MinIO

SCALABLE OBJECT STORAGE

MinIO Intro

MinIO is a high-performance, scalable object storage solution compatible with Amazon S3 APIs. It is designed for large-scale deployments and offers features like horizontal scaling and low latency.

MinIO Key Features

Scalable

Supports petabytes of data. Scales horizontally with ease.

High Performance

Optimized for high throughput and low latency.

S3 Compatibility

Seamless integration with S3 APIs. Supports common S3 operations.

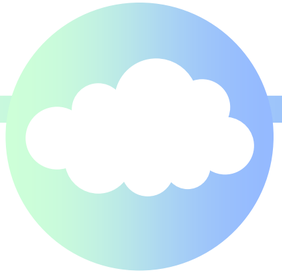
Open Source

Free and open-source software.Active community and regular updates.

Deployment Modes

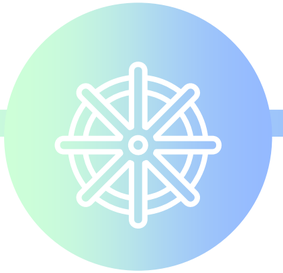
On Premise

Deploy on local infrastructure.such as windows, linux



On Cloud

Deploy on cloud platforms (AWS, Azure, GCP).

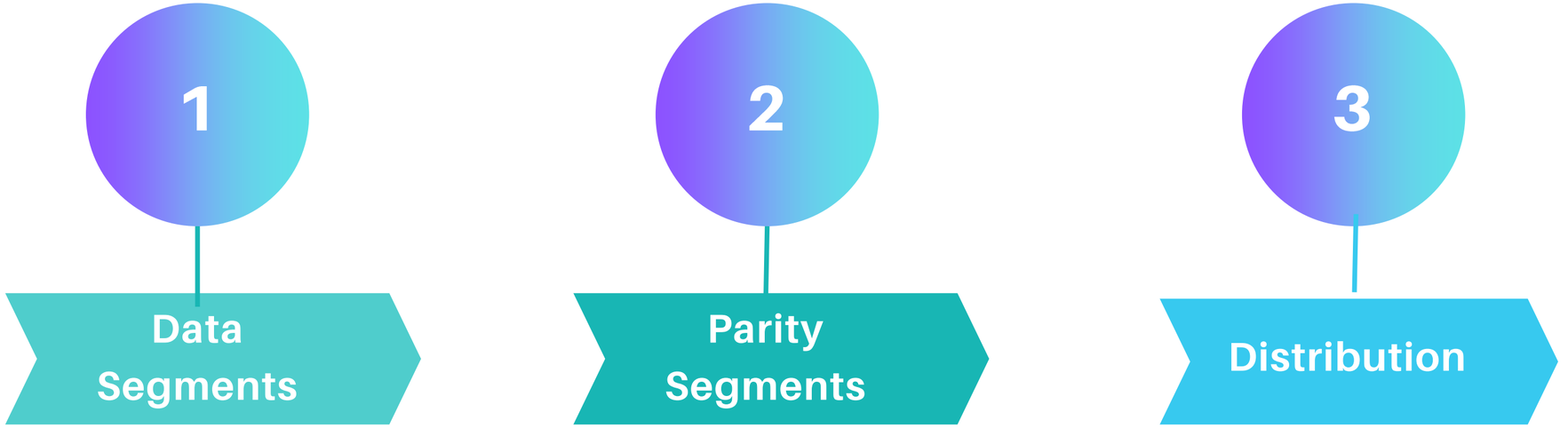


Kubernetes

Deploy as a containerized application in Kubernetes.

Erasure Coding in MinIO

Erasure Coding is a method for data protection by dividing data into segments and adding redundancy.



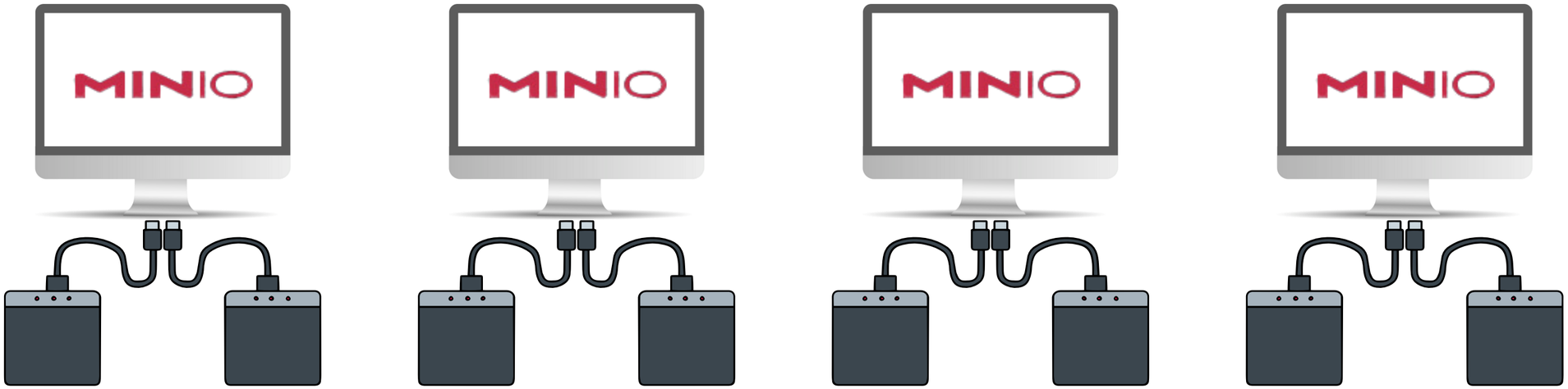
Original data split into smaller chunks.

Additional segments for redundancy.

Data and parity segments are spread across drives or nodes.

MinIO Server Pool

Components:
Drives: Storage devices within each server.
Nodes: Individual MinIO servers.
Erasure Sets: Logical groups of drives using erasure coding.



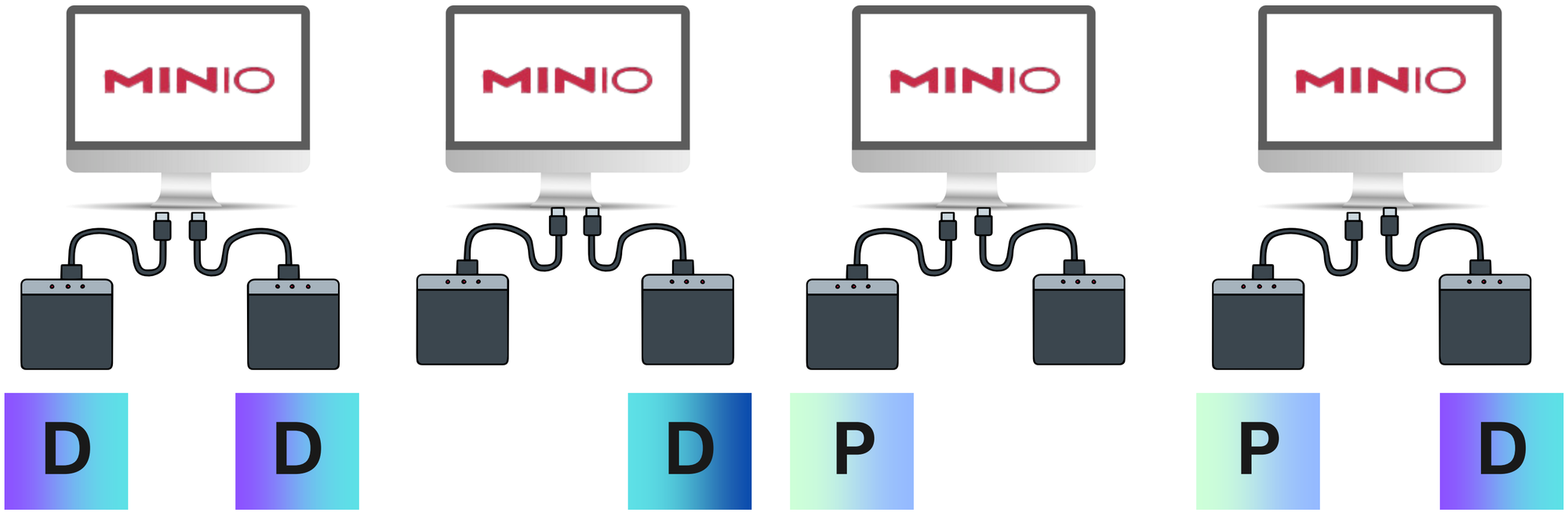
Erasure Set

Configuration within a server pool for distributing and protecting data.

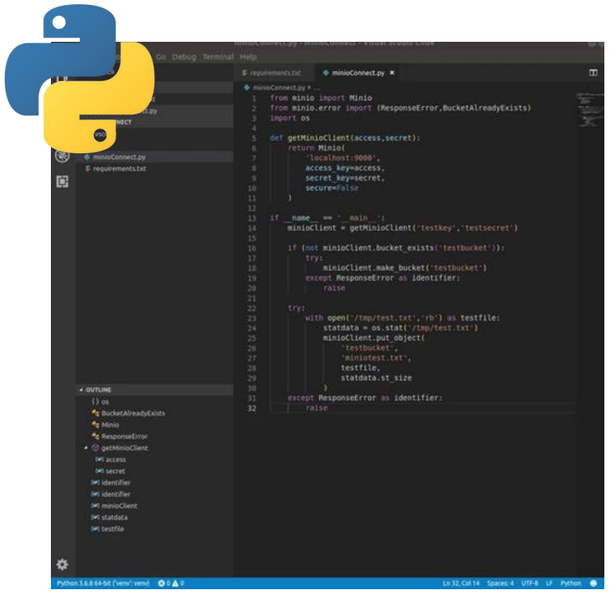
4+2 Erasure Set

Data is divided into 4 segments, with 2 parity segments for redundancy.

Tolerates up to 2 drive failures,



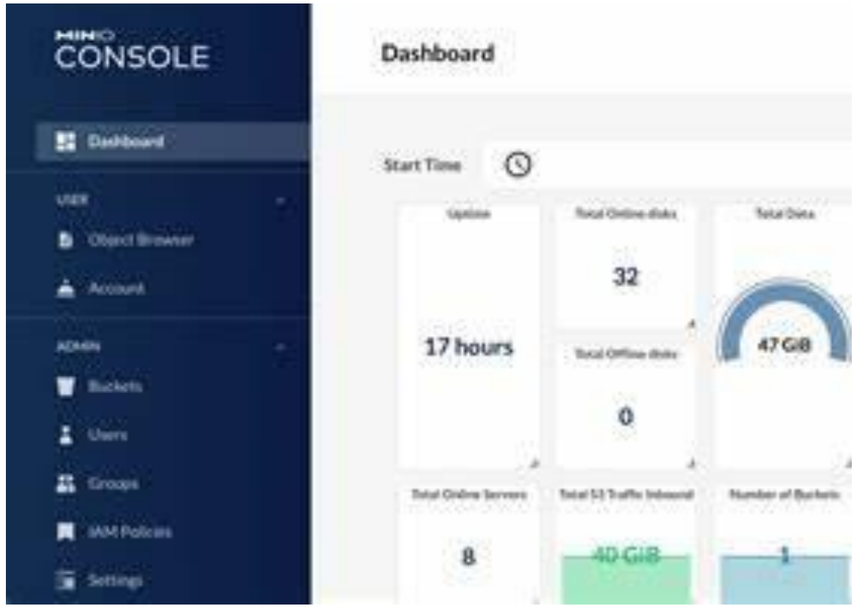
MinIO Clients




```
itslinux@foss: $ ./mc --help
NAME:
  mc - MinIO Client for cloud storage and filesystems.

USAGE:
  mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]

COMMANDS:
  alias      manage server credentials in configuration file
  ls         list buckets and objects
  mb         make a bucket
  rb         remove a bucket
  cp         copy objects
  mv         move objects
  rm         remove object(s)
  mirror     synchronize object(s) to a remote site
  cat        display object contents
  head       display first 'n' lines of an object
  pipe       stream STDIN to an object
  find       search for objects
  sql        run sql queries on objects
  stat       show object metadata
  tree       list buckets and objects in a tree format
  du         summarize disk usage recursively
```



MINIO KEY COMPONENTS

Buckets

Containers for storing objects.

Objects

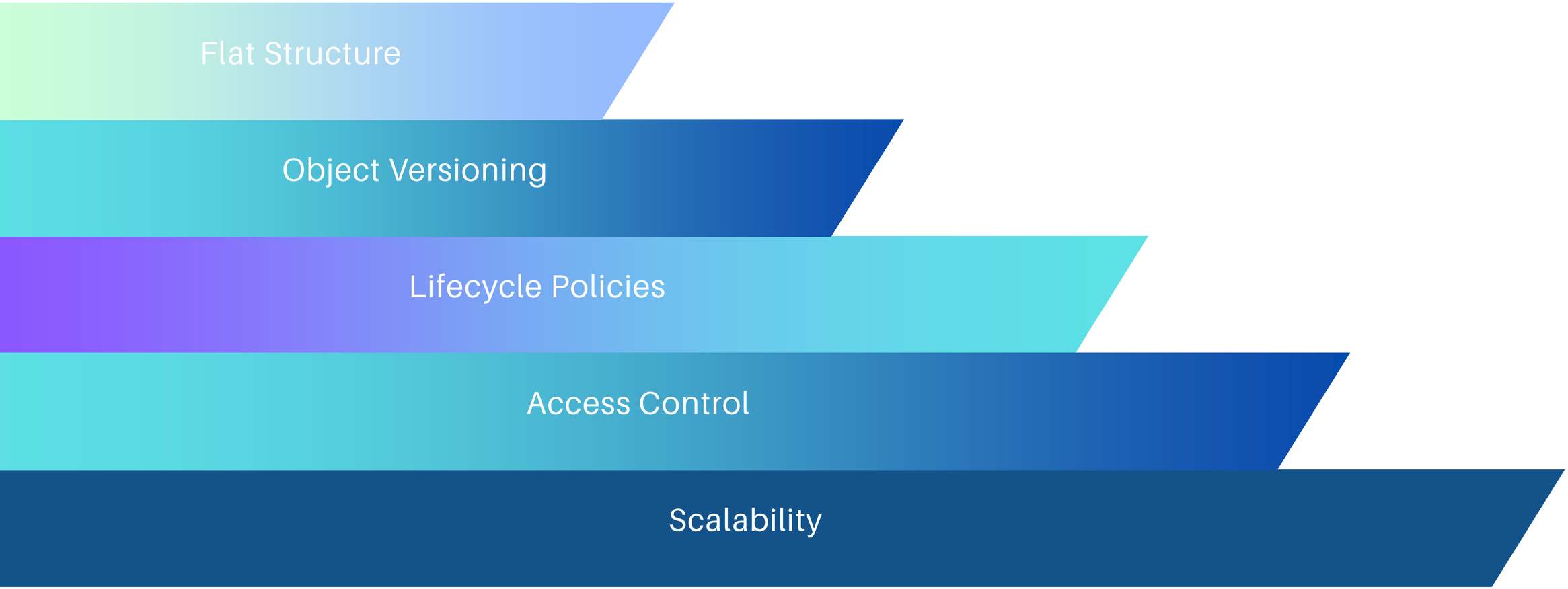
Data stored in buckets (files, images, etc.).

Policies

Control access and manage data lifecycle.

UNDERSTANDING MINIO BUCKETS

A bucket in MinIO is a container for storing objects, similar to folders in a file system.



BUCKET OPERATIONS

CREATE BUCKET

Instantiate a new bucket to store objects.

LIST BUCKETS

Retrieve a list of all buckets in the MinIO server.

DELETE BUCKET

Remove an existing bucket, including all its objects.

MinIO Object

An object is the fundamental unit of data storage, consisting of the data itself, metadata, and a unique identifier (key).

01	*Immutable*	Once an object is uploaded, it cannot be modified. To "update" an object, you must upload a new version.
02	*Unique Identifier*	Every object is identified by a unique key within a bucket, allowing for efficient data retrieval.
03	*Scalability*	Designed to store billions of objects across distributed systems.
04	*Support for Large Files*	MinIO supports multipart uploads, enabling the storage of large objects by breaking them into smaller part
05	*Data and Metadata*	Each object contains both the actual data and metadata (key-value pairs) that describe the object.

OBJECT OPERATIONS

UPLOAD

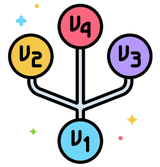
Store a new object in a bucket, assigning it a unique key.

RETRIEVE

Download the object using its key.

DELETE

Remove an object from a bucket.



VERSIONING

Retain multiple versions of an object.

VERSIONING IN MINIO

Versioning allows multiple versions of the same object to be stored in a MinIO bucket.

MULTIPLE VERSIONS

EVERY UPDATE TO AN OBJECT CREATES A NEW VERSION, ALLOWING YOU TO RETAIN A HISTORY OF CHANGES.

NON-DESTRUCTIVE OPERATIONS

Deleting an object does not permanently remove it; instead, it marks it as a delete marker, preserving older versions.

RESTORATION

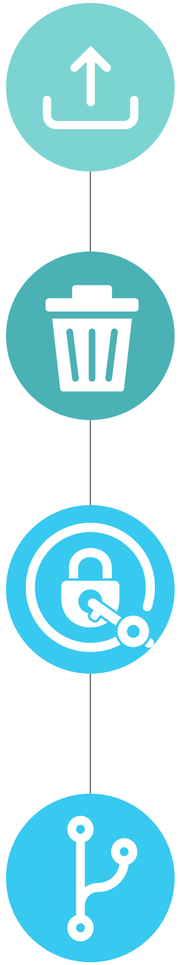
EASILY RESTORE ANY PREVIOUS VERSION OF AN OBJECT, ENSURING YOU CAN RECOVER FROM ACCIDENTAL DELETIONS OR OVERWRIT

VERSION CONTROL

ENABLE OR SUSPEND VERSIONING ON BUCKETS AS NEEDED, OFFERING FLEXIBILITY IN DATA MANAGEMENT.

How Versioning Works

]



Object Upload

When an object is uploaded, it is stored as the latest version

Object Deletion

Deleting an object does not remove it; instead, a delete marker is added, and previous versions remain

No In-Place Modification

Objects are immutable. To change an object, you have to upload a new version of it.

No Versioning

If versioning is not enabled, uploading an object with the same key will overwrite the existing object without retaining the previous version