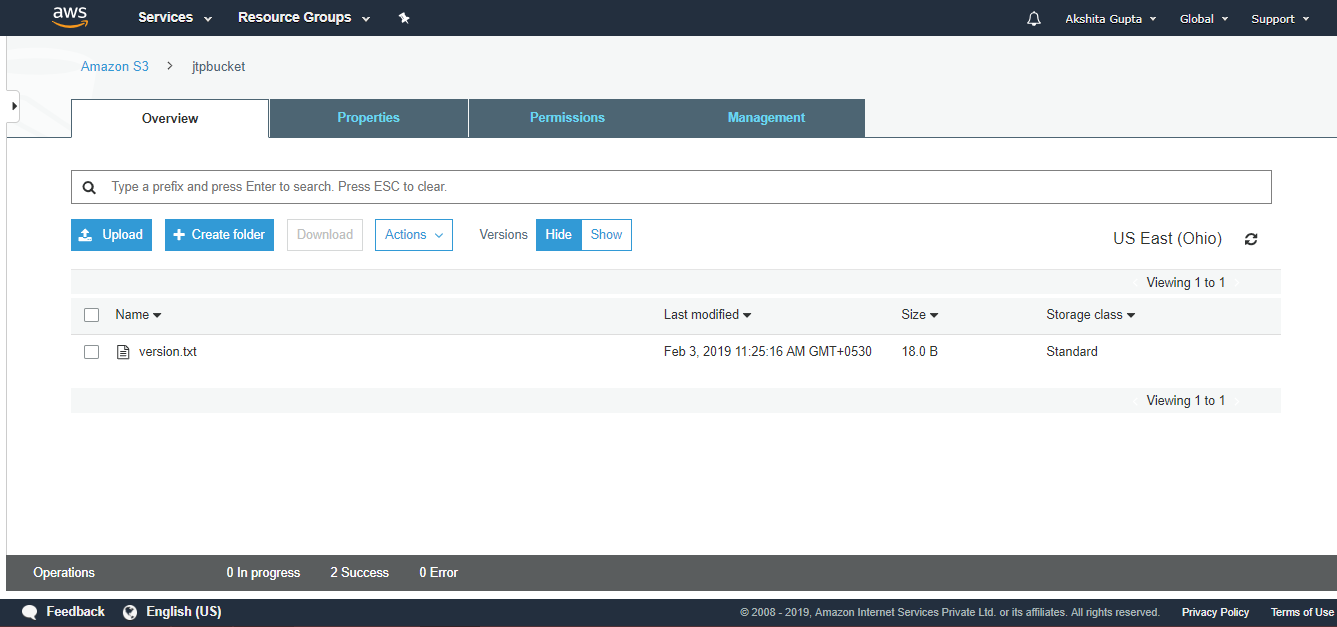


We observe from the above screen that the object is not permanently deleted; it has been restored. Therefore, the versioning concept is used to restore the objects.

* If you want to restore the object, delete the **"delete marker" by clicking on the Actions dropdown menu and click on the Delete**.



* Click on the "Hide" Versions, we will observe that the file has been restored.



### Important points to be remembered:M

* It stores all versions of an object (including all writes and even if you delete an object).
* It is a great backup tool.
* Once the versioning enabled, it cannot be disabled, only suspended.
* It is integrated with lifecycle rules.
* Versioning's MFA Delete capability uses multi-factor authentication that can be used to provide the additional layer of security.

# 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 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**

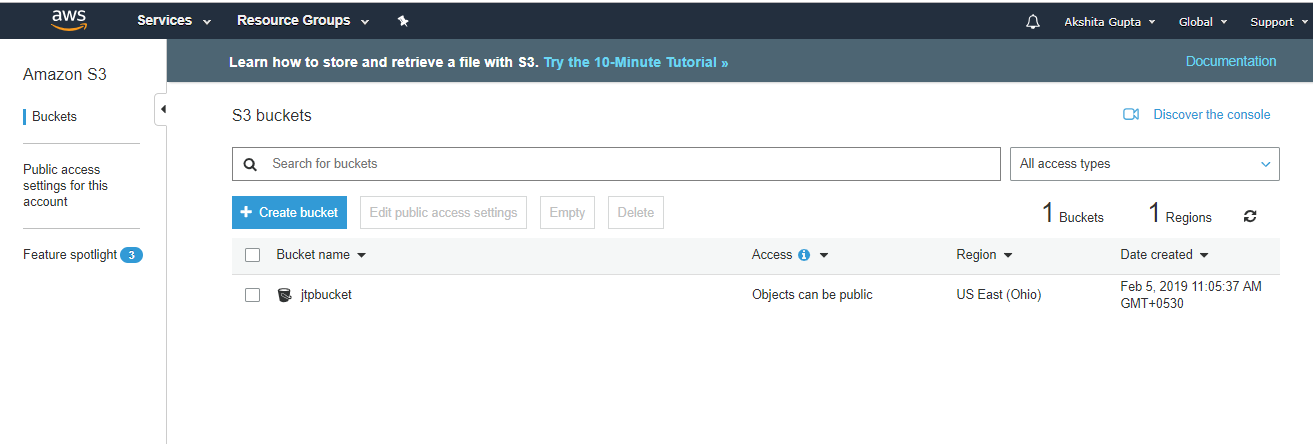
* **Create two buckets:** Create two buckets within AWS Management Console, where one bucket is a source bucket, and other is a destination bucket.
* **Enable versioning:** Cross Region Replication can be implemented only when the versioning of both the buckets is enabled.
* **Amazon S3 encrypts the 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.

**Use cases of Cross Region Replication**

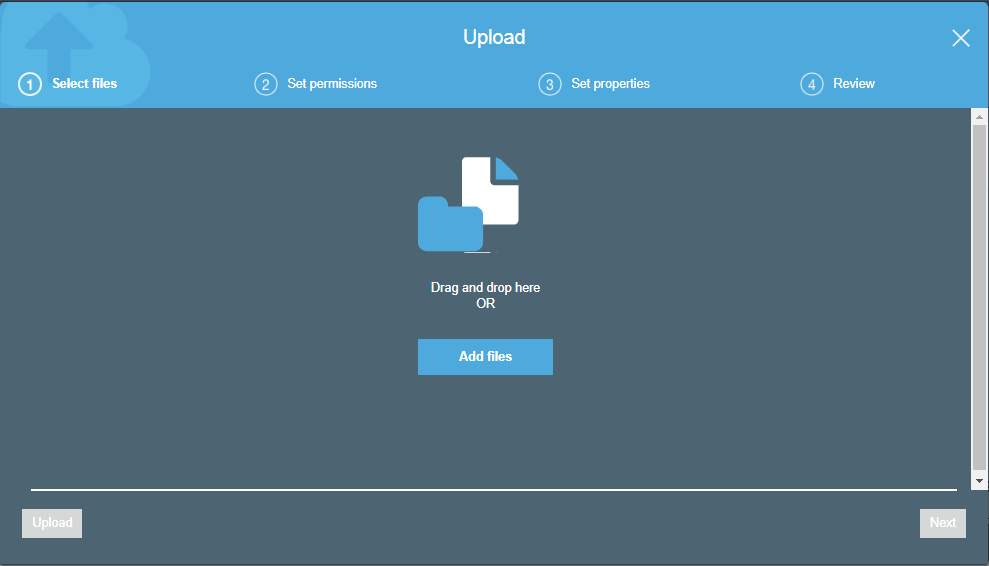
* **Compliance Requirements**  
  By default, Amazon S3 stores the data across different geographical regions or availability zone to have the availability of data. Sometimes there could be compliance requirements that you want to store the data in some specific region. Cross Region Replication allows you to replicate the data at some specific region to satisfy the requirements.
* **Minimize Latency**  
  Suppose your customers are in two geographical regions. To minimize latency, you need to maintain the copies of data in AWS region that are geographically closer to your users.
* **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.

**Let's understand the concept of Cross Region Replication through an example.**

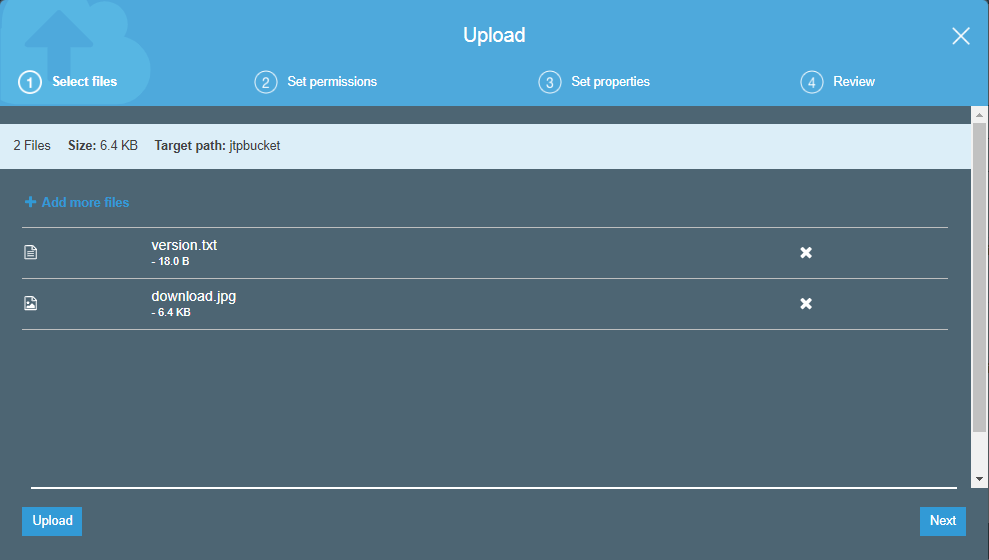
* Sign in to the AWS Management Console.
* Now, we upload the files in a **jtpbucket**. The jtpbucket is an s3 bucket created by us.



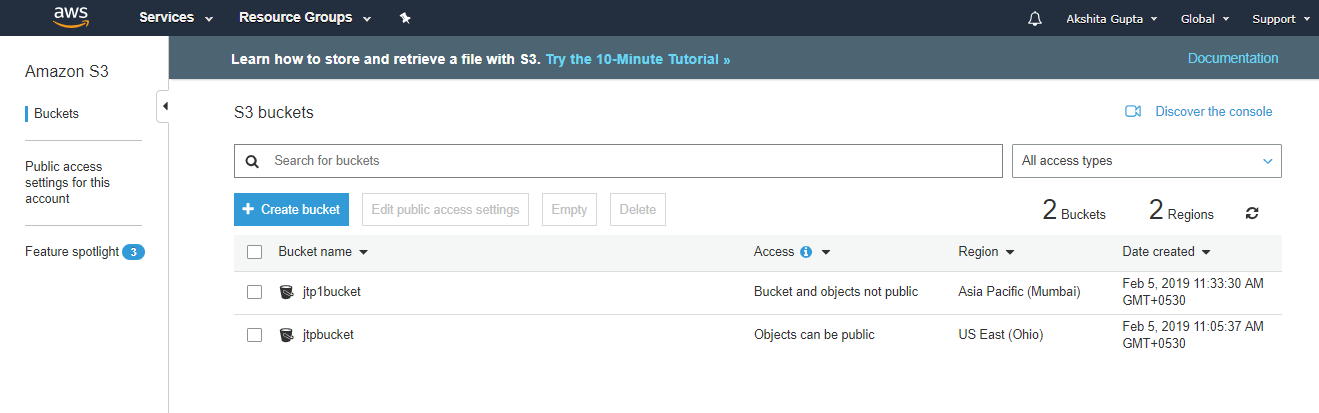
* Add the files in a bucket.



* Now, we add two files in a bucket, i.e., version.txt and download.jpg.

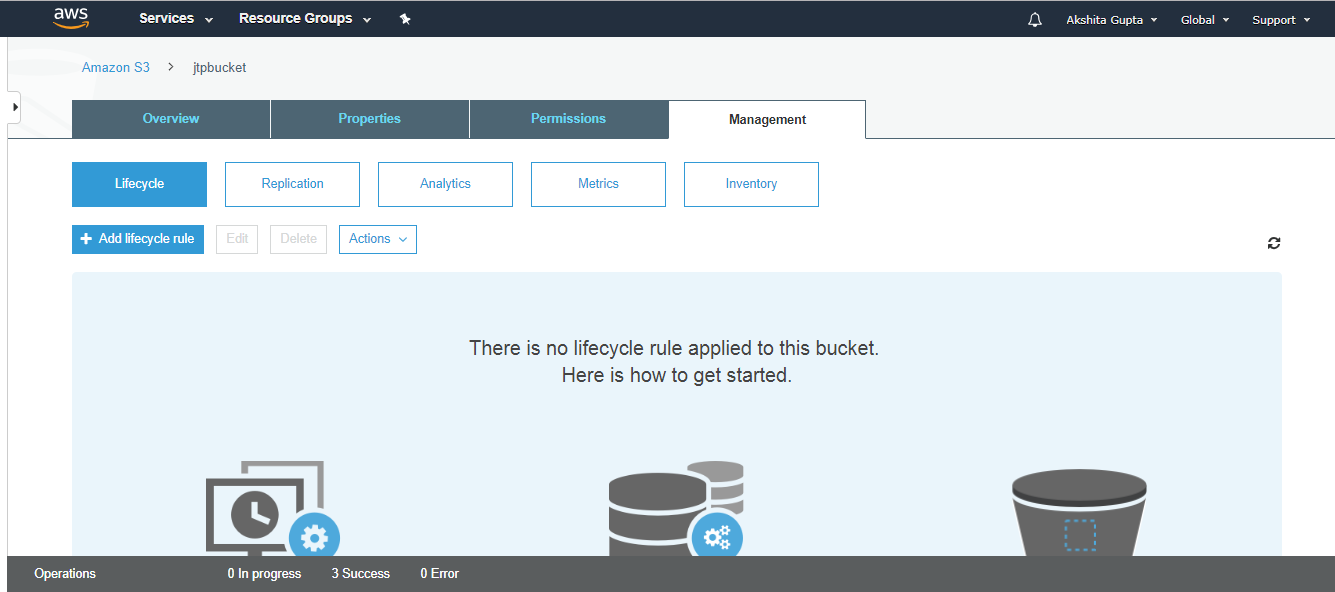


* Now, we create a new bucket whose name is **jtp1bucket** with a different region.

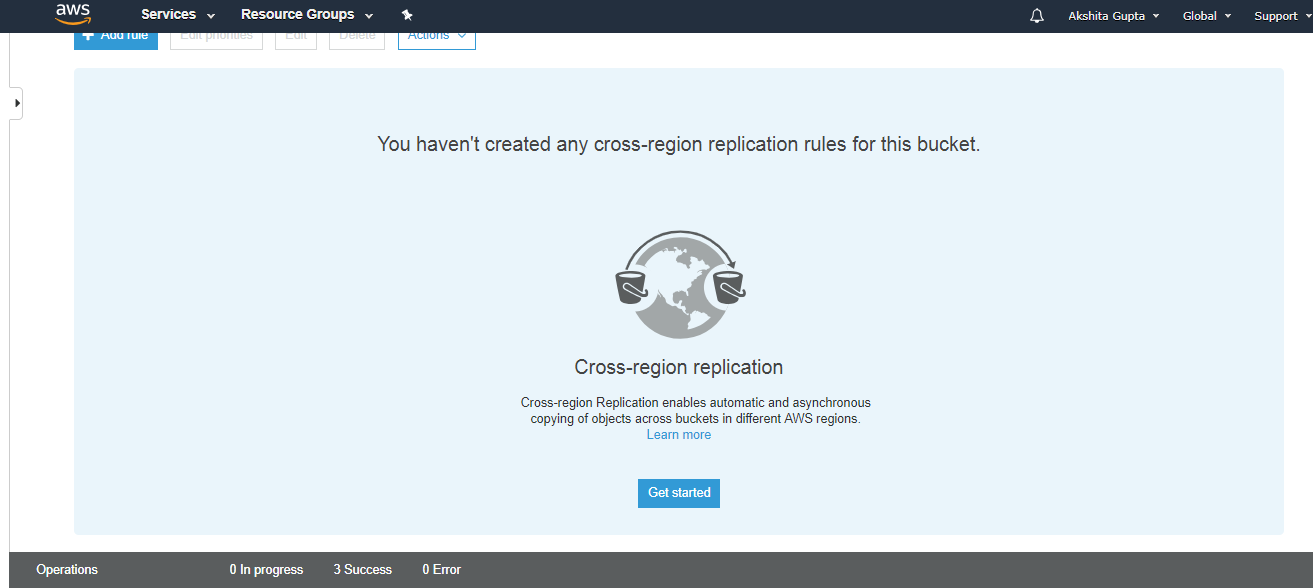


Now, we have two buckets, i.e., jtpbucket and jtp1bucket in s3.

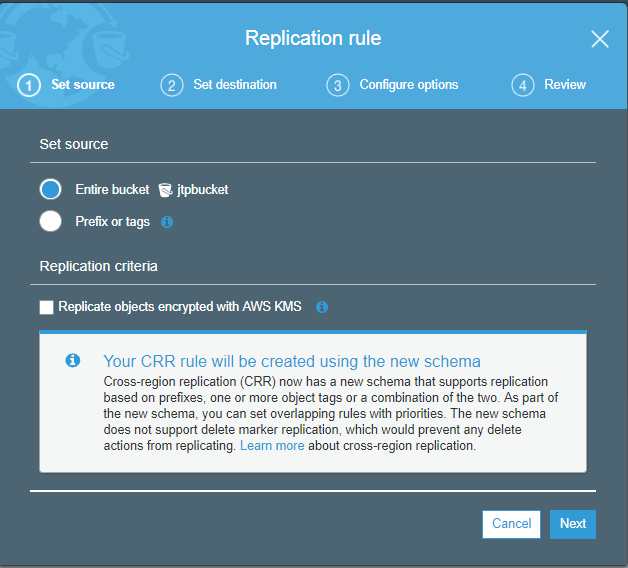
* Click on the **jtpbucket** and then move to the Management of the **jtpbucket**.



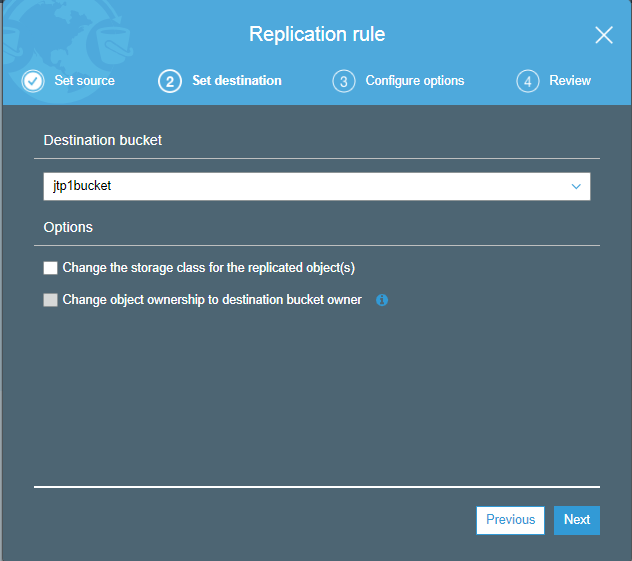
* Click on the **Replication**. On clicking, the screen appears as shown below:



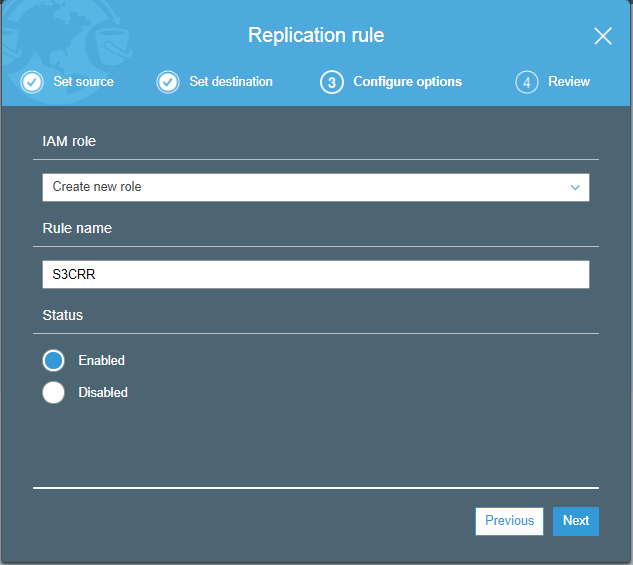
* Click on the **Get started** button.
* Enable the versioning of both the buckets.
* You can either replicate the entire bucket or tags to the destination bucket. Suppose you want to replicate the entire bucket and then click on the Next.



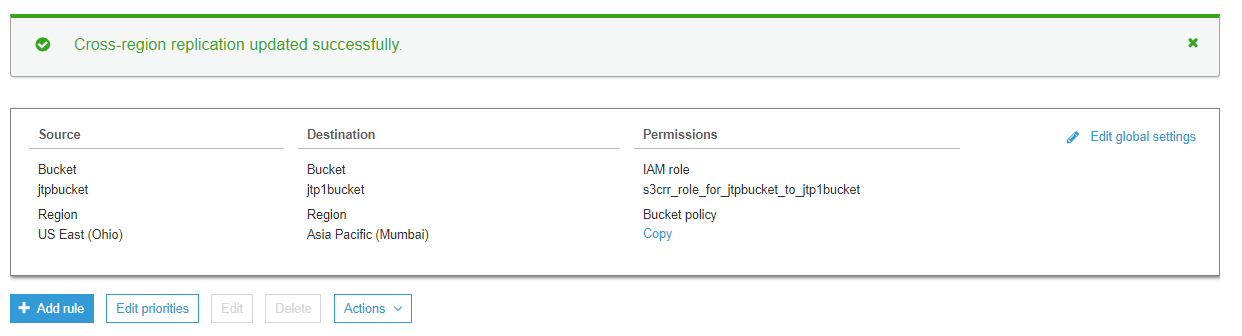
* Enter your destination bucket, i.e., jtp1bucket.



* Create a new IAM role, and the role name is S3CRR and then click on the Next.

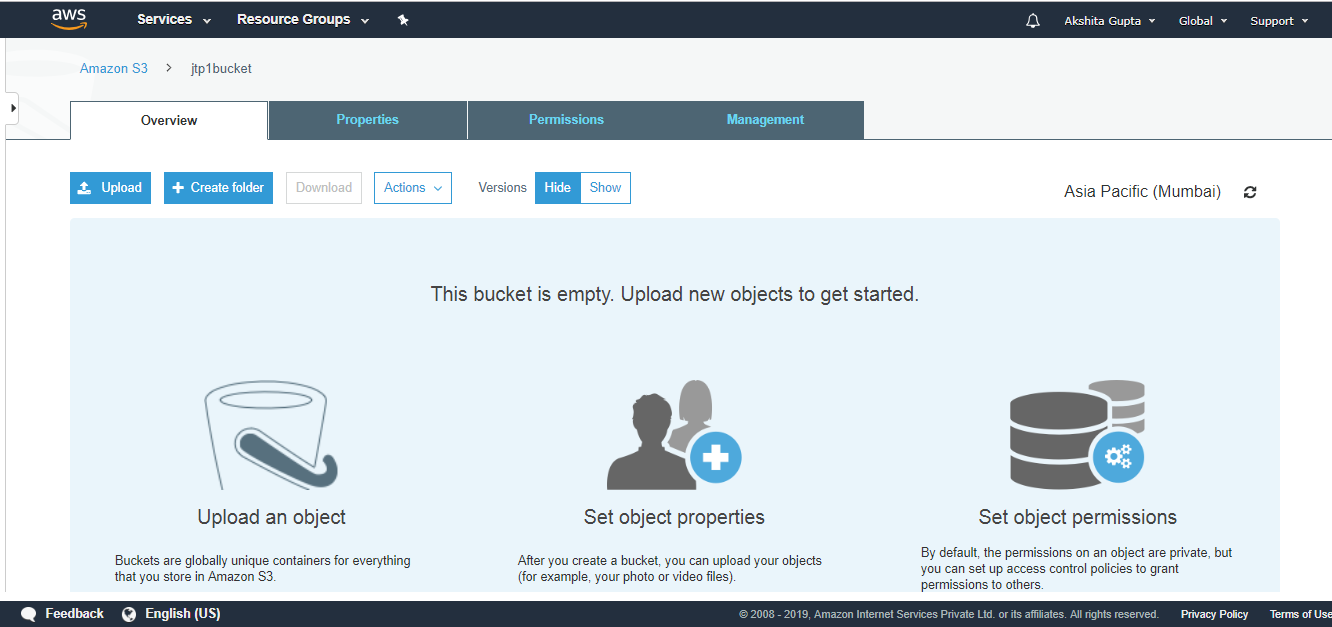


* After saving the settings, the screen appears as shown below:



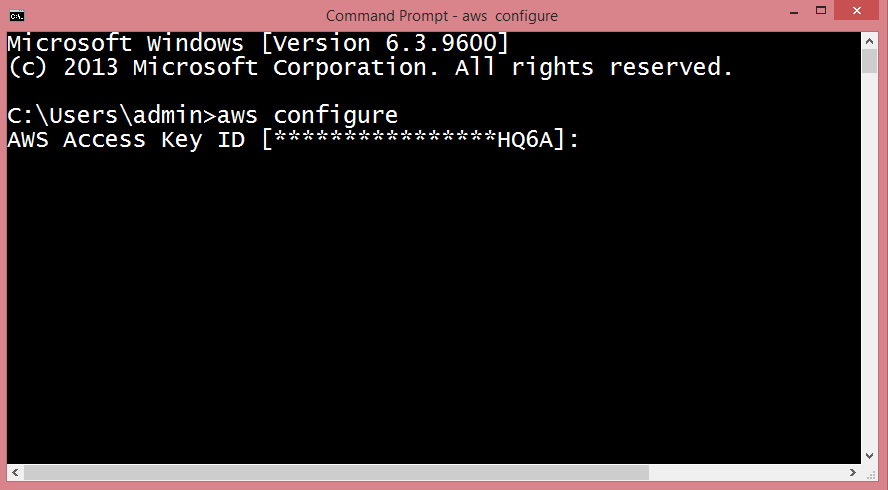
The above screen shows that the Cross region replication has been updated successfully. We can also see the source bucket and destination with their associated permissions.

* Now, we will see whether the files have been replicated from jtpbucket to the jtp1bucket. Click on the **jtp1bucket**.

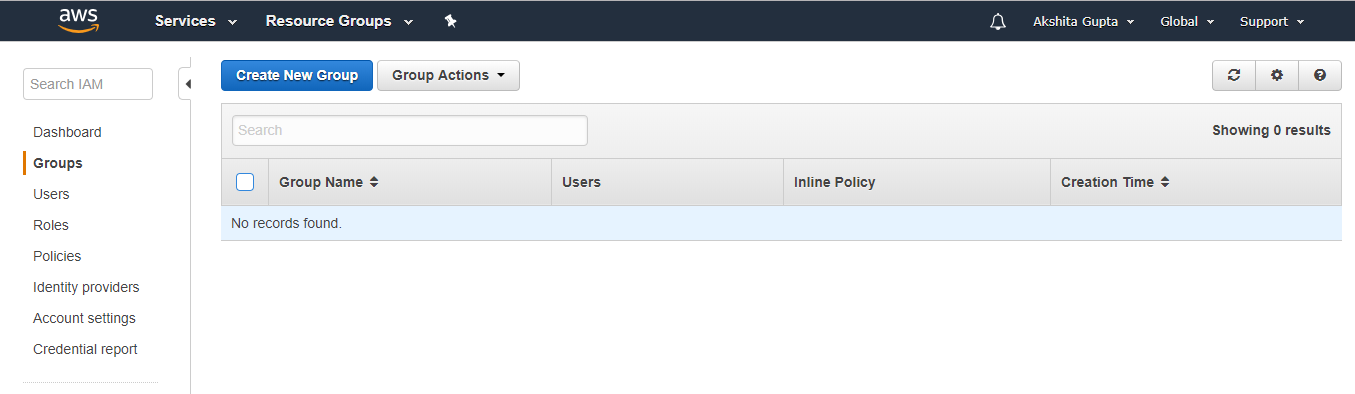


The above screen shows that the bucket is empty. Therefore, we can say that the objects do not replicate from one bucket to another bucket automatically, we can replicate only by using AWS CLI (Command Line Interface). To use the AWS CLI, you need to install the CLI tool.

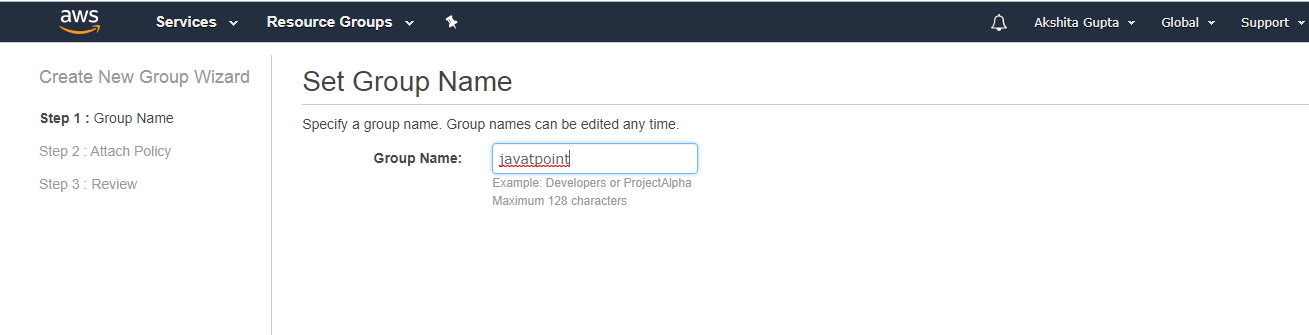
* After installation, open the cmd and type **aws configure**.



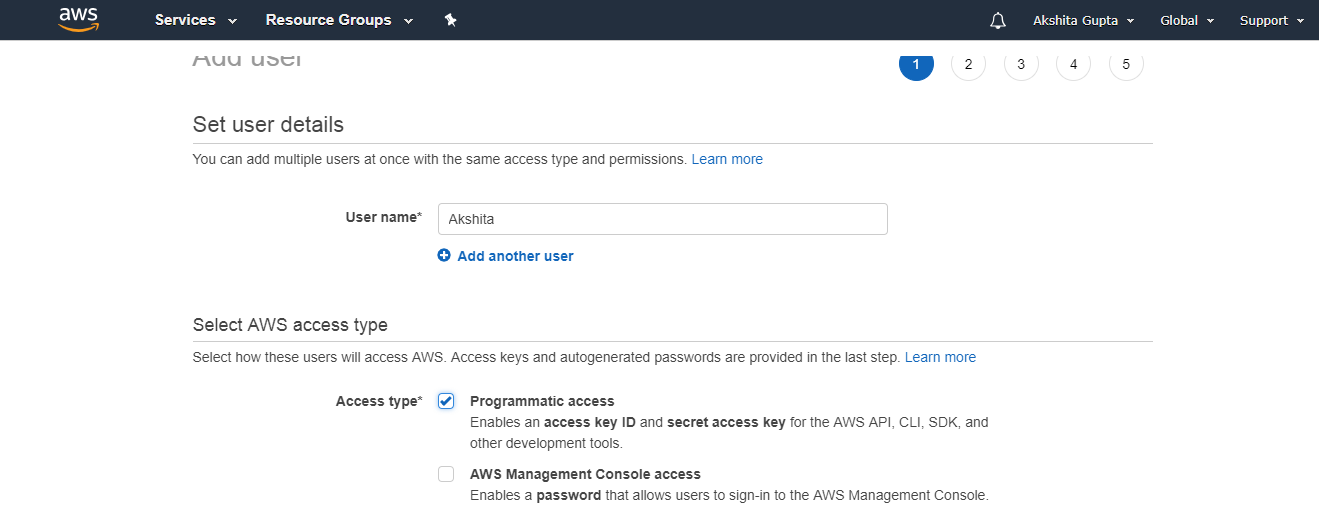
* Now, we need to generate the Access Key ID which is a user name and secret access key which is a password. To achieve this, we first need to create an IAM Group.



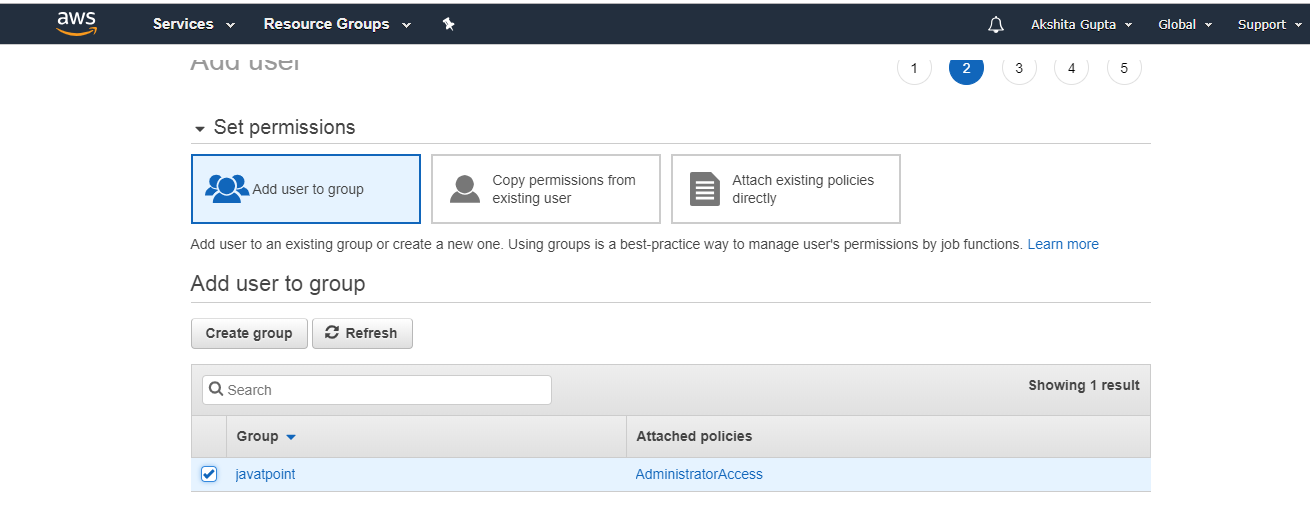
* Set the Group Name, i.e., javatpoint.



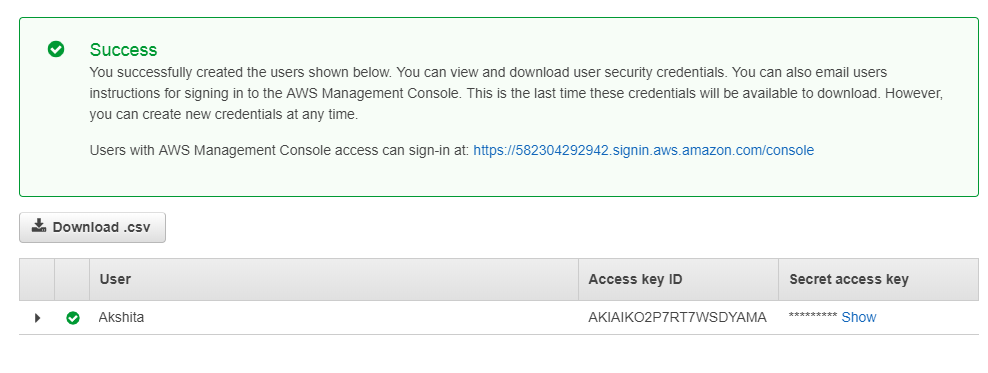
* Check the **AdministratorAccess** policy to access the AWS console through AWS CLI.
* Now, create an IAM User.
* Add the user name with programmatic access.



* Add the user to a group, i.e., javatpoint.

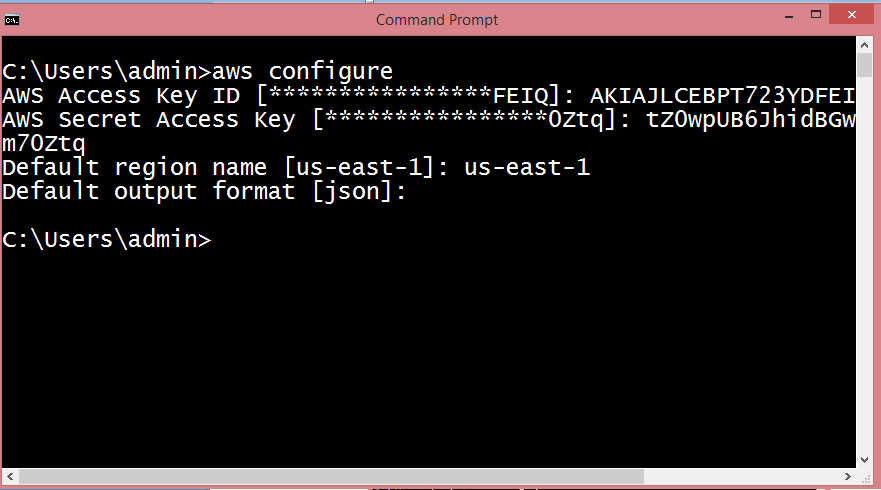


* Finally, the user is created.

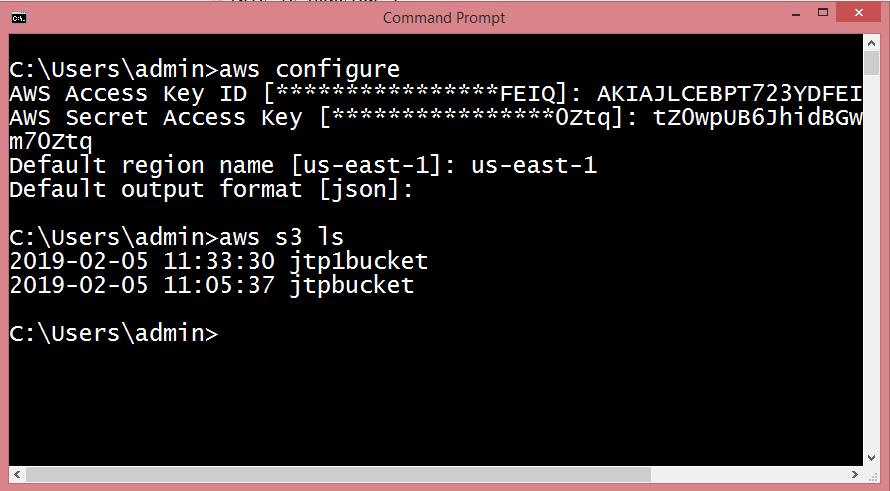


From the above screen, we observe that access key and scret access key have been generated.

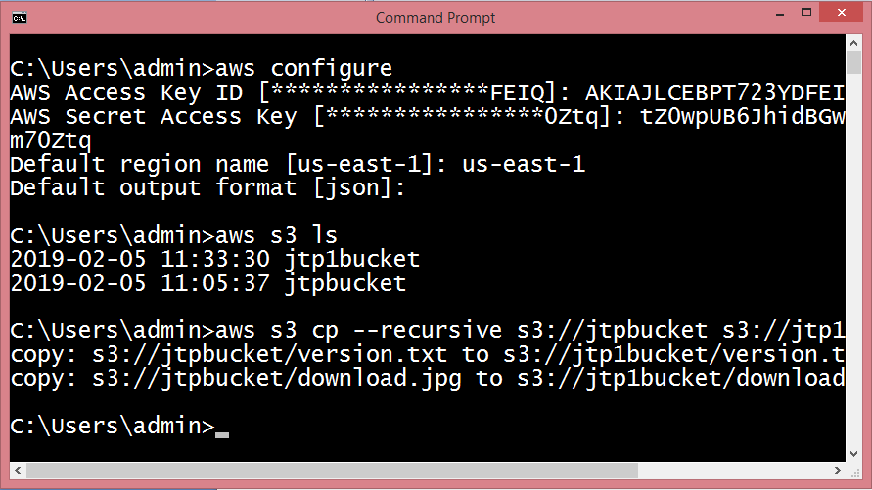
* Copy the access key and secret access key to the cmd.



* To view the S3 buckets, run the command **aws s3 ls**.

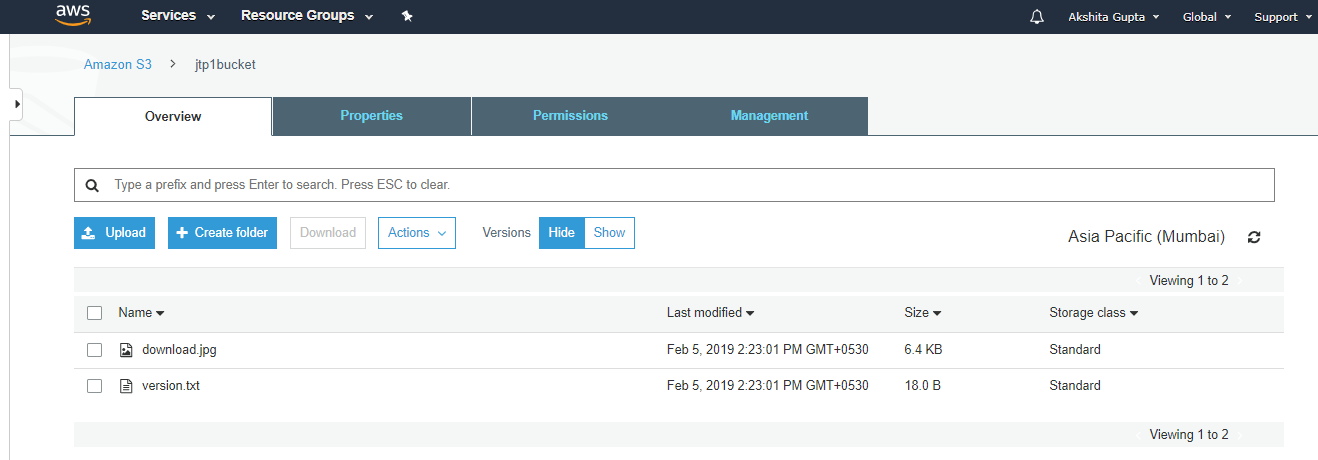


* To copy the objects of **jtpbucket to jtp1bucket, run the command aws s3 cp?recursive s3://jtpbucket s3://jtp1bucket**.



The above screen shows that the objects of **jtpbucket** have been copied to the **jtp1bucket**.

* Click on the **"jtp1bucket"**.



From the above screen, we observed that all the files in the original bucket have been replicated to another bucket, i.e., jtp1bucket.

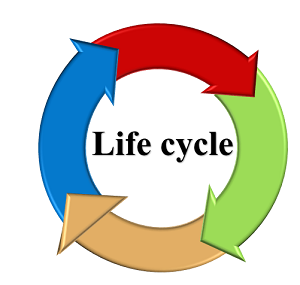
#### **Note: If any further changes made in the original bucket will always be copied to its replicated bucket.**

### Important points to be remembered:

* Versioning must be enabled on both the source and destination buckets.
* The regions of both the buckets must be unique.
* All the files in an original bucket are not replicated automatically, and they can be replicated through AWS CLI. All the subsequent files are replicated automatically.
* Files in a file cannot be replicated to multiple buckets.
* Delete markers are not replicated.
* Delete versions or Delete markers are not replicated.

Lifecycle Management

**Lifecycle Management** is used so that objects are stored cost-effectively throughout their lifecycle. A **lifecycle configuration** is a set of rules that define the actions applied by S3 to a group of objects.



**The lifecycle defines two types of actions:**

* **Transition actions:** When you define the transition to another storage class. For example, you choose to transit the objects to Standard IA storage class 30 days after you have created them or archive the objects to the Glacier storage class 60 days after you have created them.
* **Expiration actions:** You need to define when objects expire, the Amazon S3 deletes the expired object on your behalf.

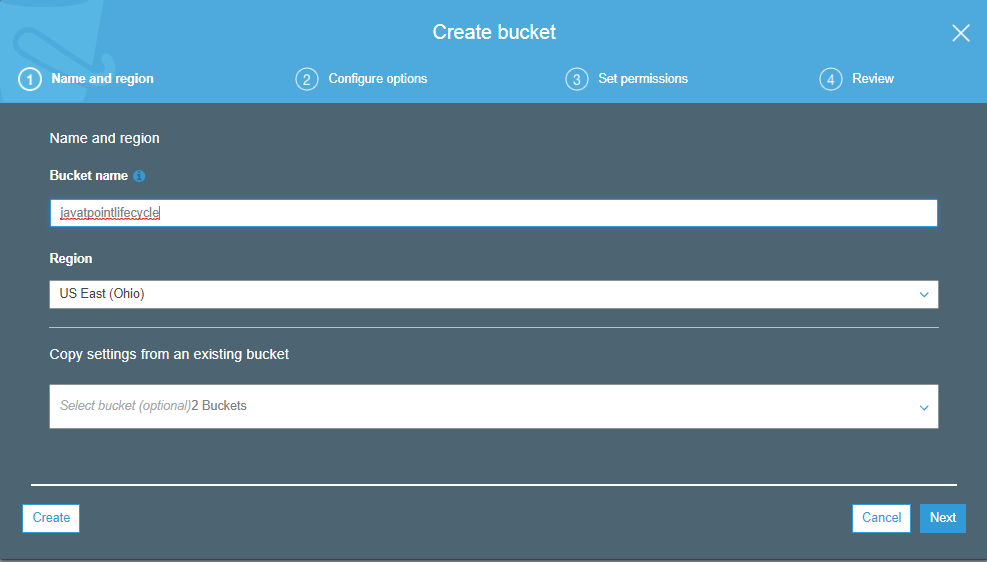
Suppose business generates a lot of data in the form of test files, images, audios or videos and the data is relevant for 30 days only. After that, you might want to transition from standard to standard IA as storage cost is lower. After 60 days, you might want to transit to Glacier storage class for the longtime archival. Perhaps you want to expire the object after 60 days completely, so Amazon has a service known as Lifecycle Management, and this service exist within S3 bucket.

**Lifecycle policies:**

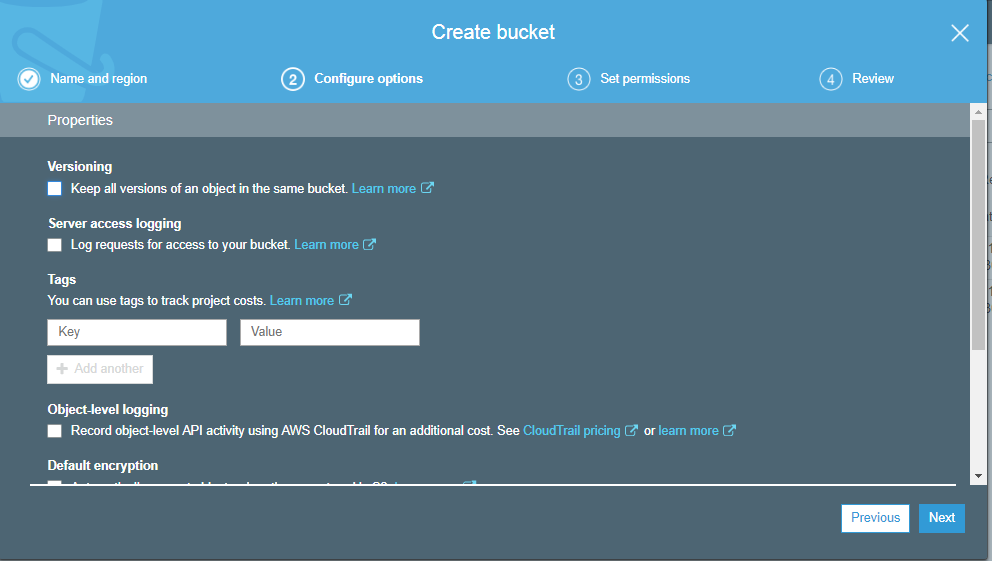
* **Use Lifecycle rules to manage your object:** You can manage the Lifecycle of an object by using a Lifecycle rule that defines how Amazon S3 manages objects during their lifetime.
* **Automate transition to tiered storage:** Lifecycle allows you to transition objects to Standard IA storage class automatically and then to the Glacier storage class.
* **Expire your objects:** Using Lifecycle rule, you can automatically expire your objects.

**Creation of Lifecycle rule**

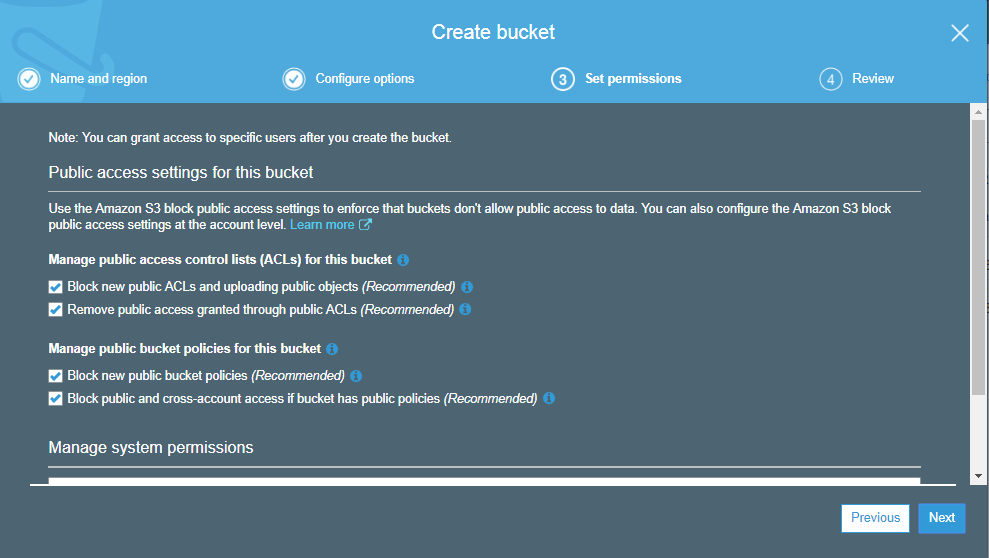
* Sign in to the AWS Management console.
* Click on the **S3** service
* Create a new bucket in S3.
* Enter the bucket name and then click on the **Next** button.



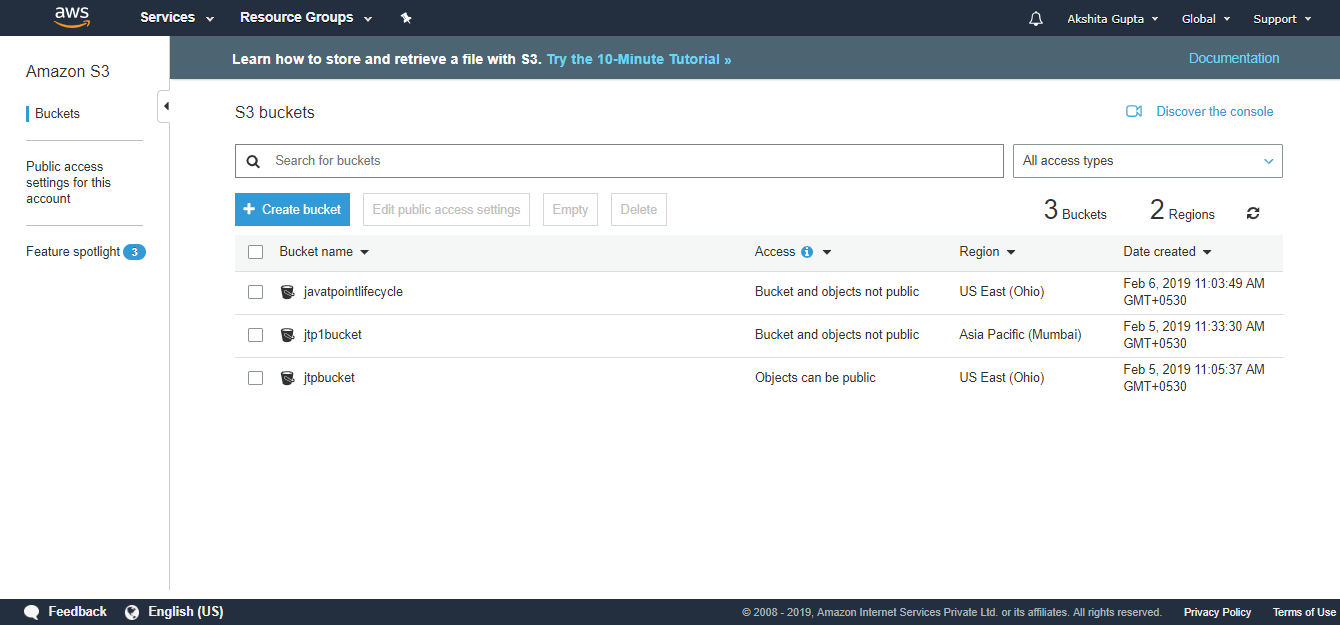
* Now, you can configure the options, i.e., you can set the versioning, server access logging, etc. I leave all the settings as default and then click on the **Next** button.



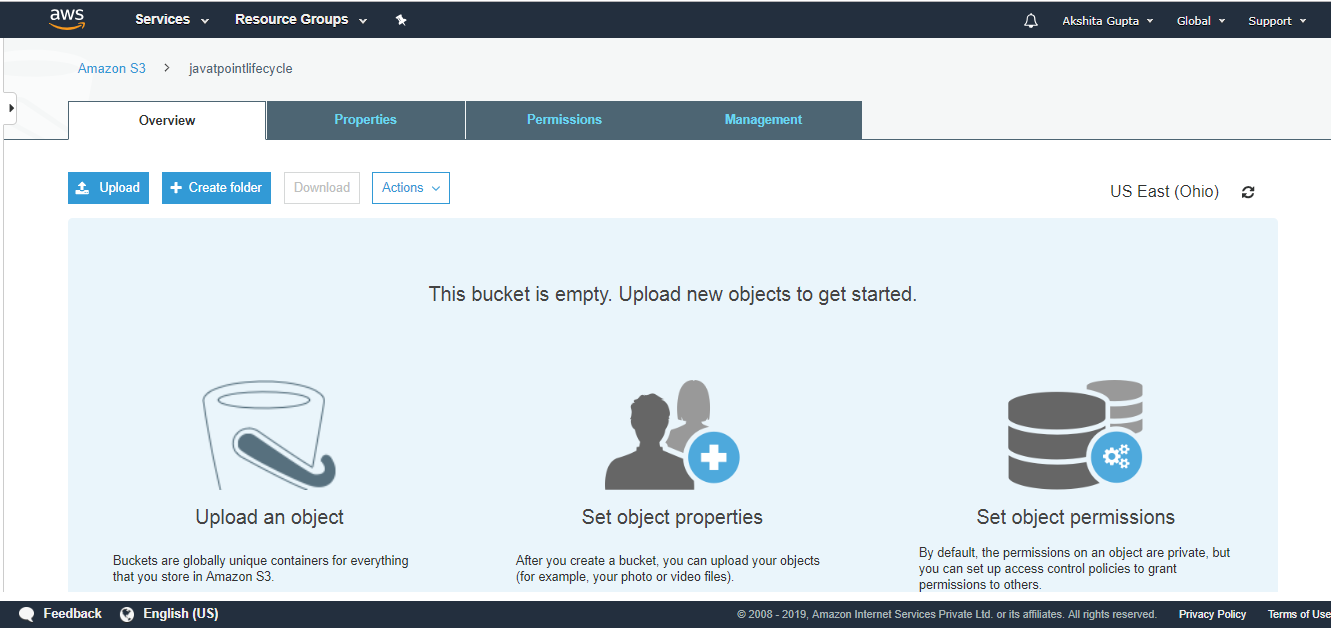
* Set the permissions. I leave all the permissions as default and then click on the **Next** button.



* Click on the **Create bucket** button.
* Finally, the new bucket is created whose name is **"javatpointlifecycle"**.

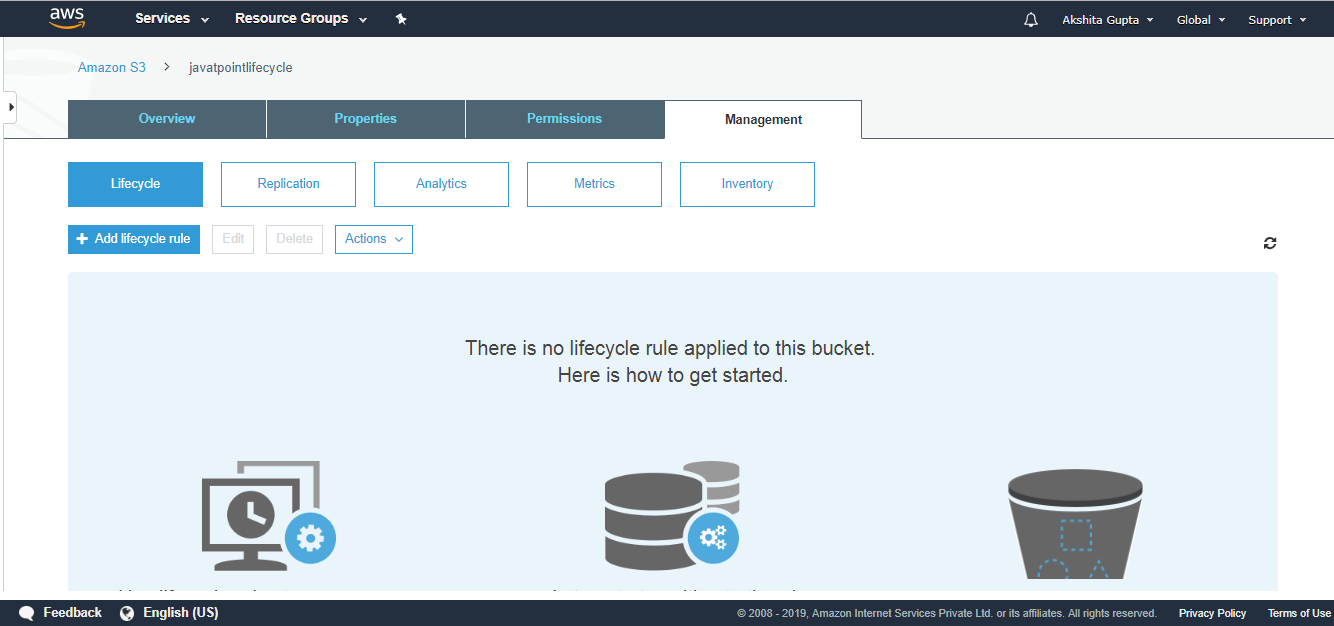


* Click on the **javatpointlifecycle** bucket.

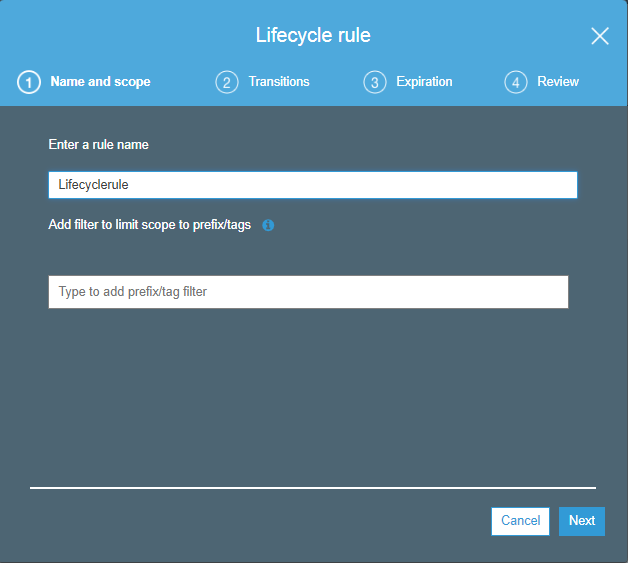


From the above screen, we observe that the bucket is empty. Before uploading the objects in a bucket, we first create the policy.

* Move to the **Management** tab; we use the lifecycle.

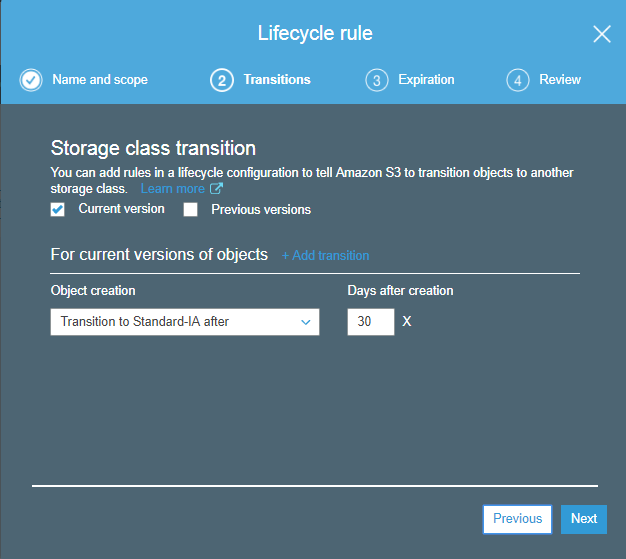


* Add Lifecycle rule and then enter the rule name. Click on the **Next**.

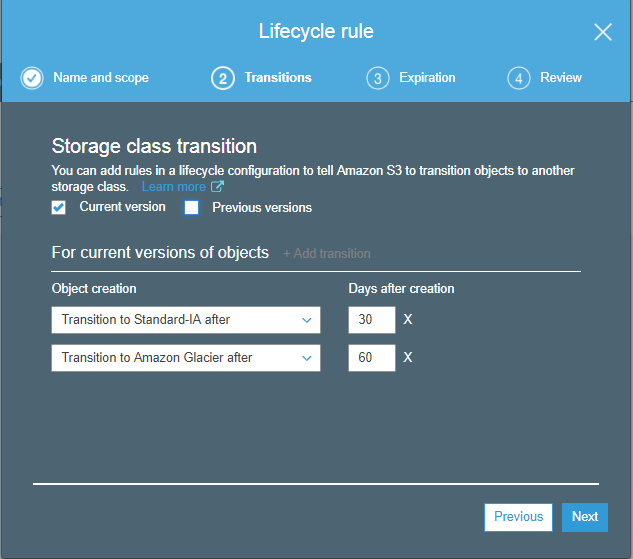


* You can create the storage class transition in both the current version and the previous version. Initially, I create the transition in the current version. Check the **current version** and then click on the **Add transition**.

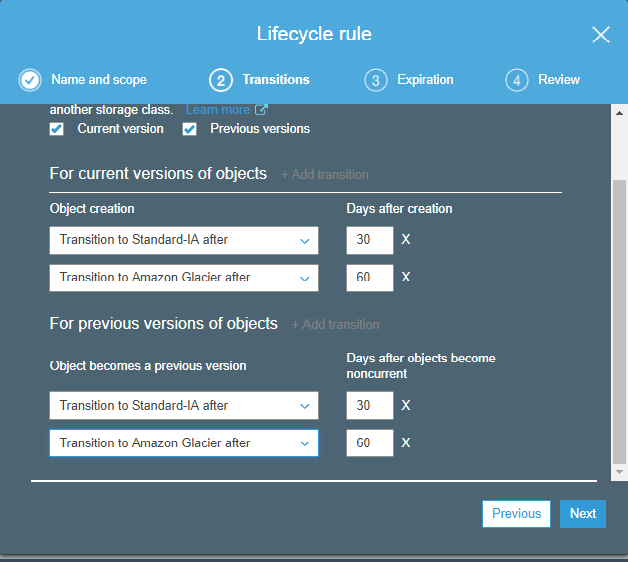
**First transition:** 30 days after the creation of an object, object's storage class is converted to Standard Infrequently access storage class.



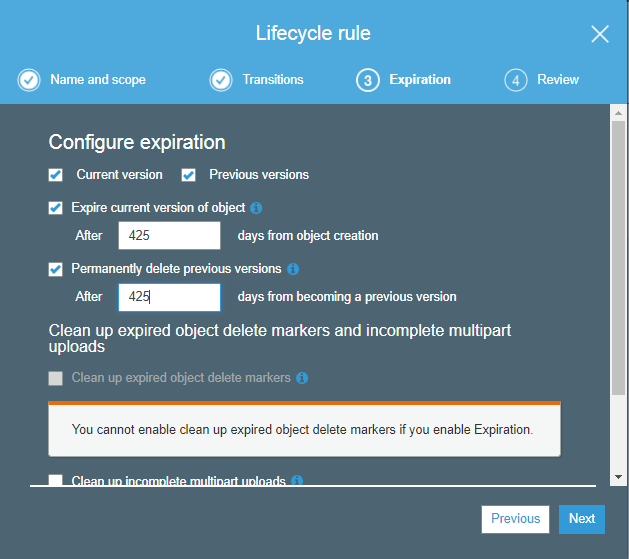
**Second transition:** 60 days after the creation of an object, object's storage class is converted to Glacier storage class.



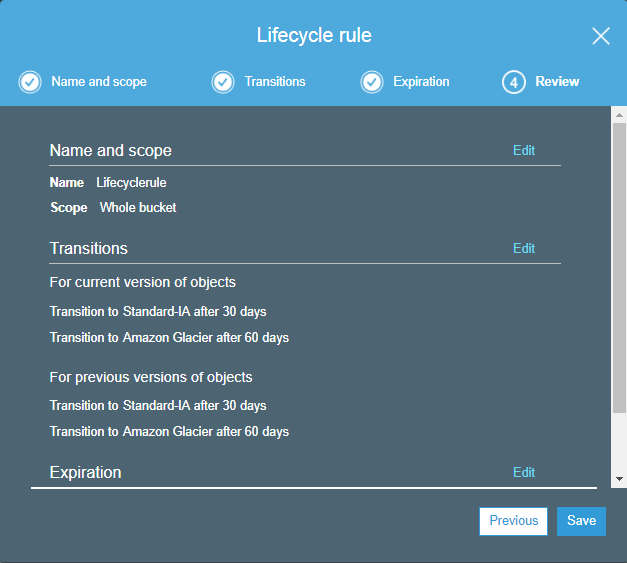
* Similarly, we can do with the previous version objects. Check the **"previous version"** and then **"Add transitions"**. Click on the **Next**.



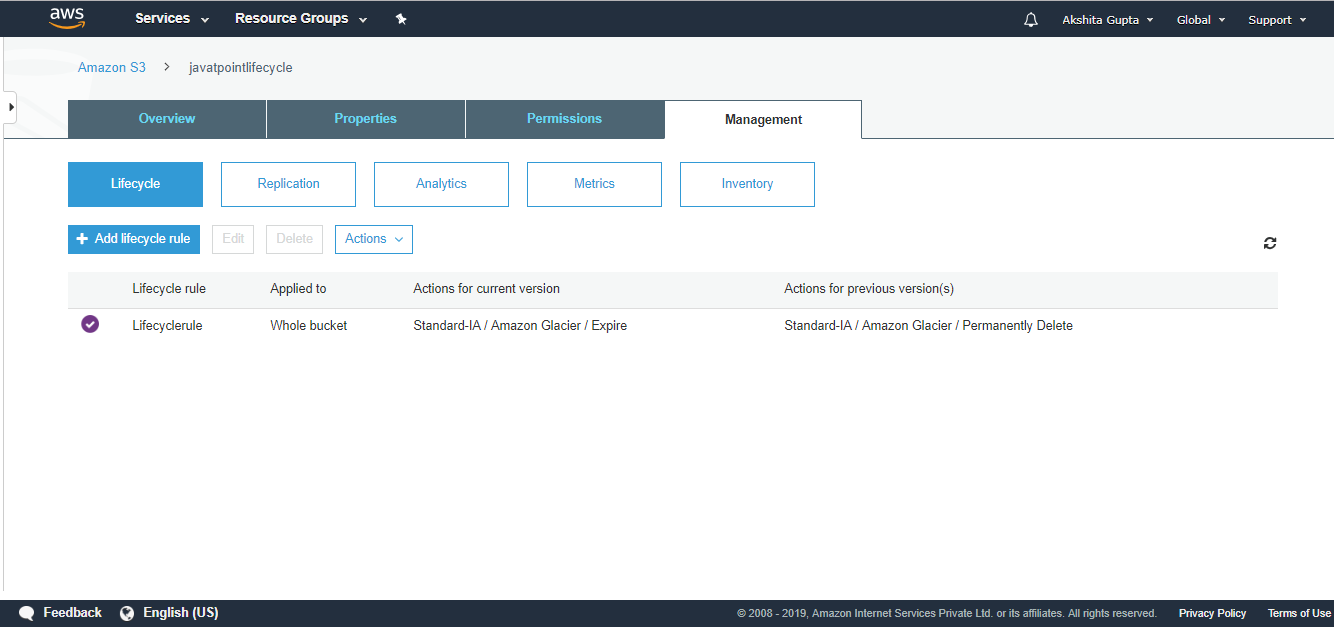
* Now, we expire the object after its creation. Suppose we expire the current and previous version objects after 425 days of its creation. Click on the **Next**.



* The Lifecycle rule is shown given below:



* Click on the **Save**.



The above screen shows that **"Lifecyclerule"** has been created.

Important points to be remembered:

* It can be used either in conjunction with the versioning or without versioning.
* Lifecycle Management can be applied to both current and previous versions.
* The following actions can be done:
  + Transition to Standard Infrequent Access storage class (after 30 days of creation date).
  + Transition to Glacier storage class (after 60 days of creation date).
  + It can also delete the objects permanently.