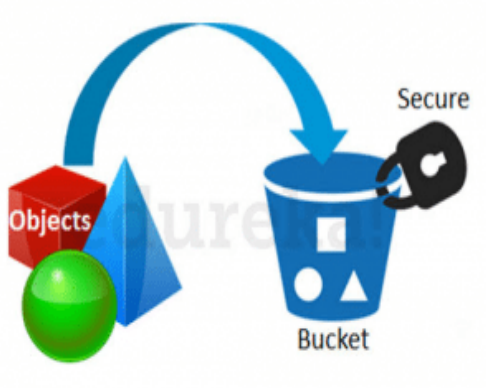
Storage in AWS

Dr. R. Karthi

Simple Storage Service (S3)

- S3 is a storage service used to store and retrieve any amount of data at any time, from anywhere on the web.
- S3 provides highly scalable, fast, inexpensive data storage infrastructure on the web.
- S3 is an object based storage where data is stored as objects within buckets.
- Buckets are containers for objects.
- Simple web interface is used access and store the data in buckets.

S3



- S3 is a object based storage.
- Objects can be of size 0 to 5TB in any format.
- Objects can be text file, html pages, images etc.
- Bucket name has to unique and is a global resource.
- Objects/ files in a bucket have the following properties:
  - key – name of the object.
  - value – data of the object.
  - Versionid – version of the object
  - Metadata – owner of the file, other information
- User can set control access for each bucket, choose the geographical region and view access logs for the bucket and its objects.
- Amazon guarantee 99.99% availability over a given year – (availability - storage system is operational and can deliver data upon request).
- Amazon guarantee 99.999999999% Durability across multiple Availability Zones (Durability - stored data does not suffer from degradation or corruption).

Storage Classes or Tiers

- S3 Standard - General-purpose storage for frequently accessed data.
- S3 Intelligent-Tiering - Data with unknown or changing access patterns.
  - Data is moved Automatically between two access tiers (S3 and S3-IT) based on changing access patterns
- S3 Standard-IA - for long-lived, but less frequently accessed data. The data store for disaster recovery of files.
- S3 One Zone-IA - stores data in a single AZ, storing secondary backup copies of on-premises data or easily re-creatable data.
- S3 Glacier - Low-cost design and is ideal for long-term archive.
- S3 Glacier Deep Archive for long-term archive and digital preservation for regulatory purpose.

IA – infrequent access **AZ- Availability Zone**

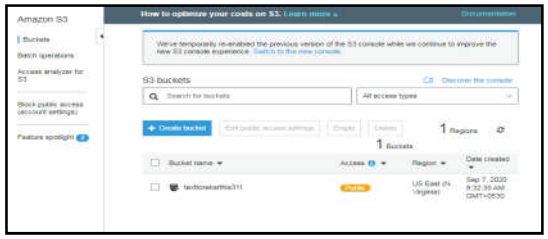
Storage Classes or Tiers

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

SLA - Service level Agreement

S3

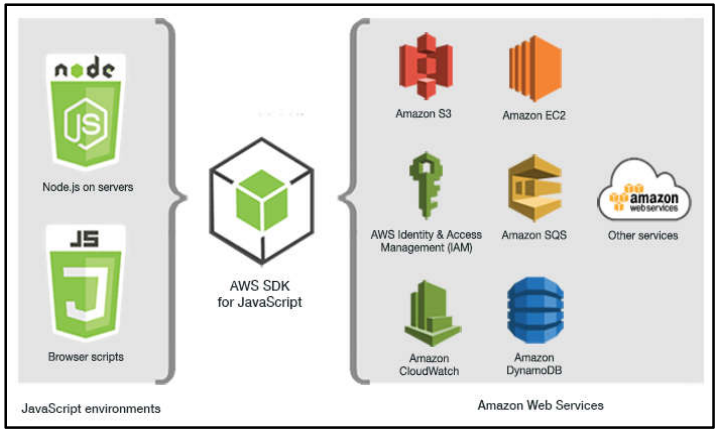
- Using AWS console – GUI based interface



- Using CLI - (Command Line Interface)
  - AWS CLI is a unified tool to manage your AWS services.
  - Control multiple AWS services

```
Karthi@Karthi-Dell MINGW64 /e/CloudComputing/2020/ClassMaterial/S3 Demos/s3sampl
es
$ aws s3 ls
2020-09-07 09:32:38 texttorekarthis311
```

- Using Node.js - aws sdk
  - JavaScript API for AWS services



Using CLI

- cp - Copies a local file or S3 object to another location locally or in S3.
  - aws s3 cp myfolder s3://mybucket/folder –recursive
- ls - List S3 objects and all S3 buckets.
  - aws s3 ls s3://mybucket
- mb - Creates an S3 bucket.
  - aws s3 mb s3://newbucketkarthi
- mv - Moves a local file or S3 object to another location locally or in S3.
  - aws s3 mv readCSV.js s3://karthimybucket123
- rb – Deletes S3 bucket.
  - aws s3 rb s3://newbucketkarthi –force
- rm - Deletes an S3 object.
  - aws s3 rm s3://mybucket/folder --recursive
- sync - Recursively copies new and updated files from the source directory to the destination.
  - aws s3 sync myfolder s3://mybucket/folder

Using Node.js – Create a Bucket

```
   // CREATE BUCKET
var AWS = require('aws-sdk');
var s3bucket = new AWS.S3();   // service interface object
 var params = {Bucket: 'textkarthis311', ACL: 'public-read'};
 s3bucket.createBucket(params, function(err, data) {
    if (err) {
       console.log("Error", err);
     } else {
       console.log("Success", data.Location);
     }
});
```

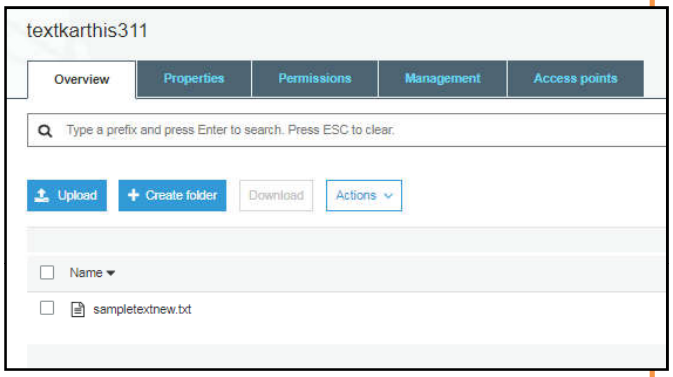

```
E:\CloudComputing\2020\ClassMaterial\S3 Demos\s3samples>node create_bucket_new.js
Success /textkarthis311
```

Using Node.js

Insert_bucket_file.js

```
var AWS = require("aws-sdk");
var s3bucket = new AWS.S3();
 var params = {Bucket: 'textkarthis311', Key:'sampletextnew.txt',
 Body: 'Hello this data!'};


   s3bucket.putObject(params, function(err, data) {
       if (err) {
           console.log("Error uploading data: ", err);
       } else {
           console.log("Successfully uploaded data to bucket");
       }
    });
```




```
E:\CloudComputing\2020\ClassMaterial\S3 Demos\s3samples>node insert_bucket_file.js
Successfully uploaded data to bucket
```

Using Node.js

download_bucket_file.js

```
var AWS = require('aws-sdk');
var s3bucket = new AWS.S3();
var params = {Bucket: 'textkarthis311', Key:'sampletextnew.txt'};
    s3bucket.getObject(params, function(err, data) {
        if (err) {
            console.log("Error uploading data: ", err);
        } else {
            console.log("Successfully dowloaded data from bucket");
            const body = Buffer.from(data.Body).toString('utf8');
            console.log(body);
        }
    });
```

```
E:\CloudComputing\2020\ClassMaterial\S3 Demos\s3samples>node download_bucket_file.js
Successfully dowloaded data from bucket
Hello this data!
```

Using Node.js

upload.js

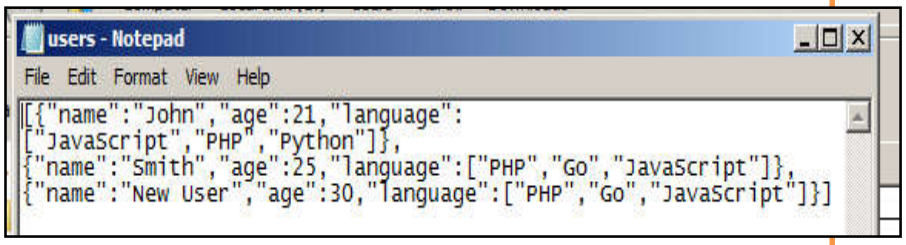
```
const fs = require('fs');
const AWS = require('aws-sdk');
const s3 = new AWS.S3( );
const fileName = 'sample.txt';
const uploadFile = () => {
  fs.readFile(fileName, (err, data) => {
     if (err) throw err;
     const params = {
         Bucket: 'textkarthis311', // pass your bucket name
         Key: 'S3_sample.txt', // file will be saved
         Body: data
    };
    s3.upload(params, function(s3Err, data) {
         if (s3Err) throw s3Err
         console.log(`File uploaded successfully at ${data.Location}`)
     });
  });
};
uploadFile();
```

```
E:\CloudComputing\2020\ClassMaterial\S3 Demos\s3samples>node upload.js
File uploaded successfully at https://textkarthis311.s3.amazonaws.com/S3_sample.txt
```

Using Node.js

Read_json.js

```
const fs = require('fs');
const AWS = require('aws-sdk');
 var s3 = new AWS.S3({apiVersion: '2006-03-01'});
 var params = {Bucket: 'karthimybucket123', Key: 'users.json',
     ResponseContentType: 'application/json' };
  const f = s3.getObject(params, function(err, data) {
    if (err) {
        console.log("Error uploading data: ", err);
    } else {
        console.log("Successfully uploaded data to bucket");
        x = data.Body.toString('utf-8');
        console.log(x);
        dat = JSON.parse(x);
        console.log(dat.length);
         for (i = 0; i < dat.length; i++) {
           console.log(dat[i].name, dat[i].age);
         }
    }
});
```




```
E:\CloudComputing\2020\ClassMaterial\S3 Demos\s3samples\Json_example>node read_json.js
Successfully uploaded data to bucket
[{"name":"John","age":21,"language":["JavaScript","PHP","Python"]},{"name":"Smith","age":25,"language":["PHP","Go","JavaScript"]},{"name":"New User","age":3
"language":["PHP","Go","JavaScript"]}]
3
John 21
Smith 25
New User 30
```

Using Node.js

Imagereadwrite.js

```
// Image Read and Write
var AWS = require("aws-sdk");
const fs = require('fs');
var path = require('path');
var s3 = new AWS.S3({apiVersion: '2006-03-01'});
var fileStream = fs.createWriteStream('E:/file1.jpg');
var s3Stream = s3.getObject({Bucket: 'karthimybucket123', Key: 'Chrysanthemum.jpg'}).
    createReadStream();
 // Listen for errors returned by the service
s3Stream.on('error', function(err) {
    // NoSuchKey: The specified key does not exist
    console.error(err);
});
 s3Stream.pipe(fileStream).on('error', function(err) {
    // capture any errors that occur when writing data to the file
    console.error('File Stream:', err);
}).on('close', function() {
    console.log('Done.');
});
```

```
E:\CloudComputing\2020\ClassMaterial\S3 Demos\s3samples>node Imagereadwrite.js
Done.
```

Functions

- getObject(params = {}, callback) - Retrieves objects from Amazon S3.
- putObject(params = {}, callback) - Adds an object to a bucket.
- upload(params = {}, [options], [callback]) - Uploads an arbitrarily sized buffer, blob, or stream to the object.
- buffer.toString(encoding, start, end) - toString() method returns the buffer object according to the specified encoding.
- JSON.parse(x) - Parsing the data with JSON.parse(), the data becomes a JavaScript object.
- fs.createWriteStream(path[, options]) – creates a writeable stream object to the filepath to start streaming out.
- fs.createReadStream(path[, options]) – creates a readable stream object to the file to start streaming in.
- Object.On(event, callback) - on() method requires name of the event to handle and callback function which is called when an event is raised.
- stream.pipe() – pipe method is used to take a readable stream and connect it to a writeable stream

Thank You