

**Introduction to Amazon S3**

This introduction to Amazon Simple Storage Service is intended to give you a detailed summary of this web service

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.

## Advantages to Amazon S3

Amazon S3 is intentionally built with a minimal feature set that focuses on simplicity and robustness. Following are some of advantages of the Amazon S3 service:

* Create Buckets – Create and name a bucket that stores data. Buckets are the fundamental container in Amazon S3 for data storage.
* Store data in Buckets – Store an infinite amount of data in a bucket. Upload as many objects as you like into an Amazon S3 bucket. Each object can contain up to 5 TB of data. Each object is stored and retrieved using a unique developer-assigned key.
* Download data – Download your data or enable others to do so. Download your data any time you like or allow others to do the same.
* Permissions – Grant or deny access to others who want to upload or download data into your Amazon S3 bucket. Grant upload and download permissions to three types of users. Authentication mechanisms can help keep data secure from unauthorized access.
* Standard interfaces – Use standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

Note

SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

## Amazon S3 Concepts

* Buckets
* Objects
* Keys
* Regions
* Amazon S3 Data Consistency Model

### Buckets

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

Buckets serve several purposes: they organize the Amazon S3 namespace at the highest level, they identify the account responsible for storage and data transfer charges, they play a role in access control, and they serve as the unit of aggregation for usage reporting.

You can configure buckets so that they are created in a specific region

### Objects

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3. 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. For more information, see Keys and Versioning.

### Keys

A key is the unique identifier for an object within a bucket. 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. You might choose a region to optimize latency, minimize costs, or address regulatory requirements. Amazon S3 currently supports the following regions:

* **US East (N. Virginia) Region** Uses Amazon S3 servers in Northern Virginia
* **US East (Ohio) Region** Uses Amazon S3 servers in Columbus Ohio
* **US West (N. California) Region** Uses Amazon S3 servers in Northern California
* **US West (Oregon) Region** Uses Amazon S3 servers in Oregon
* **Canada (Central) Region** Uses Amazon S3 servers in Montreal
* **Asia Pacific (Mumbai) Region** Uses Amazon S3 servers in Mumbai
* **Asia Pacific (Seoul) Region** Uses Amazon S3 servers in Seoul
* **Asia Pacific (Singapore) Region** Uses Amazon S3 servers in Singapore
* **Asia Pacific (Sydney) Region** Uses Amazon S3 servers in Sydney
* **Asia Pacific (Tokyo) Region** Uses Amazon S3 servers in Tokyo
* **EU (Frankfurt) Region** Uses Amazon S3 servers in Frankfurt
* **EU (Ireland) Region** Uses Amazon S3 servers in Ireland
* **EU (London) Region** Uses Amazon S3 servers in London
* **South America (São Paulo) Region** Uses Amazon S3 servers in Sao Paulo

Objects stored in a region never leave the region unless you explicitly transfer them to another region. For example, objects stored in the EU (Ireland) region never leave it. For more information about Amazon S3 regions and endpoints, go to Regions and Endpoints in the AWS General Reference.

### Amazon S3 Data Consistency Model

Amazon S3 provides read-after-write consistency for PUTS of new objects in your S3 bucket in all regions with one caveat. The caveat is that if you make a HEAD or GET request to the key name (to find if the object exists) before creating the object, Amazon S3 provides eventual consistency for read-after-write.

Amazon S3 offers eventual consistency for overwrite PUTS and DELETES in all regions.

Updates to a single key are atomic. For example, if you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never write corrupted or partial data.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However, information about the changes must replicate across Amazon S3, which can take some time, and so you might observe the following behaviors:

* A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
* A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
* A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.
* A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.

Note

Amazon S3 does not currently support object locking. If two PUT requests are simultaneously made to the same key, the request with the latest time stamp wins. If this is an issue, you will need to build an object-locking mechanism into your application.

Updates are key-based; there is no way to make atomic updates across keys. For example, you cannot make the update of one key dependent on the update of another key unless you design this functionality into your application.

The following table describes the characteristics of eventually consistent read and consistent read.

|  |  |
| --- | --- |
| **Eventually Consistent Read** | **Consistent Read** |
| Stale reads possible | No stale reads |
| Lowest read latency | Potential higher read latency |
| Highest read throughput | Potential lower read throughput |