

Reading 3.3: Object Storage with Amazon S3

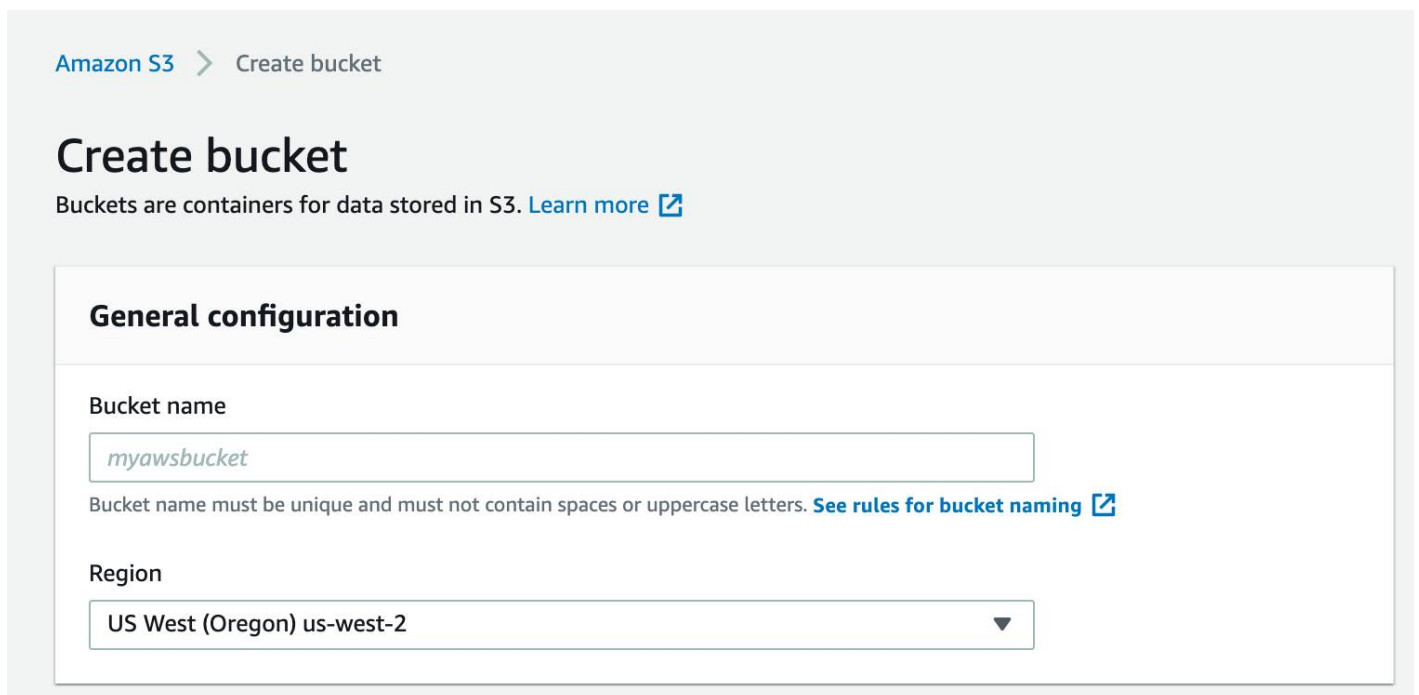
What Is Amazon S3?

Unlike Amazon EBS, Amazon S3 is a standalone storage solution that isn't tied to compute. It enables you to retrieve your data from anywhere on the web. If you've ever used an online storage service to back up the data from your local machine, then you most likely have used a service similar to Amazon S3. The big difference between those online storage services and Amazon S3 is the storage type.

Amazon S3 is an *object storage service*. Object storage stores data in a flat structure, using unique identifiers to look up objects when requested. An object is simply a file combined with metadata and that you can store as many of these objects as you'd like. All of these characteristics of object storage are also characteristics of Amazon S3.

Understand Amazon S3 Concepts

In Amazon S3, you have to store your objects in containers called **buckets**. You can't upload any object, not even a single photo, to S3 without creating a bucket first. When you create a bucket, you choose, at the very minimum, two things: the bucket name and the AWS Region you want the bucket to reside in.



The screenshot shows the 'Create bucket' page in the Amazon S3 console. At the top, there's a breadcrumb 'Amazon S3 > Create bucket'. The main heading is 'Create bucket', followed by a subtext: 'Buckets are containers for data stored in S3. [Learn more](#)'. Below this is a 'General configuration' section. It contains two fields: 'Bucket name' with a text input containing 'myawsbucket', and 'Region' with a dropdown menu showing 'US West (Oregon) us-west-2'. A note below the bucket name field states: 'Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)'.

The first part is choosing the **Region** you want the bucket to reside in. Typically, this will be a Region that you've used for other resources, such as your compute. When you choose a Region for your bucket, all objects you put inside that bucket are redundantly stored across multiple devices, across multiple Availability Zones. This level of redundancy is designed to provide Amazon S3 customers with 99.999999999% durability and 99.99% availability for objects over a given year.

The second part is choosing a **bucket name** which must be unique across all AWS accounts. AWS stops you

from choosing a bucket name that has already been chosen by someone else in another AWS account. Once you choose a name, that name is yours and cannot be claimed by anyone else unless you delete that bucket, which then releases the name for others to use.

`http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.html`



The diagram shows the URL `http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.html`. An orange arrow points from the word "Bucket" to the domain part `doc.s3.amazonaws.com`. A green arrow points from the text "Object/Key" to the path part `/2006-03-01/AmazonS3.html`.

AWS uses this name as part of the object identifier. In S3, each object is identified using a URL, which looks like this:

After the `http://`, you see the bucket name. In this example, the bucket is named `doc`. Then, the identifier uses the service name, `s3` and specifies the service provider `amazonaws`. After that, you have an implied folder inside the bucket called `2006-03-01` and the object inside the folder that is named `AmazonS3.html`. The object name is often referred to as *the key name*.

Note, you can have folders inside of buckets to help you organize objects. However, remember that there's no actual file hierarchy that supports this on the back end. It is instead a flat structure where all files and folders live at the same level. Using buckets and folders implies a hierarchy, which makes it easy to understand for the human eye.

S3 Use Cases

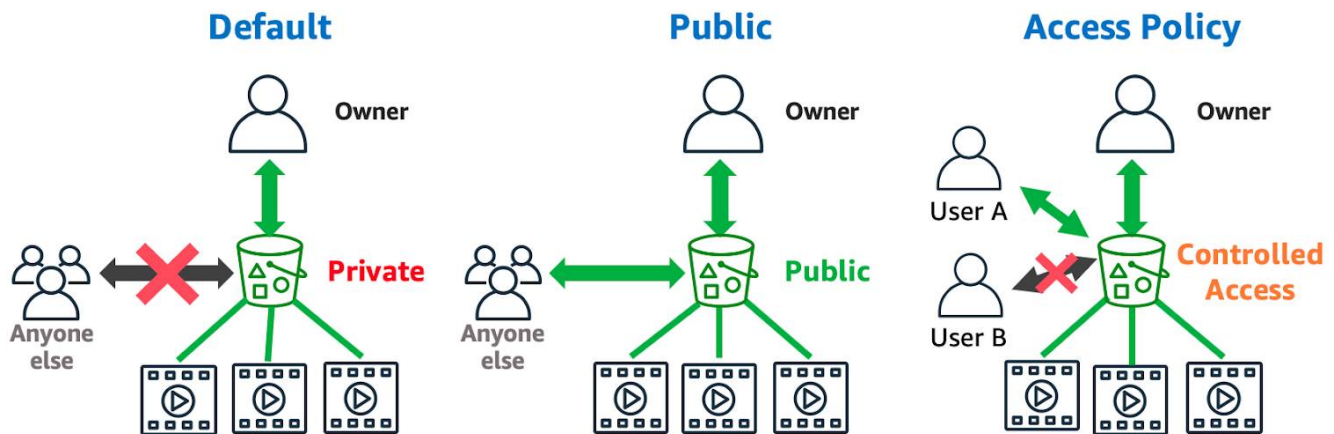
Amazon S3 is one of the most widely used storage services, with far more use cases than could fit on one screen. The following list summarizes some of the most common ways you can use Amazon S3.

- **Backup and storage:** S3 is a natural place to back up files because it is highly redundant. As mentioned in the last unit, AWS stores your EBS snapshots in S3 to take advantage of its high availability.
- **Media hosting:** Because you can store unlimited objects, and each individual object can be up to 5 TBs, S3 is an ideal location to host video, photo, or music uploads.
- **Software delivery:** You can use S3 to host your software applications that customers can download.
- **Data lakes:** S3 is an optimal foundation for a data lake because of its virtually unlimited scalability. You can increase storage from gigabytes to petabytes of content, paying only for what you use.
- **Static websites:** You can configure your bucket to host a static website of HTML, CSS, and client-side scripts.
- **Static content:** Because of the limitless scaling, the support for large files, and the fact that you access any object over the web at any time, S3 is the perfect place to store static content.

Choose the Right Connectivity Option for Your Resources

Everything in Amazon S3 is private by default. This means that all S3 resources, such as buckets, folders, and objects can only be viewed by the user or AWS account that created that resource. Amazon S3 resources are all private and protected to begin with.

If you decide that you want everyone on the internet to see your photos, you can choose to make your buckets, folders, and objects public. Keep in mind that a public resource means that everyone on the internet can see it. Most of the time, you don't want your permissions to be all or nothing. Typically, you want to be more granular about the way you provide access to your resources.



To be more specific about who can do what with your S3 resources, Amazon S3 provides two main access management features: IAM policies and S3 bucket policies.

Understand IAM Policies

Previously, you learned about creating and using IAM policies, and now you get to apply this to Amazon S3. When IAM policies are attached to IAM users, groups, and roles, the policies define which actions they can perform. IAM policies are not tied to any one AWS service and can be used to define access to nearly any AWS action.

You should use IAM policies for private buckets when:

- You have many buckets with different permission requirements. Instead of defining many different S3 bucket policies, you can use IAM policies instead.
- You want all policies to be in a centralized location. Using IAM policies allows you to manage all policy information in one location.

Understand S3 Bucket Policies

S3 bucket policies are similar to IAM policies, in that they are both defined using the same policy language in a JSON format. The difference is IAM policies are attached to users, groups, and roles, whereas S3 bucket policies are only attached to buckets. S3 bucket policies specify what actions are allowed or denied on the bucket.

For example, if you have a bucket called `employeebucket`, you can attach an S3 bucket policy to it that allows another AWS account to put objects in that bucket.

Or if you wanted to allow anonymous viewers to read the objects in `employeebucket`, then you can apply a policy to that bucket that allows anyone to read objects in the bucket using `"Effect": Allow` on the `"Action":`

```
["s3:GetObject"]".
```

Here's an example of what that S3 bucket policy might look like.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "PublicRead",
    "Effect": "Allow",
    "Principal": "*",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3:::employeebucket/*"]
  }]
}
```

S3 Bucket policies can only be placed on buckets, and cannot be used for folders or objects. However, the policy that is placed on the bucket applies to every object in that bucket.

You should use S3 bucket policies when:

- You need a simple way to do cross-account access to S3, without using IAM roles.
- Your IAM policies bump up against the defined size limit. S3 bucket policies have a larger size limit.

Encrypt S3

Amazon S3 reinforces encryption in transit (as it travels to and from Amazon S3) and at rest. To protect data at rest, you can use:

- **Server-side encryption:** This allows Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.
- **Client-side encryption:** Encrypt your data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and all related tools.

To encrypt in transit, you can use client-side encryption or Secure Sockets Layer (SSL).

Use Versioning to Preserve Objects

As you know, Amazon S3 identifies objects in part by using the object name. For example, when you upload an employee photo to S3, you may name the object `employee.jpg` and store it in a folder called `employees`. If you don't use Amazon S3 versioning, anytime you upload an object called `employee.jpg` to the `employees` folder, it overwrites the original file.

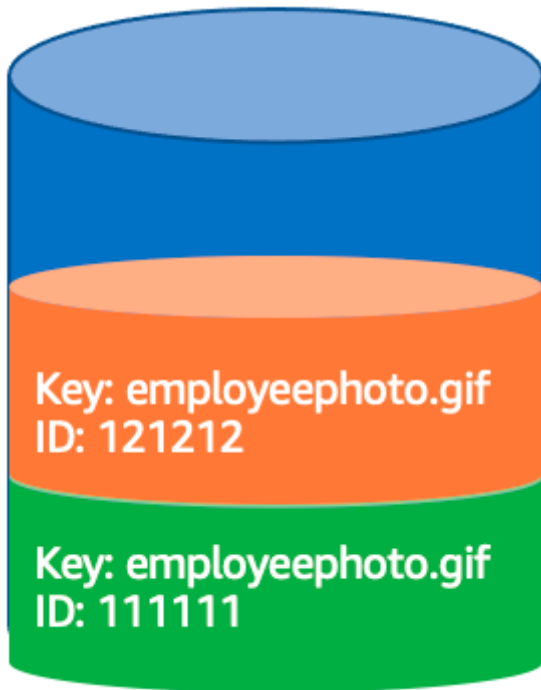
This can be an issue for several reasons.

- `employee.jpg` is a common name for an employee photo object. You or someone else who has access to that bucket might not have intended to overwrite it, and now that you have, you no longer have access to the original file.
- You may want to preserve different versions of `employee.jpg`. Without versioning, if you wanted to create a new version of `employee.jpg`, you would need to upload the object and choose a different name for it.

Having several objects all with slight differences in naming variations may cause confusion and clutter in your bucket.

So, what do you do? You use S3 versioning!

Versioning enables you to keep multiple versions of a single object in the same bucket. This allows you to preserve old versions of an object without having to use different naming constructs, in case you need to recover from accidental deletions, accidental overwrites, or even application failures. Let's see how this works.



If you enable versioning for a bucket, Amazon S3 automatically generates a unique version ID for the object being stored. In one bucket, for example, you can have two objects with the same key, but different version IDs, such as employeephoto.gif (version 111111) and employeephoto.gif (version 121212).

Versioning-enabled buckets let you recover objects from accidental deletion or overwrite.

- Deleting an object does not remove the object permanently. Instead, Amazon S3 puts a marker on the object that shows you tried to delete it. If you want to restore the object, you can remove this marker and it reinstates the object.
- If you overwrite an object, it results in a new object version in the bucket. You still have access to previous versions of the object.

Understand Versioning States

Buckets can be in one of three states.

- Unversioned (the default): No new or existing objects in the bucket have a version.
- Versioning-enabled: This enables versioning for all objects in the bucket.
- Versioning-suspended: This suspends versioning for new objects. All new objects in the bucket will not have a version. However, all existing objects keep their object versions.

The versioning state applies to all of the objects in that bucket. Keep in mind that storage costs are incurred for all objects in your bucket and all versions of those objects. To reduce your S3 bill, you may want to delete previous versions of your objects that are no longer in use.

What Are Amazon S3 Storage Classes?

When you upload an object to Amazon S3 and you don't specify the storage class, you're uploading it to the default storage class—often referred to as standard storage. When you learned about Amazon S3 in previous units, you were learning about the standard storage class without even knowing it!

S3 storage classes let you change your storage tier as your data characteristics change. For example, if you are now accessing your old photos infrequently, you may want to change the storage class those photos are stored in to save on costs.

There are six S3 storage classes.

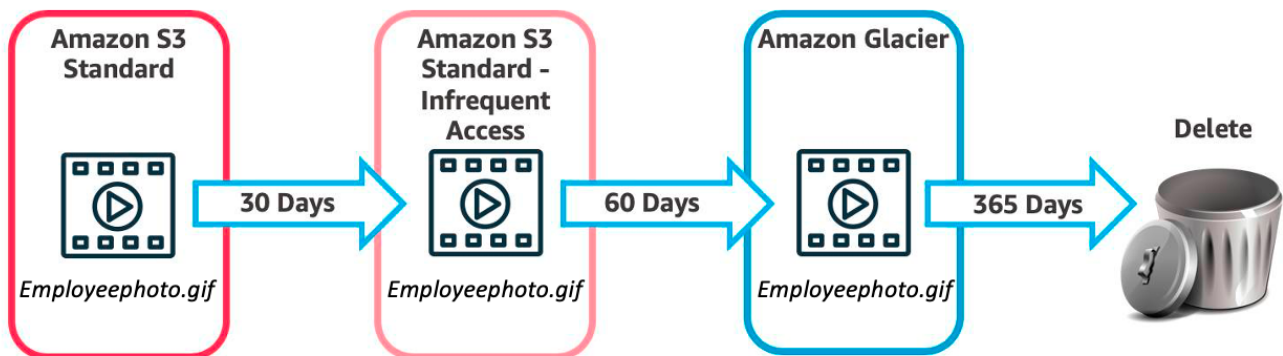
1. **Amazon S3 Standard:** This is considered general purpose storage for cloud applications, dynamic websites, content distribution, mobile and gaming applications, and big data analytics.
2. **Amazon S3 Intelligent-Tiering:** This tier is useful if your data has unknown or changing access patterns. S3 Intelligent-Tiering stores objects in two tiers, a frequent access tier and an infrequent access tier. Amazon S3 monitors access patterns of your data, and automatically moves your data to the most cost-effective storage tier based on frequency of access.
3. **Amazon S3 Standard-Infrequent Access (S3 Standard-IA):** S3 Standard-IA is for data that is accessed less frequently, but requires rapid access when needed. S3 Standard-IA offers the high durability, high throughput, and low latency of S3 Standard, with a low per-GB storage price and per-GB retrieval fee. This storage tier is ideal if you want to store long-term backups, disaster recovery files, and so on.
4. **Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA):** Unlike other S3 storage classes which store data in a minimum of three Availability Zones (AZs), S3 One Zone-IA stores data in a single AZ and costs 20% less than S3 Standard-IA. S3 One Zone-IA is ideal for customers who want a lower-cost option for infrequently accessed data but do not require the availability and resilience of S3 Standard or S3 Standard-IA. It's a good choice for storing secondary backup copies of on-premises data or easily re-creatable data.
5. **Amazon S3 Glacier:** S3 Glacier is a secure, durable, and low-cost storage class for data archiving. You can reliably store any amount of data at costs that are competitive with or cheaper than on-premises solutions. To keep costs low yet suitable for varying needs, S3 Glacier provides three retrieval options that range from a few minutes to hours.
6. **Amazon S3 Glacier Deep Archive:** S3 Glacier Deep Archive is Amazon S3's lowest-cost storage class and supports long-term retention and digital preservation for data that may be accessed once or twice in a year. It is designed for customers—particularly those in highly regulated industries, such as the Financial Services, Healthcare, and Public Sectors—that retain data sets for 7 to 10 years or longer to meet regulatory compliance requirements.

Automate Tier Transitions with Object Lifecycle Management

If you keep manually changing your objects, such as your employee photos, from storage tier to storage tier, you may want to look into automating this process using a lifecycle policy. When you define a lifecycle policy configuration for an object or group of objects, you can choose to automate two actions: transition and expiration actions.

- **Transition actions** are used to define when you should transition your objects to another storage class.
- **Expiration actions** define when objects expire and should be permanently deleted.

For example, you might choose to transition objects to S3 Standard-IA storage class 30 days after you created them, or archive objects to the S3 Glacier storage class one year after creating them.



The following use cases are good candidates for lifecycle management.

- **Periodic logs:** If you upload periodic logs to a bucket, your application might need them for a week or a month. After that, you might want to delete them.
- **Data that changes in access frequency:** Some documents are frequently accessed for a limited period of time. After that, they are infrequently accessed. At some point, you might not need real-time access to them, but your organization or regulations might require you to archive them for a specific period. After that, you can delete them.