Upload data

After getting your storage organized into buckets, it's time to add some objects by uploading data.

Depending on how you want to use your storage, there are different ways to get data into the system. A data scientist has a few large files that are used for analytics, a systems administrator needs to keep database backups synchronized with local files, and a developer is writing software that needs to read and write millions of files. Each of these scenarios is best served by different methods of data ingest.

Some applications may wish to restrict a user or Service ID to only uploading data, without any access to reading data in a bucket. This is possible through the Object Writer [IAM role](https://cloud.ibm.com/docs/cloud-object-storage?topic=cloud-object-storage-iam).

**Using the console**

Typically, using the web-based console is not the most common way to use Object Storage. Objects are limited to 200 MB and the file name and key are identical. Multiple objects can be uploaded at the same time, and if the browser allows for multiple threads each object will be uploaded by using multiple parts in parallel. Support for larger object sizes and improved performance (depending on network factors) is provided by [Aspera high-speed transfer](https://cloud.ibm.com/docs/cloud-object-storage/basics?topic=cloud-object-storage-aspera).

**Using a compatible tool**

Some users want to use a stand-alone utility to interact with their storage. As the Cloud Object Storage API supports the most common set of S3 API operations, many S3-compatible tools can also connect to Object Storage by using [HMAC credentials](https://cloud.ibm.com/docs/cloud-object-storage?topic=cloud-object-storage-uhc-hmac-credentials-main).

Some examples include file explorers like [Cyberduck](https://cyberduck.io/" \t "_blank) or [Transmit](https://panic.com/transmit/), backup utilities like [Cloudberry](https://www.cloudberrylab.com/) and [Duplicati](https://www.duplicati.com/" \t "_blank), command-line utilities like [s3cmd](https://github.com/s3tools/s3cmd" \t "_blank) or [Minio Client](https://github.com/minio/mc" \t "_blank), and many others.

**Using the API**

Most programmatic applications of Object Storage use an SDK (such as [Java](https://cloud.ibm.com/docs/services/cloud-object-storage/libraries?topic=cloud-object-storage-java), [node.js](https://cloud.ibm.com/docs/services/cloud-object-storage/libraries?topic=cloud-object-storage-node), or [Python](https://cloud.ibm.com/docs/services/cloud-object-storage/libraries?topic=cloud-object-storage-python)) or the [Cloud Object Storage API](https://cloud.ibm.com/docs/services/cloud-object-storage/api-reference?topic=cloud-object-storage-compatibility-api). Typically objects are uploaded in [multiple parts](https://cloud.ibm.com/docs/services/cloud-object-storage/basics?topic=cloud-object-storage-large-objects), with part size and number of parts configured by a Transfer Manager class.

**Conditional requests**

When making a request to read or write data, it is possible to set conditions on that request to avoid unnecessary operations. This is accomplished using the following pre-conditional HTTP headers: If-Match, If-None-Match, If-Modified-Since, and If-Unmodified-Since.

It is generally preferable to use If-Match because the granularity of the Last-Modified value is only in seconds, and may not be sufficient to avoid race conditions in some applications.

Using If-Match

On an object PUT, HEAD, or GET request, [the If-Match header](https://tools.ietf.org/html/rfc7232#section-3.1) will check to see if a provided Etag (MD5 hash of the object content) matches the provided Etag value. If this value matches, the operation will proceed. If the match fails, the system will return a 412 Precondition Failed error.

If-Match is most often used with state-changing methods (e.g., POST, PUT, DELETE) to prevent accidental overwrites when multiple user agents might be acting in parallel on the same resource (i.e., to prevent the "lost update" problem).

Using If-None-Match

On an object PUT, HEAD, or GET request, [the If-None-Match header](https://tools.ietf.org/html/rfc7232#section-3.2) will check to see if a provided Etag (MD5 hash of the object content) matches the provided Etag value. If this value does not match, the operation will proceed. If the match succeeds, the system will return a 412 Precondition Failed error on a PUT and a 304 Not Modified on GET or HEAD.

If-None-Match is primarily used in conditional GET requests to enable efficient updates of cached information with a minimum amount of transaction overhead. When a client desires to update one or more stored responses that have entity-tags, the client SHOULD generate an If-None-Match header field containing a list of those entity-tags when making a GET request; this allows recipient servers to send a 304 (Not Modified) response to indicate when one of those stored responses matches the selected representation.

Using If-Modified-Since

On an object HEAD or GET request, [the If-Modified-Since header](https://tools.ietf.org/html/rfc7232#section-3.3) will check to see if the object's Last-Modified value (for example Sat, 14 March 2020 19:43:31 GMT) is newer than a provided value. If the object has been modified, the operation will proceed. If the object has not been modified, the system will return a 304 Not Modified.

If-Modified-Since is typically used for two distinct purposes: 1) to allow efficient updates of a cached representation that does not have an entity-tag and 2) to limit the scope of a web traversal to resources that have recently changed.

Using If-Unmodified-Since

On an object PUT, HEAD, or GET request, [the If-Unmodified-Since header](https://tools.ietf.org/html/rfc7232#section-3.3) will check to see if the object's Last-Modified value (for example Sat, 14 March 2020 19:43:31 GMT) is equal to or earlier than a provided value. If the object has not been modified, the operation will proceed. If the Last-Modified value is more recent, the system will return a 412 Precondition Failed error on a PUT and a 304 Not Modified on GET or HEAD.

If-Unmodified-Since is most often used with state-changing methods (e.g., POST, PUT, DELETE) to prevent accidental overwrites when multiple user agents might be acting in parallel on a resource that does not supply entity-tags with its representations (i.e., to prevent the "lost update" problem). It can also be used with safe methods to abort a request if the selected representation does not match one already stored (or partially stored) from a prior request.