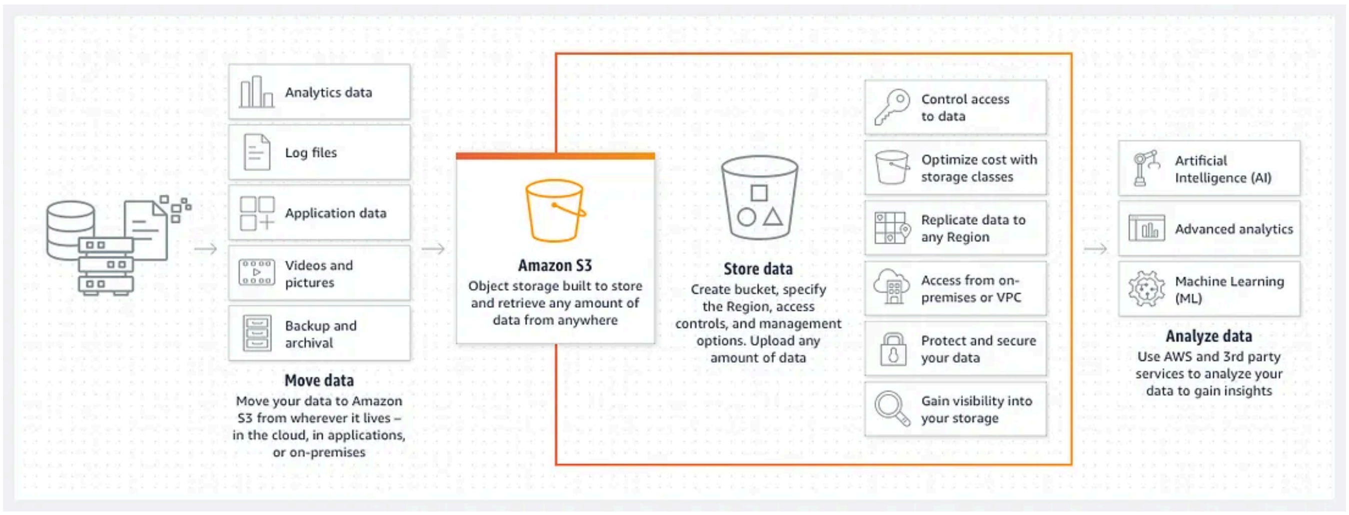
Amazon S3 in 3 Minutes: Quick Insights and Practical Examples for Easy Understanding

Nollie Chen · Follow
4 min read · Nov 7, 2023

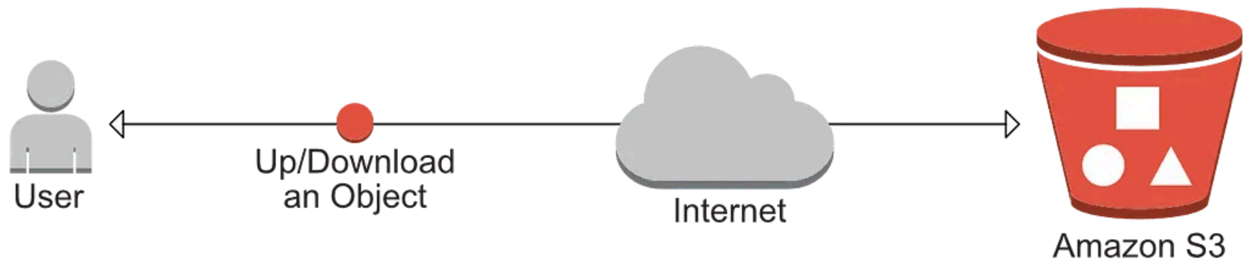
Listen Share More


Analytics data
Log files
Application data
Videos and pictures
Backup and archival
Move data
Move your data to Amazon S3 from wherever it lives – in the cloud, in applications, or on-premises
Amazon S3
Object storage built to store and retrieve any amount of data from anywhere
Store data
Create bucket, specify the Region, access controls, and management options. Upload any amount of data
Control access to data
Optimize cost with storage classes
Replicate data to any Region
Access from on-premises or VPC
Protect and secure your data
Gain visibility into your storage
Artificial Intelligence (AI)
Advanced analytics
Machine Learning (ML)
Analyze data
Use AWS and 3rd party services to analyze your data to gain insights

How does Amazon S3 work?

Amazon's Simple Storage Service, or Amazon S3, is a key player in the Amazon Web Services (AWS) suite. It's a scalable, highly available, and secure service that excels at providing object storage in the Cloud. This ability enables users to store and retrieve vast amounts of data from anywhere on the web.

The beauty of S3 lies in its flexibility and wide-ranging applications. It can act as a host for static websites, scaling as needed to accommodate traffic. It can serve as a secure storage location for database backups, ensuring data remains safe and easily retrievable. For those involved in data processing, S3 plays a crucial role in storing and retrieving data used in data processing pipelines, making it an invaluable tool in data analytics and machine learning applications.



pc: cloudonaut

At its core, Amazon's S3 service revolves around two key concepts: 'Buckets' and 'Objects'. A 'Bucket' is essentially a container or a folder used for storing data, referred to as 'Objects'. Each 'Bucket' must have a globally unique name across all AWS accounts, serving as a unique identifier. Users can create a hierarchical structure within these 'Buckets' using prefixes, simulating a folder-like structure.

'Objects' in these 'Buckets' are the actual data pieces users store in S3. An 'Object' can be any file type — from video files and JSON files to executable JAR files and ZIP files. With the maximum size of an 'Object' being 5TB, S3 offers ample space for large datasets.

Retrieving data from S3 can be done in several ways. While data can be accessed via a URL if its access permissions are set to public, a more common and secure method is to **retrieve data programmatically using AWS SDKs**. For instance, boto3, the AWS SDK for Python, provides an interface to Amazon S3.

Also, to help users manage costs and optimize for specific use cases, Amazon S3 offers various storage classes. These include the Standard Tier for frequently accessed data, the Intelligent Tier for data with changing access patterns, the Infrequent Access tier for less frequently accessed data, and the Glacier tiers for long-term archival. Each tier has different pricing and performance characteristics, allowing users to choose the one that best fits their use case.

To demonstrate how one can interact with S3, consider the following JavaScript code snippets using the AWS SDK. This sample code illustrates how to perform common operations such as uploading a file, retrieving a file, and deleting a file from an S3 bucket. It provides a practical example of how to use the 'Upload', 'getObject', and 'deleteObject' methods respectively.

```
// Sample code for S3 Operations
        const {
            Upload
        } = require("@aws-sdk/lib-storage");

        const {
            S3
        } = require("@aws-sdk/client-s3");

        // dotenv helps manage environment variables
        require('dotenv').config();


        const fs = require('fs');


        // The name of the bucket that you have created
        const BUCKET_NAME = 'nameitasyouwant';

        // we load credentials from the .env file
        const s3 = new S3({
            credentials: {
                accessKeyId: process.env.ID,
                secretAccessKey: process.env.SECRET
            },
            region: 'us-east-1',
        });


        // upload a file
        const uploadFile = async (fileContent, fileName) => {
            console.log('fileName', fileName);
            // Setting up S3 upload parameters
            const params = {
                Bucket: BUCKET_NAME,
                Key: fileName, // File name we want to upload
                Body: fileContent // the buffer
            };

            // Uploading files to the bucket

        const data = await  new Upload({
            client: s3,
            params
        }).done();
        console.log(`File uploaded successfully. ${data.Location}`);
        // return the URL of the object on S3
            return data.Location;
        };
```

```
// retrieve a file
const retrieveFile = (fileName) => {
    // Setting up S3 read parameters
    const params = {
        Bucket: BUCKET_NAME,
        Key: fileName, // File name we want to retrieve
    };

    // download file from the bucket
    s3.getObject(params, function(err, data) {
        if (err) {
            throw err;
        }
        console.log(`File downloaded successfully. ${data.Body}`);
        // do something with the file
        const fStream = fs.createWriteStream(`${fileName}`);
        fStream.write(data.Body);
         fStream.end();
        // return data
        return data.Body;
    });
};



// delete a file
const deleteFile = (fileName) => {
    // Setting up S3 delete parameters
    const params = {
        Bucket: BUCKET_NAME,
        Key: fileName, // File name we want to delete
    };

    // download file from the bucket
    s3.deleteObject(params, function(err, data) {
        if (err) {
            // throw err;
            return false;
        }
        console.log(`File deleted successfully. ${data}`);
        return true;
    });
};



module.exports ={uploadFile, retrieveFile, deleteFile}
```

In summary, Amazon S3 is an incredibly powerful and flexible storage service within the AWS ecosystem. Its durability, scalability, and high availability make it an excellent choice for a wide range of applications and use cases. Whether you're hosting a website, backing up data, or executing complex data processing tasks, Amazon S3 has the features and capabilities to meet your storage needs.

Relevant Resources: AWS Amazon S3 Getting Started

Aws S3 AWS JavaScript

Follow

Written by Nollie Chen

601 Followers · 22 Following

SDE Intern @AWS | @UPenn | CS gradute | nolliechy@gmail.com | ig: alconollie | linkedin: HuiYu(Nollie) Chen

No responses yet

What are your thoughts?

Respond

More from Nollie Chen