Cloud Storage Providers

Amazon Simple Storage Service (S3)

- The best-known cloud storage service is Amazon's Simple Storage Service (S3), launched in 2006.
- Amazon S3 is designed to make computing easier for developers.
- Amazon S3 provides an interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web.
- Amazon S3 is intentionally built with a minimal feature set that includes the following functionality:

  • Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each.

  The number of objects that can be stored is unlimited.

  • Each object is stored and retrieved via a unique developer-assigned key.

  • Objects can be made private or public, and rights can be assigned to specific users.

  • Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

Design Requirements

Amazon built S3 to fulfill the following design requirements:

• **Scalable** Amazon S3 can scale in terms of storage, request rate, and users to support an unlimited number of web-scale applications.

- **Reliable** Store data durably, with 99.99 percent availability. Amazon says it does not allow any downtime.

• **Fast** Amazon S3 was designed to be fast enough to support high-performance applications. Server-side latency must be insignificant relative to Internet latency. Any performance bottlenecks can be fixed by simply adding nodes to the system.

• **Inexpensive** Amazon S3 is built from inexpensive commodity hardware components. As a result, frequent node failure is the norm and must not affect the overall system. It must be hardware-agnostic, so that savings can be captured as Amazon continues to drive down infrastructure costs.

• **Simple** Building highly scalable, reliable, fast, and inexpensive storage is difficult. Doing so in a way that makes it easy to use for any application anywhere is more difficult. Amazon S3 must do both.

Design Principles

Amazon used the following principles of distributed system design to meet Amazon S3 requirements:

• **Decentralization** It uses fully decentralized techniques to remove scaling bottlenecks and single points of failure.

• **Autonomy** The system is designed such that individual components can make decisions based on local information.

• **Local responsibility** Each individual component is responsible for achieving its consistency; this is never the burden of its peers.

• **Controlled concurrency** Operations are designed such that no or limited concurrency control is required.

• **Failure toleration** The system considers the failure of components to be a normal mode of operation and continues operation with no or minimal interruption.

• **Controlled parallelism** Abstractions used in the system are of such granularity that parallelism can be used to improve performance and robustness of recovery or the introduction of new nodes.

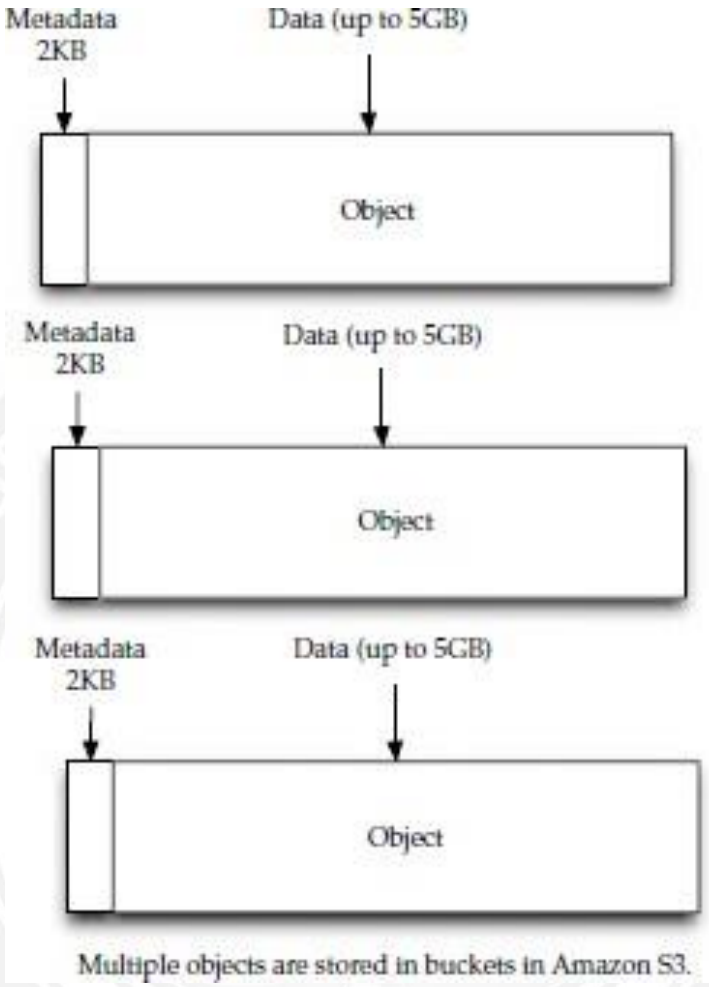
• **Small, well-understood building blocks** Do not try to provide a single service that does everything for everyone, but instead build small components that can be used as building blocks for other services.

• **Symmetry** Nodes in the system are identical in terms of functionality, and require no or minimal node-specific configuration to function.

• **Simplicity** The system should be made as simple as possible, but no simpler.

How S3 Works

Amazon keeps its lips pretty tight about how S3 works, but according to Amazon, S3's design aims to provide scalability, high availability, and low latency at commodity costs. S3 stores arbitrary objects at up to 5GB in size, and each is accompanied by up to 2KB of metadata. Objects are organized by *buckets*. Each bucket is owned by an AWS account and the buckets are identified by a unique, user-assigned key.



Multiple objects are stored in buckets in Amazon S3.

Buckets and objects are created, listed, and retrieved using either a REST-style or SOAP interface.

Objects can also be retrieved using the HTTP GET interface or via BitTorrent. An access control list restricts who can access the data in each bucket. Bucket names and keys are formulated so that they can be accessed using HTTP. Requests are authorized using an access control list associated with each bucket and object, for instance:

http://s3.amazonaws.com/examplebucket/examplekey

http://examplebucket.s3.amazonaws.com/examplekey

The Amazon AWS Authentication tools allow the bucket owner to create an authenticated URL with a set amount of time that the URL will be valid.