**AMAZON WEB SERVICE**

**Aws:**Aws is a cloud computing platform, that offers variety of services to help business and organization at lower cost and innovate faster.

It offers services like compute, store, database etc,.

**Features Of AWS:**

* Cost effective (follows pay as you go method)
* Scalability and elasticity
* Security and compliance
* Flexibility

**AWS S3:**

It is a highly scalable object storage service that allows you to store data manage data in the cloud.

It supports various types of data like structured,semi-structured and unstructured data.

**Key Concepts:**

**Bucket**: it is a container for objects. Where we can store any number of files.

**Object**: object is a single file within a bucket

**Key**: a unique identifier for an object(file path)

**Tags**: Tags are key-value pairs used to categorize and manage S3 objects.

**Key Features:**

* High availability and duarabilty
* S3 automatically replicates across multiple availability zones
* Security
* Performance
* Cost effective

**Boto3:**

Boto3 is the official AWS SDK for Python. It's a powerful library that simplifies interactions with various AWS services, including Amazon S3. With Boto3, you can efficiently manage and utilize the vast capabilities of AWS from within your Python applications like like uploading, downloading, and managing objects.

1. create\_bucket

* Creates a new S3 bucket to store objects.
* Use case: Storing static website content, data backups, or application data.

2. delete\_bucket

* Deletes an empty S3 bucket.
* Use case: Removing unused buckets to save storage costs.

3. list\_buckets

* Lists all S3 buckets associated with your AWS account.
* Use case: Managing and monitoring your S3 storage.

4. upload\_fileobj

* Uploads a file-like object to an S3 bucket.
* Use case: Backing up local files, deploying application code, or storing user-generated content.

5. download\_fileobj

* Downloads an object from an S3 bucket to a file-like object.
* Use case: Retrieving files for local processing or analysis.

6. delete\_object

* Deletes an object from an S3 bucket.
* Use case: Removing unwanted or outdated files to save storage costs.

7. list\_objects\_v2

* Lists objects in an S3 bucket.
* Use case: Inventorying objects, searching for specific files, or implementing data lifecycle policies.

8. copy\_object

* Copies an object within or between S3 buckets.
* Use case: Creating backups, replicating data across regions, or reorganizing object storage.

9. put\_object\_acl

* Sets the access control list (ACL) for an object.
* Use case: Controlling who can access your objects and with what permissions.

10. put\_object\_tagging

* Adds tags to an object.
* Use case: Organizing and categorizing objects for easier search and retrieval.

**To create s3 session:**

import boto3

# Create a session using your AWS credentials

session = boto3.Session(

aws\_access\_key\_id='YOUR\_ACCESS\_KEY',

aws\_secret\_access\_key='YOUR\_SECRET\_KEY',

region\_name='YOUR\_REGION' # e.g., 'us-west-1'

)

# Connect to the S3 service

s3 = session.resource('s3')

**CRUD Operations:**

**Uploading object into S3:**

def upload\_file(bucket\_name, file\_name, object\_name=None):

if object\_name is None:

object\_name = file\_name

try:

s3.Bucket(bucket\_name).upload\_file(file\_name, object\_name)

print(f"{file\_name} has been uploaded to {bucket\_name}/{object\_name}")

except Exception as e:

print(f"Failed to upload {file\_name} to {bucket\_name}/{object\_name}: {e}")

**Downloading a file from S3:**

def download\_file(bucket\_name, object\_name, file\_name=None):

if file\_name is None:

file\_name = object\_name

try:

s3.Bucket(bucket\_name).download\_file(object\_name, file\_name)

print(f"{object\_name} has been downloaded from {bucket\_name} to {file\_name}")

except Exception as e:

print(f"Failed to download {object\_name} from {bucket\_name}: {e}")

**Listing objects in an S3 bucket:**

def list\_objects(bucket\_name):

try:

bucket = s3.Bucket(bucket\_name)

for obj in bucket.objects.all():

print(obj.key)

except Exception as e:

print(f"Failed to list objects in {bucket\_name}: {e}")

**Deleting an object from S3:**

def delete\_file(bucket\_name, object\_name):

try:

s3.Object(bucket\_name, object\_name).delete()

print(f"{object\_name} has been deleted from {bucket\_name}")

except Exception as e:

print(f"Failed to delete {object\_name} from {bucket\_name}: {e}")

**Adding tag to object:**

import boto3

s3 = boto3.client('s3')

response = s3.put\_object\_tagging(

Bucket='your-bucket-name',

Key='your-object-key',

Tagging={'TagSet': [

{'Key': 'tag1', 'Value': 'value1'},

{'Key': 'tag2', 'Value': 'value2'},

]}

)