**Uploading and Downloading Files in S3 with Node.js**

[AWS S3](https://aws.amazon.com/s3/) is probably the most utilised AWS storage services. It is affordable, highly available, convenient and easy to use. To interact with any AWS services, Node.js requires [AWS SDK for JavaScript](https://aws.amazon.com/sdk-for-node-js/).

Let’s first create a project folder called nodeS3 and install SDK. Then, create the main program file and data folder. In the data folder, drop any file you want. In this example, I am using a json file called data.json.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

mkdir nodeS3

npm init -y

npm install aws-sdk

touch app.js

mkdir data

[/cc]

Next, you need to create a bucket for uploading a file (after [configuring your AWS CLI](https://docs.aws.amazon.com/lambda/latest/dg/setup-awscli.html)). Let’s create a bucket with the s3 command.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

Aws s3 mb s3://your.bucket.name

[/cc]

**Uploading File**

First of all, you need to import the aws-sdk module and create a new S3 object. It uses the credentials that you set for your AWS CLI. Locking in API version for S3 object is optional. Here is the further document on the [S3 class](https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/S3.html).

There are two methods you can use to upload a file, upload() and putObject(). Both methods are using different API calls. The major difference is upload() allows you to define concurrency and part size for large files while putObject() has lesser control. For a smaller file, both methods are fine. In general, I recommend to use upload().

**Simple File Upload Example**

In this example, we are using the async readFile function and uploading the file in the callback. As the file is read, the data is converted to a binary format and passed it to the upload Body parameter.

<pre class=" toolbar:2 lang:js decode:true" >

</pre>

**Downloading File**

To download a file, we can use getObject().The data from S3 comes in a binary format. In the example below, the data from S3 gets converted into a String object with toString() and write to a file with writeFileSync method. Alternatively, you can create the stream reader on getObject method and pipe to a stream writer as described [here](https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/requests-using-stream-objects.html).

<pre class=" toolbar:2 lang:js decode:true" >

</pre>