# 18 Uploading Fliles to Amazon S3 (Simple Storage Service)

## Objectives

- Students will create an HTML form to upload files
- Students will configure a server to accept file uploads
- Students will store uploaded files using Amazon's S3 Simple Storage Service

# Sending an HTML File

Create some basic project infrastructure. Edit the `index.html` file yourself to make a simple homepage.

```
npminit -y
mkdir static
touch index.js
touch index.html
```

Configure the server to return the HTML page.

```
const express = require('express');
const app = express();
app.use(express.static('static'));

app.get('/', (req, res) => {
  res.sendFile('index.html');
});
```

# Build an HTML Form for File Upload

```
<form action="/photos/upload" method="POST" enctype="multipart/form-data">
  <input type="file" name="picture" />
  <input type="submit" />
</form>
```

- `action` - defines what route on the server the form will submit to
- `method` - defines what HTTP method the request will use
- `enctype` - configures the form to upload all it's information in "multipart" format which will allow files to be sent.
- Use `type="file"` to configure an input to trigger a file-select window.

# File Upload Route

- multer Documentation

This route uses `multer` middleware. It's configured to look for form data named `picture`. It will gather the file data and save it to the destinationed folder configured as `uploads/`.

```
const express = require('express');
const router = new express.Router();

const multer  = require('multer');
const upload = multer({ dest: 'uploads/' });

router.post('/upload', upload.single('picture'), function (req, res, next) {
  // req.file is the `avatar` file
  // req.body will hold the text fields, if there were any
  res.send(req.file);
});

module.exports = router;
```

# AWS .env Variables

```
AWS_BUCKET='your bucket name'
AWS_ACCESS_KEY_ID='your aws key'
AWS_SECRET_ACCESS_KEY='your secret aws key'
```

- Navigate to AWS and sign in
- Click on your username in the uppser right and select My Security Credentials
- Expand the `Access Keys` section
- Click `Create New Access Key`
- Toggle `Show Access Key`
- Copy `Access Key ID:` to your .env
- Copy `Secret Access Key:` to your .env
- Fill in the name of whatever bucket you're using

**Note:** When you navigate to My Security Credentials you may see a message encouraging you to "Get Started with IAM Users." Simply ignore it and choose "Continue to Security Credentials."

# Configuring AWS

Make sure you call `require('dotenv').config()` in your main server file so everything in your `.env` file gets attached to `process.env`.

```
const AWS = require('aws-sdk');
const s3 = new AWS.S3();

const path = require('path');
const multer  = require('multer');
const upload = multer({ dest: 'uploads/' });

router.post('/upload', upload.single('picture'), function (req, res, next) {
  let ext = path.extname(req.file.originalname);
  let params = {
    ACL: 'public-read',
    Bucket: process.env.AWS_BUCKET,
    Key: `${req.file.filename}${ext}`,
    Body: fs.createReadStream(req.file.path)
  }

  s3.upload(params, (err, s3Data) => {
    res.send(s3Data);
  });
});
```

# Saving S3 Data to the database

Save S3 responses as documents in MongoDB to keep track of everything we've uploaded.

```
s3.upload(params, (err, s3Data) => {
  new Photo({
    url: s3Data.Location
  })
  .save()
  .then(photo => {
    res.send(photo);
  });
});
```

# Testing File Uploads

`superagent` has a method called `.attach()` that allows us to attach a file to a server request.

```
require('dotenv').config();
const request = require('superagent');

describe('S3 Uploads', () => {
  it('should be able to upload images', (done) => {
    let imageLocation = './data/photo1.png';
    let uploadUrl = 'http://localhost:3000/photos/upload';

    request.post(uploadUrl)
    .attach('picture', imageLocation)
    .end((err, res) => {
      let amazonUrl = process.env.AWS_BUCKET + '.s3.amazonaws.com';
      let isAmazonUrl = res.body.url.includes(amazonUrl);
      expect(isAmazonUrl).toBe(true);

      done();
    });
  });
});
```

# Common superagent Mistake

Make sure you give the `.attach()` method the proper name of the form field where the server expects to see the image data. Here my server expects forms to have the file input be named `picture`:

**index.html**

```
<input type="file" name="picture" />
```

**index.js**

```
router.post('/upload', upload.single('picture'), function (req, res, next) {

});
```

**upload.test.js**

```
request.post(uploadUrl)
.attach('picture', imageLocation)
```

# Check for Understanding

Why do we save S3 responses to the database?

# Check for Understanding

## Why do we save S3 responses to the database?

Storing the responses in our database allows us to query for everything we've uploaded, and associate file uploads with any other information we want to save.

# Check for Understanding

How can we associate file uploads with users?

# Check for Understanding

## How can we associate file uploads with users?

Build database relationships between a `User` model and your uploaded resource model just like you did in Lab 17!

Use Auth middlewear to have users signup, log in, and to be able to identify them on every request.

Add a `userId` property to the uploaded resources you're saving!

# Check for Understanding

## What can we upload to S3?

# Check for Understanding

## What can we upload to S3?

Pretty much anything!