Node.js SDK 文档

概述

本文档基于 AWS Node.js SDK 编写。

开发准备

SDK 获取

为 Node.js 安装 SDK 的首选方法是使用 Node.js 程序包管理器 npm。要执行此操作，请在命令行中键入此内容。

npm install aws-sdk

说明

如果您在安装中遇到如下错误消息：

npm WARN deprecated node-uuid@1.4.8: Use uuid module instead

请在命令行中输入如下命令：

npm uninstall --save node-uuid

npm install --save uuid

快速入门

> 确认您已经理解 `对象存储` 基本概念，如 `Bucket`、`Object`、`Endpoint`、`AccessKey` 和 `SecretKey` 等。

下面介绍如何使用node.js SDK来访问对象存储服务，包括查看Bucket列表，上传文件，下载文件，查看文件列表等。

查看Bucket列表

```
const AWS = require('aws-sdk');
AWS.config.update({
     "accessKeyId": "access_key",
     "secretAccessKey": "secret_Key",
     "endpoint": "http://oss-cnbj01.cdsgss.com",
     "sslEnabled": false,
     "s3ForcePathStyle": true
});
const s3 = new AWS.S3();
s3.endpoint = new AWS.Endpoint("http://oss-cnbj01.cdsgss.com");
//Client初始化结束
//列出该用户拥有的桶
s3.listBuckets(function(err,data){
  if(err){console.log(err);}
  else{console.log (data);}
});
```

- 代码中，access_key 与 secret_key 是在对象存储服务中创建用户后，在页面上申请到的给用户用于访问对象存储的认证信息。而url是对象存储服务提供的服务地址及端口。

  在初始化之后，就可以利用s3进行各种操作了。

> 获取用户所拥有的bucket列表。用户必须有有效的Access Key。匿名请求将不允许此操作。

新建Bucket

创建桶。桶用于对象存储用户上传的对象。

```
//创建桶
const params = {
    Bucket:'oss-test'
};

s3.createBucket(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log('Bucket Created Successfully', data.Location);
});
```

上传文件

把当前目录下的oss.js文件上传到对象存储服务器

```
const fs = require('fs');
const fileContent = fs.readFileSync("oss.js");
const params = {
    Bucket: 'oss-test',
    Key: 'oss.js', // File name you want to save as in S3
    Body: fileContent
};
// Uploading files to the bucket
s3.upload(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log('File uploaded successfully', data.Location);
});
```

下载文件

把对象存储服务器上的Object下载到本地文件`downloaded.js`。

```
    var s3Params = {
        Bucket: 'oss-test',
        Key: 'oss.js'
    };
    const filePath = './downloaded.js';
    s3.getObject(s3Params, function(err, res) {
        if (err) console.error(err);
        fs.writeFileSync(filePath, res.Body.toString());
        console.log(filePath,' has been created!');
    });
```

列举文件

列举Bucket下的文件：

```
    var listparams = {
      Bucket: 'oss-test',
      Delimiter: '',
    }

    s3.listObjects(listparams, function (err, data) {
      if(err) console.log(err, err.stack);
      console.log(data);
    });
```

删除文件

删除对象存储服务器端桶下的对象

```
var deleteParams = {
    Bucket: 'oss-test',
    Key: 'oss.js'
};
s3.deleteObject(deleteParams, function(err, res) {
    if (err) console.error(err);
    console.log('object has been deleted!');
});
```

初始化

确定EndPoint

请先阅读理解对象存储中的AccessKey，SecretKey，Endpoint相关的概念。

下面的代码设置对象存储的访问域名为Endpoint参数：

```
//初始化http连接
const AWS = require('aws-sdk');
AWS.config.update({
     "accessKeyId": "access_key",
     "secretAccessKey": "secret_Key",
     "endpoint": "http://oss-cnbj01.cdsgss.com",
     "sslEnabled": false,
     "s3ForcePathStyle": true
});
const s3 = new AWS.S3();
s3.endpoint = new AWS.Endpoint("http://oss-cnbj01.cdsgss.com");
//初始化https
const AWS = require('aws-sdk');
const https = require('https');
//Client初始化
AWS.config.update({
     "accessKeyId": "testak",
     "secretAccessKey": "33774a2e8567",
     "endpoint": "oss-cnbj01.cdsgss.com",
     "sslEnabled":true,
     "s3ForcePathStyle": true,
     "s3DisableBodySigning": false,
     //"signatureVersion":"v2",
     //"signature_v2":true
});
const s3 = new AWS.S3();
s3.endpoint = new AWS.Endpoint("oss-cnbj01.cdsgss.com");
```

管理存储空间

存储空间（Bucket）是对象存储服务器上的命名空间，也是计费、权限控制、日志记录等高级功能的管理实体。

查看所有Bucket

可以遍历所有的Bucket的对象：

```
const AWS = require('aws-sdk');
AWS.config.update({
     "accessKeyId": "access_key",
     "secretAccessKey": "secret_Key",
     "endpoint": "http://oss-cnbj01.cdsgss.com",
     "sslEnabled": false,
     "s3ForcePathStyle": true
});
const s3 = new AWS.S3();
s3.endpoint = new AWS.Endpoint("http://oss-cnbj01.cdsgss.com");
//Client初始化结束
//列出该用户拥有的桶
s3.listBuckets(function(err,data){
  if(err){console.log(err);}
  else{console.log (data);}
});
```

创建Bucket

```
//创建桶
const params = {
    Bucket:'oss-test'
};
s3.createBucket(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log('Bucket Created Successfully', data.Location);
});
```

删除Bucket

删除指定的桶。只有该桶中所有的对象被删除了，该桶才能被删除。另外，只有该桶的拥有者才能删除它，与桶的访问控制权限无关。

```
const params = {
    Bucket:'oss-test'
};
s3.deleteBucket(params, function(err, data){
   if (err) console.log(err, err.stack);
   else console.log('Bucket delete Successfully');
});
```

查看Bucket访问权限

```
var bucketParams = {
    Bucket: 'oss-test',
}
s3.getBucketAcl(bucketParams, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else if (data) {
    console.log("Success", data['Grants']);
  }
});
```

设置Bucket访问权限

```
var bucketParams1 = {
    Bucket: 'oss-test',
    ACL: 'private',
}
s3.putBucketAcl(bucketParams1, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

> ACL有如下几种"private","public-read","public-read-write","authenticated-read"

上传文件

对象存储有多种上传方式，不同的上传方式能够上传的数据大小也不一样。普通上传（PutObject）最多只能上传小于或等于5GB的文件；而分片上传（MultipartUpload）每个分片可以达到5GB，合并后的文件能够达到5TB。

首先介绍普通上传，我们会详细展示提供数据的各种方式，即方法中的 data 参数。其他上传接口有类似的data参数，不再赘述。

上传本地文件

把当前目录下的local.txt文件上传到对象存储服务器

```
const fs = require('fs');
const fileContent = fs.readFileSync("oss.js");
const params = {
    Bucket: 'oss-test',
    Key: 'oss.js', // File name you want to save as in S3
    Body: fileContent
};
// Uploading files to the bucket
s3.upload(params, function(err, data) {
    if (err) {
        console.log(err, err.stack);
    }
    else console.log('File uploaded successfully', data.Location);
});
```

断点续传

当需要上传的本地文件很大，或网络状况不够理想，往往会出现上传到中途就失败了。此时，如果对已经上传的数据重新上传，既浪费时间，又占用了网络资源。Node.js　SDK提供了分片上传接口，用于断点续传本地文件或者并发上传文件不同的区域块来加速上传。

分片上传

采用分片上传，用户可以对上传做更为精细的控制。这适用于诸如预先不知道文件大小、并发上传、自定义断点续传等场景。

```
let Uploader = require('s3-upload-streams');
const app = new (require('koa'))();
const AWS = require('aws-sdk');
const fs = require('fs');



let someFilePath = '/root/v0.4.3.0.tar';
AWS.config.update({
    "accessKeyId": "accesskey",
    "secretAccessKey": "scretkey",
    //"region": "us-east-1",
    "endpoint": "http://oss-cnbj01.cdsgss.com",
    "sslEnabled": false,
    "s3ForcePathStyle": true
    });
const BUCKET_NAME = 'oss-test';
const s3 = new AWS.S3();
s3.endpoint = new AWS.Endpoint("http://oss-cnbj01.cdsgss.com");
const partSize = 5 * 1024 * 1024;
let s3Uploader = new Uploader(s3, " oss-test ", partSize, 1000);
let stream = fs.createReadStream(someFilePath);
let uploadIdPromise = s3Uploader.startUpload({ Key: 'testbig' }, stream, { orginalPath: '/'});
```

```
    stream.on('end', () => {
        uploadIdPromise
        .then(uploadId => s3Uploader.completeUpload(uploadId))
        .then((metadata) => {
             console.log('Uploaded ${metadata.additionalMetadata.orginalPath} to ${metadata.location}');
     })
    .catch(err => console.log(err));
    });
```

下载文件

把对象存储服务器上的Object下载到本地文件`downloaded.js`：

```
    var s3Params = {
        Bucket: 'oss-test',
        Key: 'oss.js'
    };
    const filePath = './downloaded.js';
    s3.getObject(s3Params, function(err, res) {
        if (err) console.error(err);
        fs.writeFileSync(filePath, res.Body.toString());
        console.log(filePath,' has been created!');
    });
```

管理文件

通过Node.js SDK，用户可以罗列、删除、拷贝文件，也可以查看文件信息，更改文件元信息等。

罗列文件

Node.js SDK提供了一系列的迭代器，用于列举文件、分片上传等。

简单罗列

列举Bucket里的文件：

```
    var listparams = {
      Bucket: 'oss-test',
      Delimiter: 'test',
    }

    s3.listObjects(listparams, function (err, data) {
      if(err)throw err;
      console.log(data);
    });
```

按目录罗列

只列举目录为”test“的所有文件：

```
var listparams = {
  Bucket: 'oss-test',
  MaxKeys: 20,
  Delimiter: '/',
  Prefix: 'test/'
};
s3.listObjectsV2(listparams, function (err, data) {
  if(err)throw err;
  console.log('sssss', data);
});
```

判断文件是否存在

可以使用head接口来判断对象是否存在

```
var headParams = {
    Bucket: 'oss-test',
    Key: 'oss.js',
};
s3.headObject(headParams, function(err, res) {
    if (err) console.error(err);
    console.log('object has been found!');
});
```

删除文件

删除单个文件

```
var deleteParams = {
    Bucket: 'oss-test',
    Key: 'oss.js'
};
s3.deleteObject(deleteParams, function(err, res) {
    if (err) console.error(err);
    else console.log('object has been deleted!');
});
```

删除多个文件

批量删除oss-test下多个对象

```
var deleteParam = {
    Bucket: 'oss-test',
    Delete: {Objects: [
        {Key: 'oss1.js'},
        {Key: 'oss2.js'},
        {Key: 'oss3.js'}]
    }
};
s3.deleteObjects(deleteParam, function(err, data) {
    if (err) console.log(err, err.stack);
    else console.log('delete', data);
});
```

拷贝文件

把Bucket名为src-bucket下的source.txt拷贝到dst_bucket下且命名为的target.txt文件。

```
rams = {
  Bucket: "oss-test",
  CopySource: "oss-test/test/oss.js",
  Key: "oss.js"
};
s3.copyObject(rams, function(err, data) {
   if (err) console.log(err, err.stack); // an error occurred
   else     console.log(data);           // successful response
});
```

查看文件访问权限

查看文件的访问权限

```
rams = {
  Bucket: "oss-test",
  Key: "oss.js"
};
s3.getObjectAcl(rams, function(err, data) {
   if (err) console.log(err, err.stack); // an error occurred
   else     console.log(data.Grants);           // successful response
});
```

使用私有链接下载

对于私有Bucket，可生成私有链接（又称为“签名URL”），下面是生成私有链接下载，该链接在3600 秒后失效

```
const myBucket = 'oss-test'
const myKey = 'oss.js'
const signedUrlExpireSeconds = 60*60

const url = s3.getSignedUrl('getObject', {
    Bucket: myBucket,
    Key: myKey,
    Expires: signedUrlExpireSeconds
})
console.log(url)
```