## Uploading Files to AWS S3 with Node.js

### What is S3?

S3, or Simple Storage Service, is a cloud storage service provided by Amazon Web Services (AWS). Using S3, you can host any number of files while paying for only what you use.

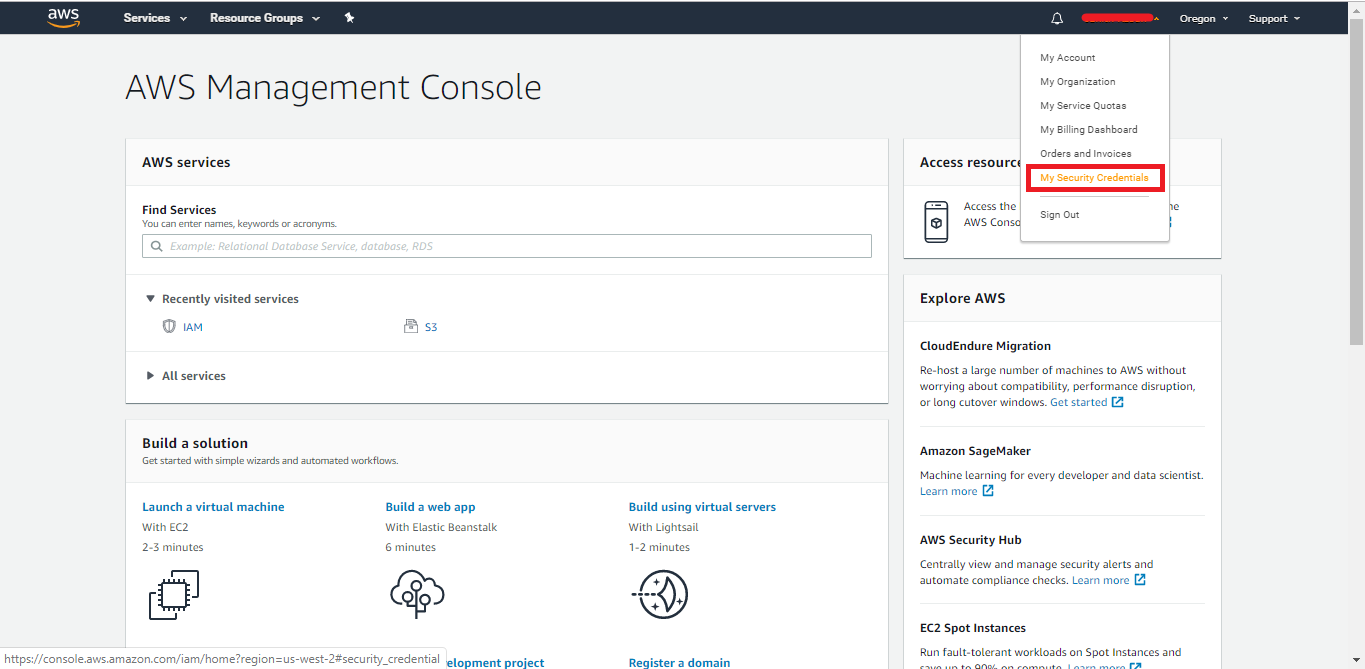
S3 also provides multi-regional hosting to customers by their region and thus are able to really quickly serve the requested files with minimum delay.

### Setting up the Environment

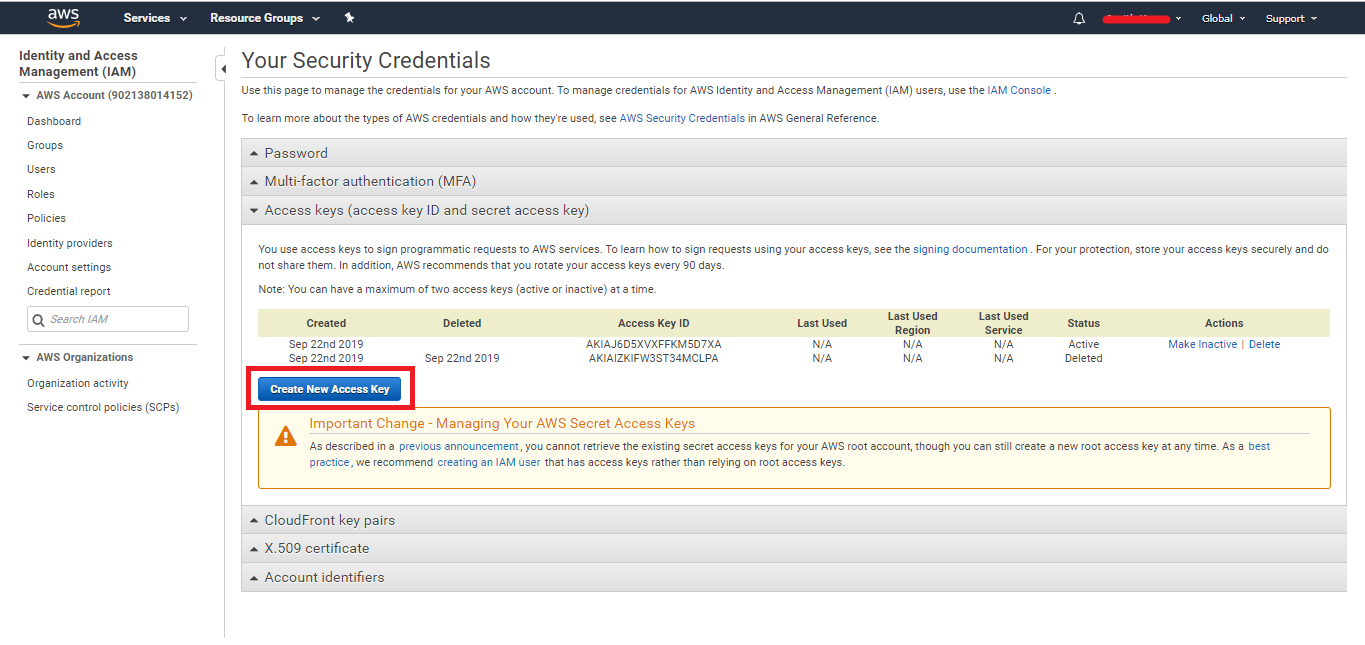
#### AWS Credentials

To get started, you need to generate the AWS Security Key Access Credentials first. To do so, login to your AWS Management Console.

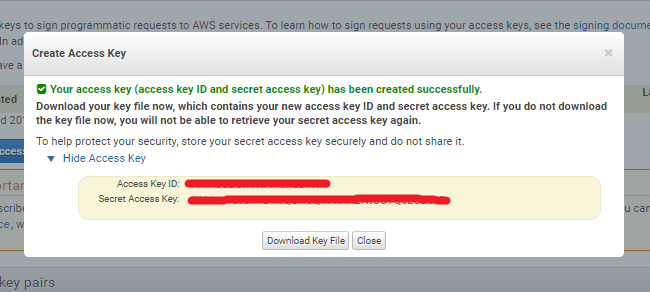
Click on your username:



Then select Access Keys -> Create New Access Key:



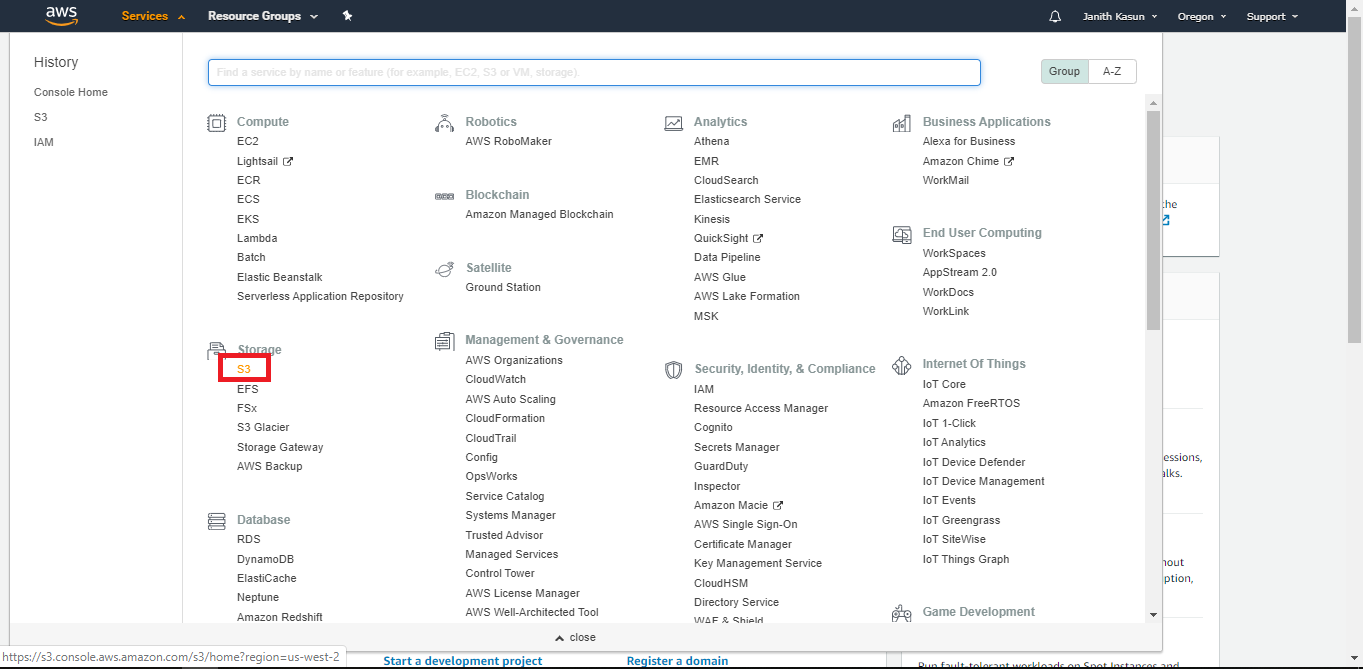
After that you can either copy the Access Key ID and Secret Access Key from this window or you can download it as a .CSV file:



#### Creating an S3 Bucket

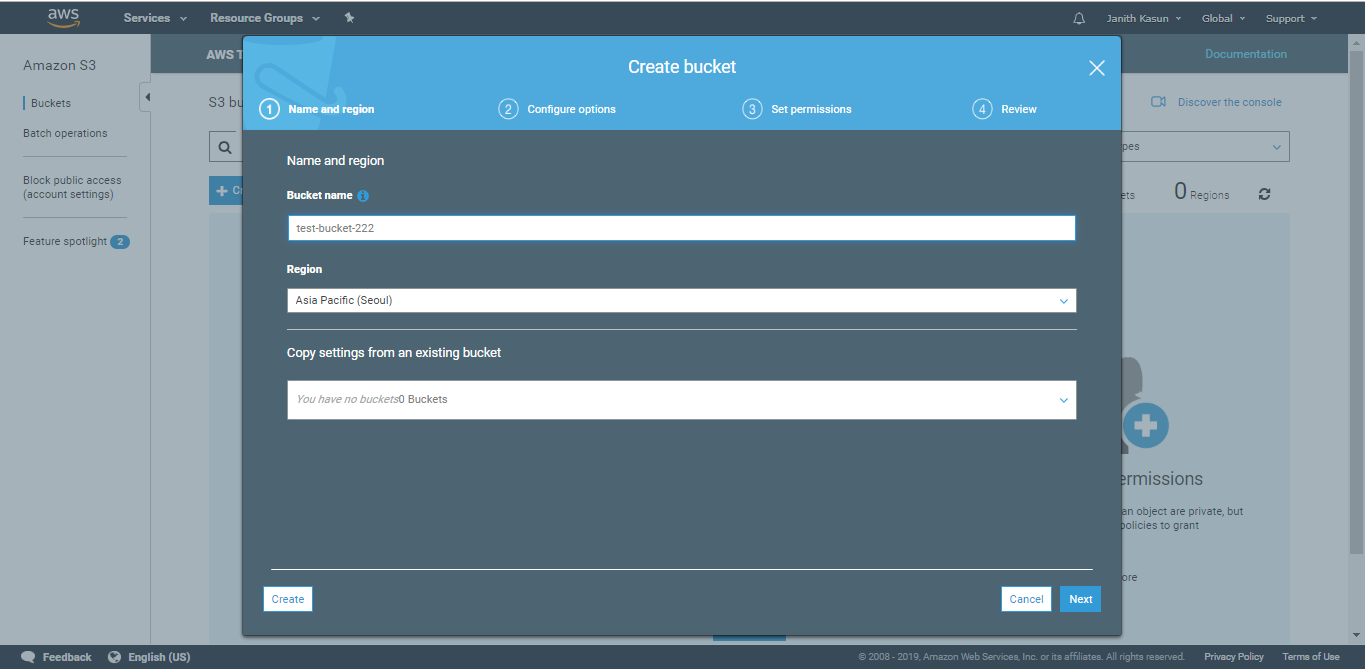
Now let's create a AWS S3 Bucket with proper access. We can do this using the AWS management console or by using Node.js.

To create an S3 bucket using the management console, go to the S3 service by selecting it from the service menu:



Select "Create Bucket" and enter the name of your bucket and the region that you want to host your bucket. If you already know from which region the majority of your users will come from, it's wise to select a region as close to their's as possible. This will ensure that the files from the server will be served in a more optimal timeframe.

The name you select for your bucket should be a unique name among all AWS users, so try a new one if the name is not available:



Follow through the wizard and configure permissions and other setting per your requirements.

To create the bucket using Node.js, we'll first have to set up our development environment.

**Development Environment**

So now let's get started with our example by configuring a new Node.js project:

$ npm init

To start using any AWS Cloud Services in Node.js, we have to install the *AWS SDK* (System Development Kit).

Install it using your preferred package manager - we'll use npm:

$ npm i --save aws-sdk

**Implementation**

**Creating an S3 Bucket**

If you have already created a bucket manually, you may skip this part. But if not, let's create a file, say, create-bucket.js in your project directory.

Import the aws-sdk library to access your S3 bucket:

const AWS = require('aws-sdk');

Now, let's define three constants to store ID, SECRET, and BUCKET\_NAME. These are used to identify and access our bucket:

*// Enter copied or downloaded access ID and secret key here*

const ID = '';

const SECRET = '';

*// The name of the bucket that you have created*

const BUCKET\_NAME = 'test-bucket';

Now, we need to initialize the S3 interface by passing our access keys:

const s3 = new AWS.S3({

accessKeyId: ID,

secretAccessKey: SECRET

});

With the S3 interface successfully initialized, we can go ahead and create the bucket:

const params = {

Bucket: BUCKET\_NAME,

CreateBucketConfiguration: {

*// Set your region here*

LocationConstraint: "eu-west-1"

}

};

s3.createBucket(params, function(err, data) {

if (err) console.log(err, err.stack);

else console.log('Bucket Created Successfully', data.Location);

});

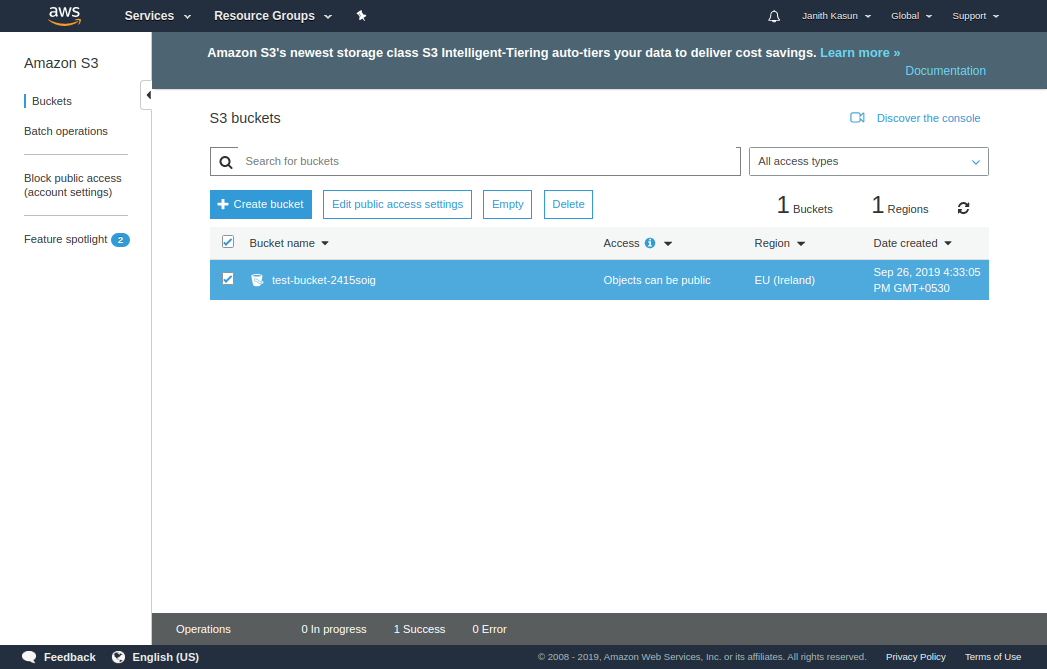
At this point we can run the code and test if the bucket is created on the cloud:

$ node create-bucket.js

If the code execution is successful you should see the success message, followed by the bucket address in the output:

Bucket Created Successfully http://test-bucket-2415soig.s3.amazonaws.com/

You can visit your S3 dashboard and make sure the bucket is created:



#### Uploading Files

At this point, let's implement the file upload functionality. In a new file, e.g. upload.js, import the aws-sdk library to access your S3 bucket and the fs module to read files from your computer:

const fs = require('fs');

const AWS = require('aws-sdk');

We need to define three constants to store ID, SECRET, and BUCKET\_NAME and initialize the S3 client as we did before.

Now, let's create a function that accepts a fileName parameter, representing the file we want to upload:

const uploadFile = (fileName) => {

*// Read content from the file*

const fileContent = fs.readFileSync(fileName);

*// Setting up S3 upload parameters*

const params = {

Bucket: BUCKET\_NAME,

Key: 'cat.jpg', *// File name you want to save as in S3*

Body: fileContent

};

*// Uploading files to the bucket*

s3.upload(params, function(err, data) {

if (err) {

throw err;

}

console.log(`File uploaded successfully. ${data.Location}`);

});

};

Before we upload the file, we need to read its contents as a buffer. After reading it, we can define the needed parameters for the file upload, such as Bucket, Key, and Body.

Besides these three parameters, there's a long list of other optional parameters. To get an idea of the things you can define for a file while uploading, here are a few useful ones:

* StorageClass: Define the class you want to store the object. S3 is intended to provide fast file serving. But in case files are not accessed frequently you can use a different storage class. For an example, if you have files which are hardly touched you can store in "S3 Glacier Storage" where the price is very low compared to "S3 Standard Storage". But it will take more time to access those files in case you need it and is covered with a different service level agreement.
* ContentType: Sets the image MIME type. The default type will be "binary/octet-stream". Adding a MIME type like "image/jpeg" will help browsers and other HTTP clients to identify the type of the file.
* ContentLength: Sets the size of the body in bytes, which comes in handy if body size cannot be determined automatically.
* ContentLanguage: Set this parameter to define which language the contents is in. This will also help HTTP clients to identify or translate the content.

For the Bucket parameter, we'll use our bucket name, whereas for the Key parameter we'll add the file name we want to save as, and for the Body parameter, we'll use fileContent.

With that done, we can upload any file by passing the file name to the function:

uploadFile('cat.jpg');

You can replace "cat.jpg" with a file name that exists in the same directory as the code, a relative file path, or an absolute file path.

At this point, we can run the code and test out if it works:

$ node upload.js

If every thing is fine, you should see output like what is shown below with a link to your file, which is stored in data.Location:

File uploaded successfully. https://test-bucket-1242tsr.s3.ap-northeast-2.amazonaws.com/cat.jpg

If there is any error it should be displayed on the console as well.

Additionally, you can go to your bucket in the AWS Management Console and make sure the file is uploaded.

