Localstack practices

Localstack reference:

https://localstack.cloud

Benefits:

- You can work offline
- You don't need a shared 'dev' bucket that everyone on your team uses
- You can easily wipe & replace your local buckets
- You don't need to worry about paying for AWS usage
- You don't need to log into AWS 😛

Stack:

- Nodejs
- Express
- brew if is necesary
- Localstack (AWS suit)
- Docker & Docker Compose
- Jest & SuperTest for testing
- Swagger for documentation

First Goal (Fake AWS locally with LocalStack)

Reference

Localstack within a node app

Localstack allows you to emulate a number of AWS services on your computer. We are going to use a S3 iin this example.

Initial setup

1. Install docker if you haven't already.
2. Install **AWS CLI** oficial or using brew, even though we aren't going to be working with "real" AWS, we'll use this to talk to our local docker containers.
3. Once the AWS CLI is installed, run `aws configure` to create some credentials. Even though we're talking to our "fake" local service, we still need credentials. You can enter real credentials (as described here), or dummy ones. Localstack requires that these details are present, but doesn't actually validate them.
4. Create a new directory for you project, and within it a new nodejs project `npm init -y`.
5. Make a few files inside you project:

- aws.js
- docker-compose.yml
- test-upload.js

- .env

6. Add an image to your project directory and rename it to `test-upload.jpg`.
7. Install aws-sdk, dotenv `npm install --save aws-sdk dotenv`

Docker config

Run Localstack with docker-compose to set this up.

`docker-compose.yml`

```
version: "3.2"
services:
  localstack:
    image: localstack/localstack:0.11.3
    container_name: "fake-aws-s3"
    network_mode: bridge
    ports:
      - "4566:4566"
      - "8055:8080"
    environment:
      - SERVICES=s3
      - DEBUG=1
      - DATA_DIR=/tmp/localstack/data
    volumes:
      - "./.localstack:/tmp/localstack"
      - "/var/run/docker.sock:/var/run/docker.sock"
```

The line `4566:4566` does the same thing, but binds a whole range of ports. These particular port numbers are what Localstack uses as endpoints for the various APIs. We'll see more about this in a little bit.

`SERVICES=s3`: You can define a list of AWS services to emulate. In our case, we're just using S3, but you can include additional APIs, i.e.

`SERVICES=s3,lambda`: There's more on this in the Localstack docs.

`DEBUG=1`: 📜 Show me all of the logs!

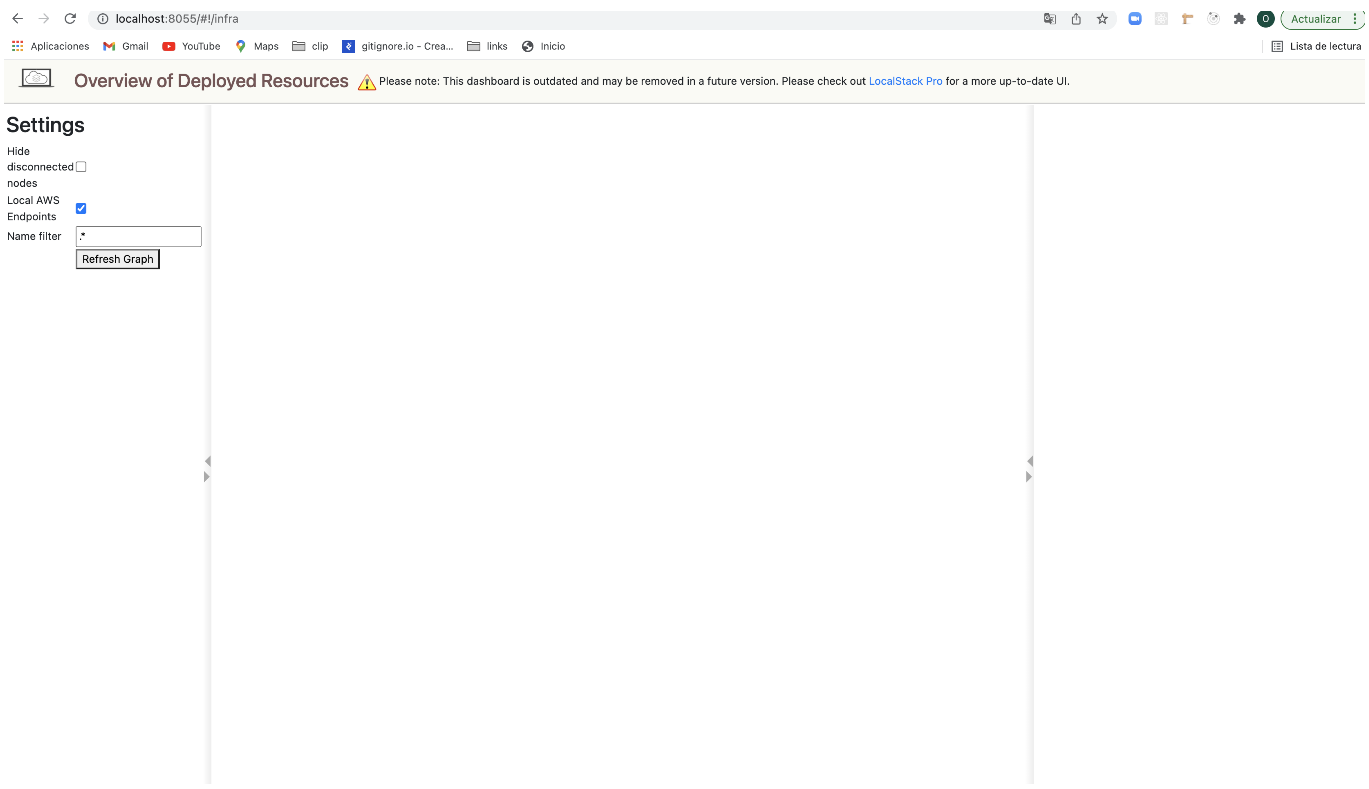
`DATA_DIR=/tmp/localstack/data`: This is the directory where Localstack will save its data internally.

Starting our container

Run in terminal:

`docker-compose up`

To make sure it's working, we can visit http://localhost:8055 to see localstack's web UI:



Overview of Deployed Resources ⚠ Please note: This dashboard is outdated and may be removed in a future version. Please check out LocalStack Pro for a more up-to-date UI.

Settings

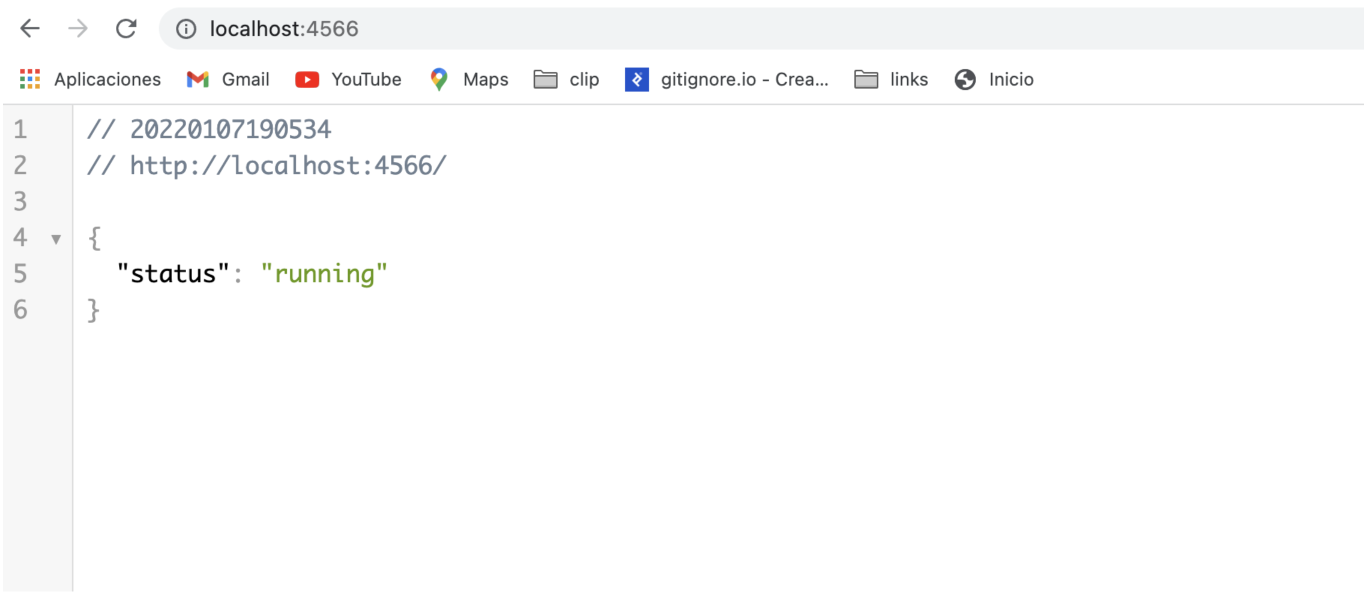
Hide disconnected nodes ☐

Local AWS Endpoints ☑

Name filter .*

Refresh Graph

Similary, our s3 endpoint http://localhost:4566:



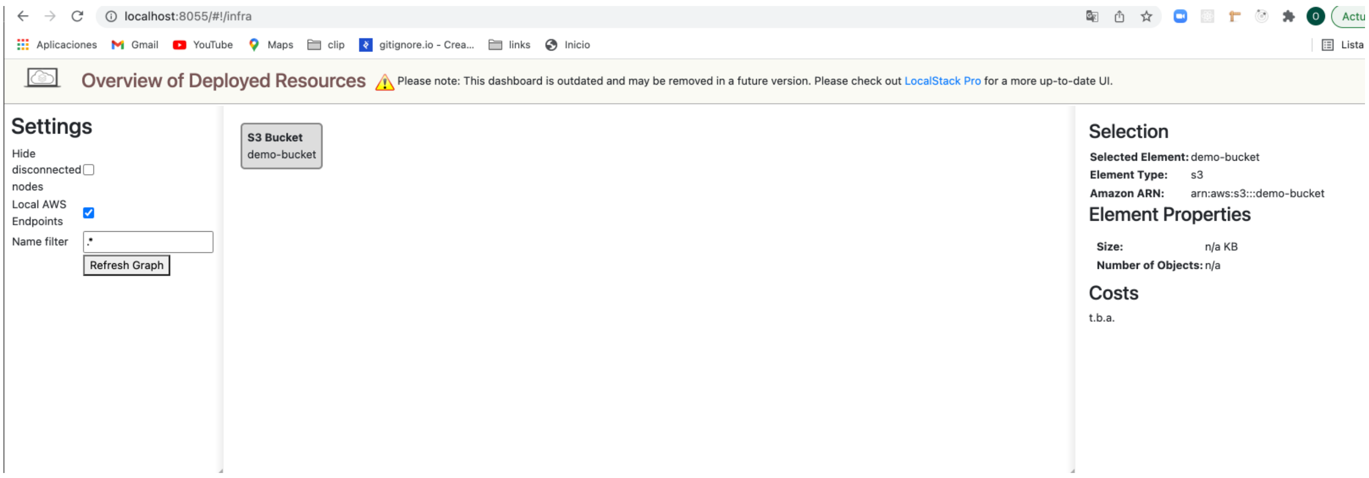
```
// 20220107190534
// http://localhost:4566/

{
  "status": "running"
}
```

Working with Localstack

AWS is now inside our computer. After we start upload files, we need to create and configure a **bucket**. We will do this using rhw AWS CLI that we installed earlier, using the `-- endpoint-url` flag to talk to Localstack instead.

1. Create a bucket: `aws --endpoint-url=http://localhost:4566 s3 mb s3://demo-bucket`
2. Attach an ACL (Access control list) to the bucket so it's readable: `aws --endpoint-url=http://localhost:4566 s3api put-bucket-acl --bucket demo-bucket --acl public-read`

Now, when we visit the web UI, we will see our bucket:



Overview of Deployed Resources ⚠ Please note: This dashboard is outdated and may be removed in a future version. Please check out LocalStack Pro for a more up-to-date UI.

Settings

Hide disconnected nodes ☐
Local AWS Endpoints ☑
Name filter .*
Refresh Graph

S3 Bucket
demo-bucket

Selection

Selected Element: demo-bucket
Element Type: s3
Amazon ARN: arn:aws:s3:::demo-bucket

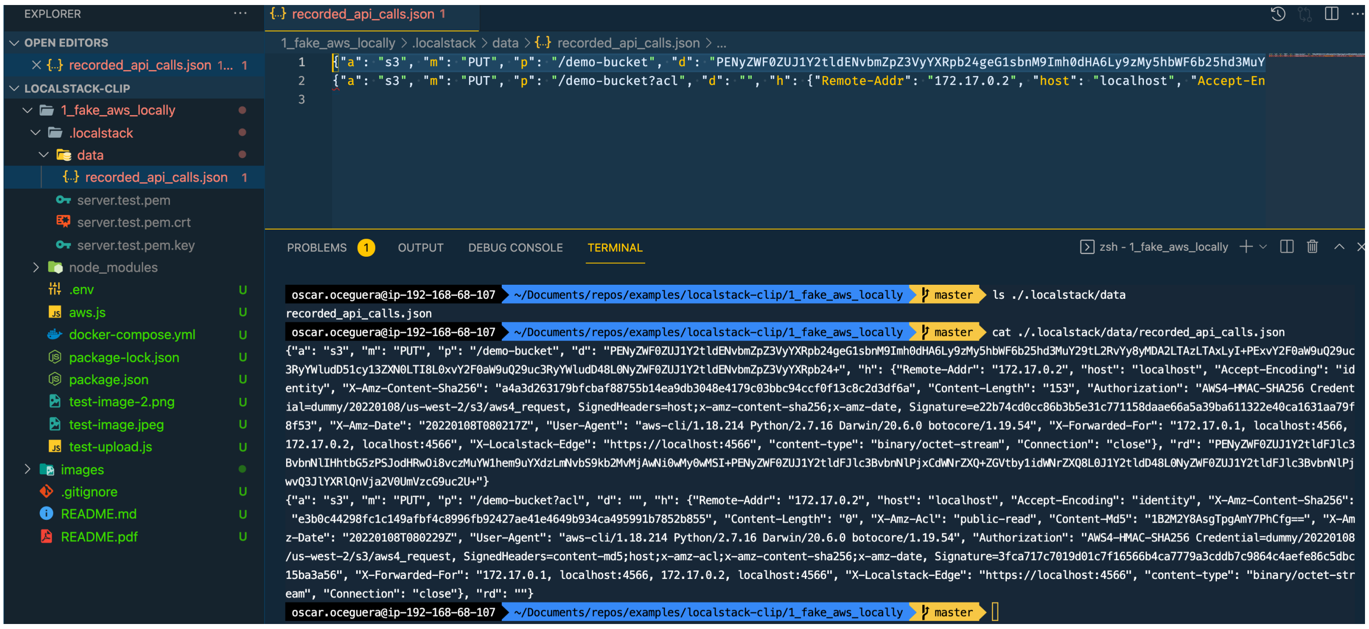
Element Properties

Size: n/a KB
Number of Objects: n/a

Costs

t.b.a.

If you use `volumes` in your docker settings, let's paause for a moment to look at what's going on in `./.localstack/data`:



```
{"a": "s3", "m": "PUT", "p": "/demo-bucket", "d": "PENyZWF0ZUJ1Y2tldENvbmZpZ3VyYXRpb24geG1sbnM9Imh0dHA6Ly9zMy5hbWF6b25hd3MuY
{"a": "s3", "m": "PUT", "p": "/demo-bucket?acl", "d": "", "h": {"Remote-Addr": "172.17.0.2", "host": "localhost", "Accept-En
```

```
oscar.oceguera@ip-192-168-68-107 ~/Documents/repos/examples/localstack-clip/1_fake_aws_locally master ls ./.localstack/data
recorded_api_calls.json
oscar.oceguera@ip-192-168-68-107 ~/Documents/repos/examples/localstack-clip/1_fake_aws_locally master cat ./.localstack/data/recorded_api_calls.json
{"a": "s3", "m": "PUT", "p": "/demo-bucket", "d": "PENyZWF0ZUJ1Y2tldENvbmZpZ3VyYXRpb24geG1sbnM9Imh0dHA6Ly9zMy5hbWF6b25hd3MuY29tL2RvYy8yMDA2LTAzLTAxLyI+PExvY2F0aW9uQ29uc
3RyYWludD51cy13ZXN0LTI8L0xvY2F0aW9uQ29uc3RyYWludD48L0NyZWF0ZUJ1Y2tldENvbmZpZ3VyYXRpb24+", "h": {"Remote-Addr": "172.17.0.2", "host": "localhost", "Accept-Encoding": "id
entity", "X-Amz-Content-Sha256": "a4a3d263179bfcbaf88755b14ea9db3048e4179c03bbc94ccf0f13c8c2d3df6a", "Content-Length": "153", "Authorization": "AWS4-HMAC-SHA256 Credent
ial=dummy/20220108/us-west-2/s3/aws4_request, SignedHeaders=host;x-amz-content-sha256;x-amz-date, Signature=e22b74cd0cc86b3b5e31c771158daae66a5a39ba611322e40ca1631aa79f
8f53", "X-Amz-Date": "20220108T080217Z", "User-Agent": "aws-cli/1.18.214 Python/2.7.16 Darwin/20.6.0 botocore/1.19.54", "X-Forwarded-For": "172.17.0.1, localhost:4566,
172.17.0.2, localhost:4566", "X-Localstack-Edge": "https://localhost:4566", "content-type": "binary/octet-stream", "Connection": "close"}, "rd": "PENyZWF0ZUJ1Y2tldFJlc3
BvbnNlIHhtbG5zPSJodHRwOi8vczMuYW1hem9uYXdzLmNvbS9kb2MvMjAwNi0wMy0wMSI+PENyZWF0ZUJ1Y2tldFJlc3BvbnNlPjxCdWNrZXQ+ZGVtby1idWNrZXQ8L0J1Y2tldD48L0NyZWF0ZUJ1Y2tldFJlc3BvbnNlPj
wvQ3JlYXRlQnVja2V0UmVzcG9uc2U+"}
{"a": "s3", "m": "PUT", "p": "/demo-bucket?acl", "d": "", "h": {"Remote-Addr": "172.17.0.2", "host": "localhost", "Accept-Encoding": "identity", "X-Amz-Content-Sha256":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855", "Content-Length": "0", "X-Amz-Acl": "public-read", "Content-Md5": "1B2M2Y8AsgTpgAmY7PhCfg==", "X-Am
z-Date": "20220108T080229Z", "User-Agent": "aws-cli/1.18.214 Python/2.7.16 Darwin/20.6.0 botocore/1.19.54", "Authorization": "AWS4-HMAC-SHA256 Credential=dummy/20220108
/us-west-2/s3/aws4_request, SignedHeaders=content-md5;host;x-amz-acl;x-amz-content-sha256;x-amz-date, Signature=3fca717c7019d01c7f16566b4ca7779a3cddb7c9864c4aefe86c5dbc
15ba3a56", "X-Forwarded-For": "172.17.0.1, localhost:4566, 172.17.0.2, localhost:4566", "X-Localstack-Edge": "https://localhost:4566", "content-type": "binary/octet-str
eam", "Connection": "close"}, "rd": ""}
oscar.oceguera@ip-192-168-68-107 ~/Documents/repos/examples/localstack-clip/1_fake_aws_locally master
```

Here, we can see that Localstack is recording all API calls in this JSON file. When the container restarts, it will re-apply these calls - this is how we are able to keep our data between restarts. Once we start uploading, we won't see new files appear in this directory. Instead, our uploads will be recorded in this file as raw data. (You could include this file in your repo if you wanted to share the state of the container with others - but depending on how much you upload, it's going to become a pretty big file)

Uploading from our app

We will just make a simple **upload** function and try uploading an image a few times to our S3 butcket.

`.env`, our enviroment variables

```
AWS_ACCESS_KEY_ID='123'
AWS_SECRET_KEY='xyz'
AWS_BUCKET_NAME='demo-bucket'
```

Note: it doesn't matter what your AWS key & secret are, as long as they aren't empty.

`aws.js`, the module for our upload function

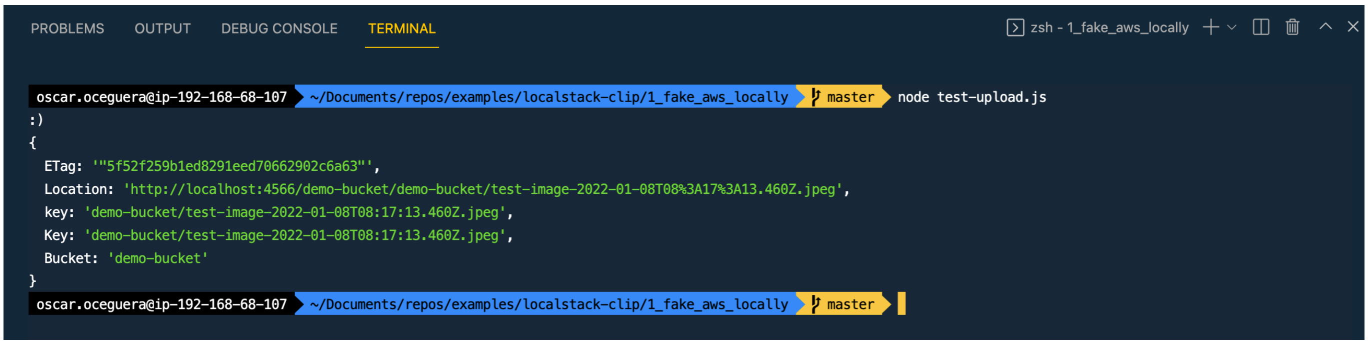
```
const AWS = require("aws-sdk");
require("dotenv").config();

const credentials = {
  accessKeyId: process.env.AWS_ACCESS_KEY_ID,
  secretAccessKey: process.env.AWS_SECRET_KEY,
};

const useLocal = process.env.NODE_ENV !== "production";

const bucketName = process.env.AWS_BUCKET_NAME;

const s3client = new AWS.S3({
  credentials,
  endpoint: useLocal ? "http://localhost:4566" : undefined,
  s3ForcePathStyle: true,
});

const uploadFile = async (data, name) =>
  new Promise((resolve) => {
    s3client.upload(
      {
        Bucket: bucketName,
        Key: `${bucketName}/${name}`,
        Body: data,
      },
      (err, response) => {
        if (err) throw err;
        resolve(response);
      }
    );
  });

module.exports = uploadFile;
```

`test-upload.js`, which implements th eupload function

```
const fs = require("fs");
const path = require("path");
const uploadFile = require("./aws");

const testUpload = () => {
  const filePath = path.resolve(__dirname, "test-image.jpg");
  const fileStream = fs.createReadStream(filePath);
  const now = new Date();
  const fileName = `test-image-${now.toISOString()}.jpg`;
```

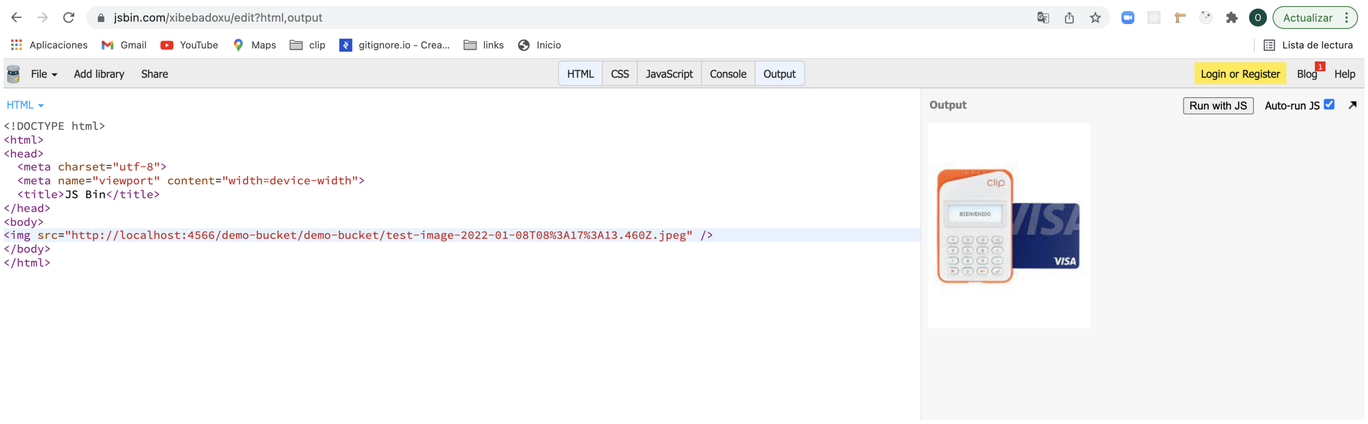
```
    uploadFile(fileStream, fileName)
      .then((response) => {
        console.log(":)");
        console.log(response);
      })
      .catch((err) => {
        console.log(":|");
        console.log(err);
      });
  };

  testUpload();
```

Run `node test-upload.js` in your terminal



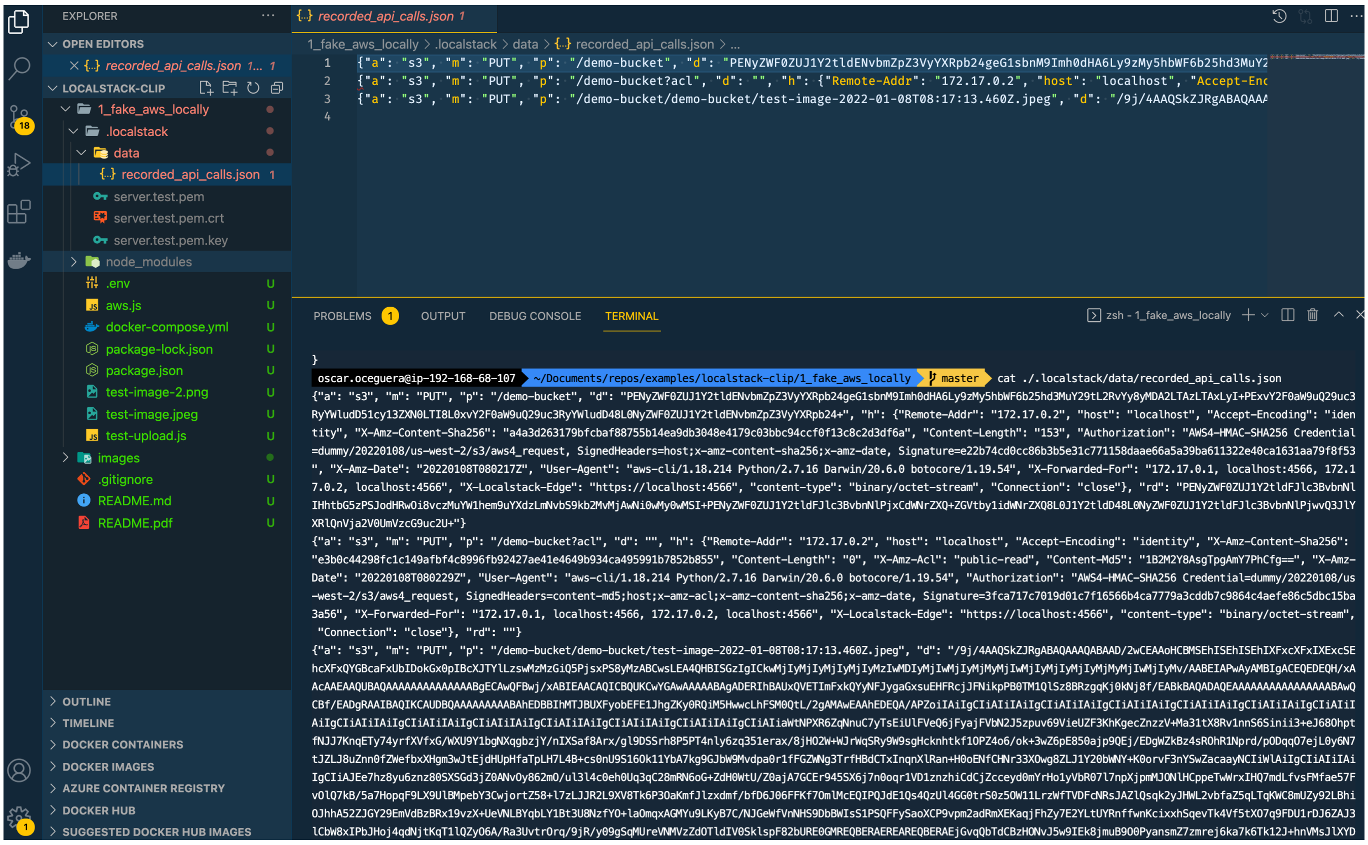
```
oscar.ocegura@ip-192-168-68-107 ~/Documents/repos/examples/localstack-clip/1_fake_aws_locally  master  node test-upload.js
:)
{
  ETag: '"5f52f259b1ed8291eed70662902c6a63"',
  Location: 'http://localhost:4566/demo-bucket/demo-bucket/test-image-2022-01-08T08%3A17%3A13.460Z.jpeg',
  key: 'demo-bucket/test-image-2022-01-08T08:17:13.460Z.jpeg',
  Key: 'demo-bucket/test-image-2022-01-08T08:17:13.460Z.jpeg',
  Bucket: 'demo-bucket'
}
oscar.ocegura@ip-192-168-68-107 ~/Documents/repos/examples/localstack-clip/1_fake_aws_locally  master 
```

Copy the URL in the Location property of the response and paste it into your browser. The browser will immediately download the image. If you want to see it in your browser, you can use something like JS Bin:



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>JS Bin</title>
</head>
<body>
<img src="http://localhost:4566/demo-bucket/demo-bucket/test-image-2022-01-08T08%3A17%3A13.460Z.jpeg" />
</body>
</html>
```

Then, if you look at .localstack/data/recorded_api_calls.json again, you'll see it filled up with the binary data of the image:



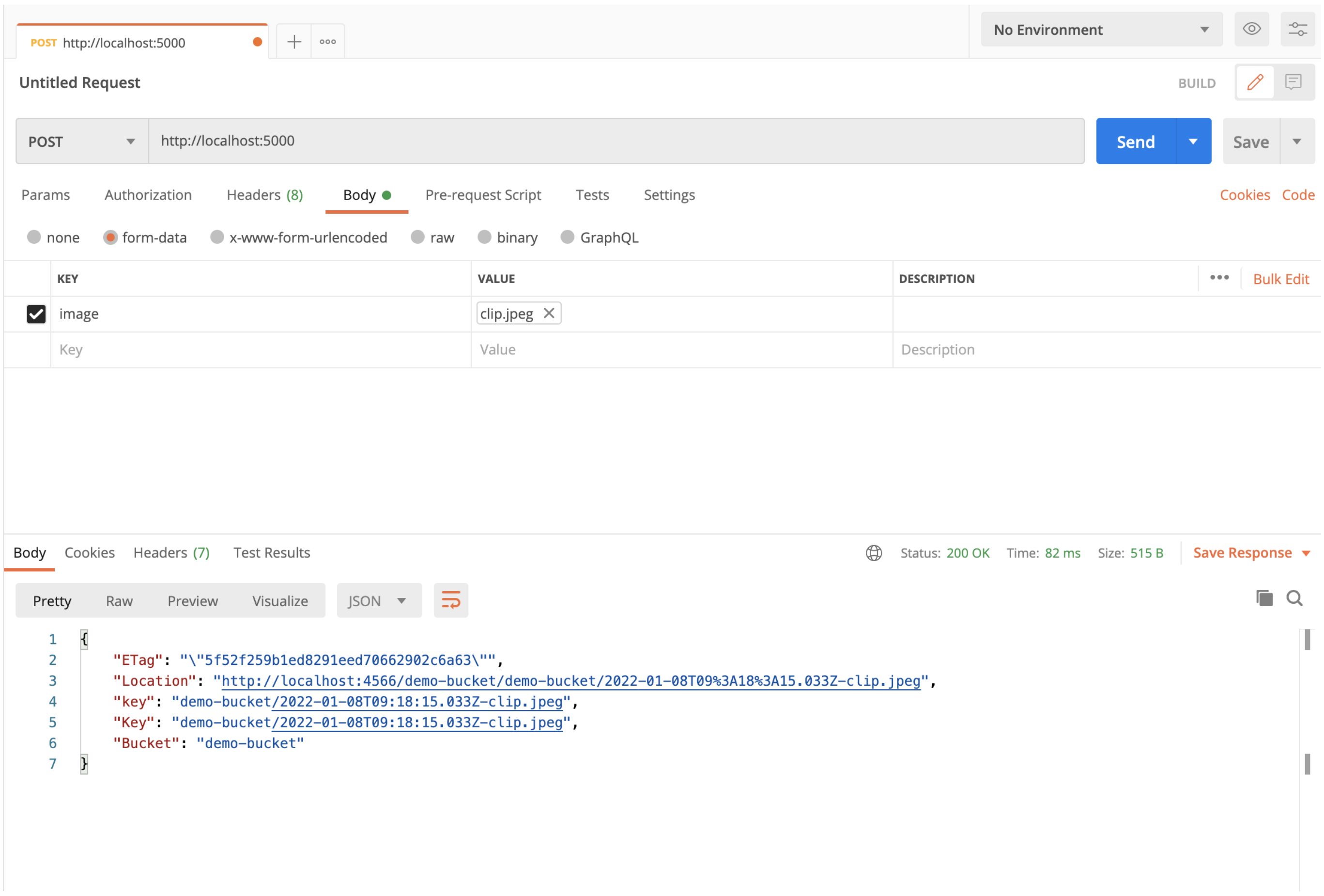
```
{"a": "s3", "m": "PUT", "p": "/demo-bucket", "d": "PENyZWF0ZUJ1Y2tldENvbmZpZ3VyYXRpb24geG1sbnM9Imh0dHA6Ly9zMy5hbWF6b25hd3MuY2
{"a": "s3", "m": "PUT", "p": "/demo-bucket?acl", "d": "", "h": {"Remote-Addr": "172.17.0.2", "host": "localhost", "Accept-Enc
{"a": "s3", "m": "PUT", "p": "/demo-bucket/demo-bucket/test-image-2022-01-08T08:17:13.460Z.jpeg", "d": "/9j/4AAQSkZJRgABAQAAA
```

```
}
oscar.oceguera@ip-192-168-68-107 ~/Documents/repos/examples/localstack-clip/1_fake_aws_locally  master  cat ./.localstack/data/recorded_api_calls.json
{"a": "s3", "m": "PUT", "p": "/demo-bucket", "d": "PENyZWF0ZUJ1Y2tldENvbmZpZ3VyYXRpb24geG1sbnM9Imh0dHA6Ly9zMy5hbWF6b25hd3MuY29tL2RvYy8yMDA2LTAzLTAxLyI+PExvY2F0aW9uQ29uc3
RyYWludD51cy13ZXN0LTI8L0xvY2F0aW9uQ29uc3RyYWludD48L0NyZWF0ZUJ1Y2tldENvbmZpZ3VyYXRpb24+", "h": {"Remote-Addr": "172.17.0.2", "host": "localhost", "Accept-Encoding": "iden
tity", "X-Amz-Content-Sha256": "a4a3d263179bfcbaf88755b14ea9db3048e4179c03bbc94ccf0f13c8c2d3df6a", "Content-Length": "153", "Authorization": "AWS4-HMAC-SHA256 Credential
=dummy/20220108/us-west-2/s3/aws4_request, SignedHeaders=host;x-amz-content-sha256;x-amz-date, Signature=e22b74cd0cc86b3b5e31c771158daae66a5a39ba611322e40ca1631aa79f8f53
", "X-Amz-Date": "20220108T080217Z", "User-Agent": "aws-cli/1.18.214 Python/2.7.16 Darwin/20.6.0 botocore/1.19.54", "X-Forwarded-For": "172.17.0.1, localhost:4566, 172.1
7.0.2, localhost:4566", "X-Localstack-Edge": "https://localhost:4566", "content-type": "binary/octet-stream", "Connection": "close"}, "rd": "PENyZWF0ZUJ1Y2tldFJlc3BvbnNl
IHhtbG5zPSJodHRwOi8vczMuYW1hem9uYXdzLmNvbS9kb2MvMjAwNi0wMy0wMSI+PENyZWF0ZUJ1Y2tldFJlc3BvbnNlPjxCdWNrZXQ+ZGVtby1idWNrZXQ8L0J1Y2tldD48L0NyZWF0ZUJ1Y2tldFJlc3BvbnNlPjwvQ3JlY
XRlQnVja2V0UmVzcG9uc2U+"}
{"a": "s3", "m": "PUT", "p": "/demo-bucket?acl", "d": "", "h": {"Remote-Addr": "172.17.0.2", "host": "localhost", "Accept-Encoding": "identity", "X-Amz-Content-Sha256":
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855", "Content-Length": "0", "X-Amz-Acl": "public-read", "Content-Md5": "1B2M2Y8AsgTpgAmY7PhCfg==", "X-Amz-
Date": "20220108T080229Z", "User-Agent": "aws-cli/1.18.214 Python/2.7.16 Darwin/20.6.0 botocore/1.19.54", "Authorization": "AWS4-HMAC-SHA256 Credential=dummy/20220108/us
-west-2/s3/aws4_request, SignedHeaders=content-md5;host;x-amz-acl;x-amz-content-sha256;x-amz-date, Signature=3fca717c7019d01c7f16566b4ca7779a3cddb7c9864c4aefe86c5dbc15ba
3a56", "X-Forwarded-For": "172.17.0.1, localhost:4566, 172.17.0.2, localhost:4566", "X-Localstack-Edge": "https://localhost:4566", "content-type": "binary/octet-stream",
 "Connection": "close"}, "rd": ""}
{"a": "s3", "m": "PUT", "p": "/demo-bucket/demo-bucket/test-image-2022-01-08T08:17:13.460Z.jpeg", "d": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAoHCBMSEhISEhISEhIXFxcXFxIXExcSE
hcXFxQYGBcaFxUbIDokGx0pIBcXJTYlLzswMzMzGiQ5PjsxPS8yMzABCwsLEA4QHBISGzIgICkwMjIyMjIyMjIyMjIyMzIwMDIyMjIwMjIyMjMyMjIwMjIyMjIyMjIyMjMyMjIwMjIyMv/AABEIAPwAyAMBIgACEQEDEQH/xA
AcAAEAAQUBAQAAAAAAAAAAAAAABgECAwQFBwj/xABIEAACAQICBQUKCwYGAwAAAAABAgADERIhBAUxQVETImFxkQYyNFJygaGxsuEHFRcjJFNikpPB0TM1QlSz8BRzgqKj0kNj8f/EABkBAQADAQEAAAAAAAAAAAAAAAABAwQ
CBf/EADgRAAIBAQIKCAUDBQAAAAAAAAAAABAhEDBBIhMTJBUXFyobEFE1JhgZKy0RQiM5HwwcLhFSM0QtL/2gAMAwEAAhEDEQA/APZoiIAiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiI
AiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiIAiIgCIiAIiaWtNPXR6ZqNnuC7yTsEiUlFVeQ6jFyajFVbN2J5zpuv69VieUZF3KhKgecZnzzV+Ma31tX8Rv1nnS6S6sXn6hbVHiErtnhtkcf10PZ4o6
fNJJ7KnqETy74yrfXVfxG/WXU9Y1bgNXqgbzjY/nIXSaf8rd/l/ok+3wF6oy6zq351erax/8jH02W+WJrWqSRy9W9sgHcknhtkf10PZ4o6/ok+3wZ6pE850ajp9QEj/EDgWZkBz4sROhR1Nprd/pODqq07ejL0y6N7
tJZLJ8uZnn0fZWefbxXHgm3wJtEjdHUpHfaTpLH7L4B+cs0nU9S16Ok11YbA7kg9GJbW9Mvdpa0r1fFGZWNg3TrfHBdCTxInqnXlRan+H0oENfCHNr33XOwg8ZLJ1Y20bWNY+K0orvF3nYSwZacayNCIiWlAiIgCIiAIIiAi
IgCIiAJEe7hz8yu6znz80SXSGd3jZ0ANvOy862m0/ul3l4c0eh0Uq3qC28mRN6j7l7npXjpmMJONlHCppeTwWrxIHQ7mdLfvsFMfae57F
vOlQ7kB/5a7HopqF9LX9UlBMpebY3CwjortZ58+l7zLJJR2L9XV8Tk6P30aKmfJlzvbfaZ5qLTqKWC8mUZ92LBhi
OJhhA52ZJGY29EmVdBzBRx19vzX+UeVNLBYqbLY1Bt3U8+laOmqxAGMYu9LKyB7C/NJGeWfVnNHS9DbBWIsS1PSQFFySaoXCP9vpm2adRmXEKaqjFhZy7E2YLtUkaq2fwnKcixhSqevTk4Vf5tX07O9s6XRz6ZZLt9YzrGKSNAIJ
lCbW8xIPbJHoj4qdwSPXlsxsXszdbn71Oy4Bj7zrSTdc+vRjpM2lr+sWcAz7H4/hn9i8ZnzOjBcnGfnn2P8AEOjzR1XhZpJE1r2wklVnclMhI/IhBb81o/Knh7i9LXaUH8NkZ/O9xU2j5OcXAO6kijdOrU8mP9N/8Ir8/AEzLqF
```

EXERCISE

Add a Endpoint to upload files (images) and other that list all images, install:

- express
- express-fileupload



POST http://localhost:5000

KEY	VALUE	DESCRIPTION
✓ image	clip.jpeg	
Key	Value	Description

Body — Headers (7) — Status: 200 OK Time: 82 ms Size: 515 B

```
{
    "ETag": "\"5f52f259b1ed8291eed70662902c6a63\"",
    "Location": "http://localhost:4566/demo-bucket/demo-bucket/2022-01-08T09%3A18%3A15.033Z-clip.jpeg",
    "key": "demo-bucket/2022-01-08T09:18:15.033Z-clip.jpeg",
    "Key": "demo-bucket/2022-01-08T09:18:15.033Z-clip.jpeg",
    "Bucket": "demo-bucket"
}
```

Second Goal (Localstack/Nodejs/DynamoDB/Swagger/Jest)

- Create a new directory for you project
- Within it a new nodejs project `npm init -y`.
- install dependecies with npm:
  - `npm install --save aws-sdk body-parser cors express swagger-ui-express xid-js`
  - `npm install --save-dev jest nodemon supertest`
- Create new files:
  - src/index.js
  - src/app.js
  - src/router.js
  - src/tests/integration.test.js
  - src/swagger.json
  - Dockerfile
  - docker-compose.yml
  - jest.config.js

Setup routes `src/routes`

```
const express = require("express");
const router = express.Router();

module.exports = router;
```

Begin a simple express server `src/app.js`

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");

const router = require("./router");
const swaggerUI = require("swagger-ui-express");
const swaggerDocument = require("./swagger.json");
const app = express();

app.use(cors());

app.get("/", (req, res) => {
  const msg = {
    message: "Welcome to API!",
  };

  res.status(200).send(msg);
});

app.get("/status", (req, res) => {
  const status = {
    status: "ok",
    api: "soccer-player-api",
    time: Date.now(),
  };
  res.status(200).send(status);
});

router.use("/docs", swaggerUI.serve, swaggerUI.setup(swaggerDocument));

app.use(bodyParser.json());
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);

app.use((req, res, next) => {
  res.removeHeader("X-Powered-By");
  next();
});

app.use("/", router);

module.exports = app;
```

Write document `src/swagger.json`

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Soccer Players API",
    "description": "Tiny RESTful API using Node.js: allows you to manage
soccer players.",
    "contact": {
      "name": "Mauro Bonfietti",
      "url": "https://github.com/maurobonfietti"
    },
    "version": "1.0"
  },
  "paths": {
    "/": {
      "get": {
        "tags": ["Info"],
        "summary": "Get help",
        "description": "Get help about this API.",
        "operationId": "Gethelp",
        "parameters": [],
        "responses": {
          "200": {
            "description": "",
            "headers": {
              "X-Powered-By": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "Express"
                  }
                }
              },
              "Access-Control-Allow-Origin": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "*"
                  }
                }
              },
              "Content-Length": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "43"
                  }
                }
              },
              "ETag": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "W/\"2b-h6BMmxGhWkCo5dA5bJgq/pDo3sQ\""
                  }
                }
              },
              "Date": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "Mon, 23 Dec 2019 21:03:20 GMT"
                  }
                }
              },
              "Connection": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "keep-alive"
                  }
                }
              }
            },
            "content": {
              "application/json; charset=utf-8": {
                "schema": {
                  "$ref": "#/components/schemas/Gethelp"
                },
                "example": {
                  "message": "Welcome to Soccer Player API!"
                }
              }
            }
          }
        },
        "deprecated": false
      }
    },
    "/status": {
      "get": {
        "tags": ["Info"],
        "summary": "Health check",
        "description": "Get status of this API.",
        "operationId": "Healthcheck",
        "parameters": [],
        "responses": {
          "200": {
            "description": "",
            "headers": {
              "X-Powered-By": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "Express"
                  }
                }
              },
              "Access-Control-Allow-Origin": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "*"
                  }
                }
              },
              "Content-Length": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "62"
                  }
                }
              },
              "ETag": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "W/\"3e-r8UILbmoe9Rk/g5iPSk00uuUUuk\""
                  }
                }
              },
              "Date": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "Mon, 23 Dec 2019 21:03:27 GMT"
                  }
                }
              },
              "Connection": {
                "content": {
                  "text/plain": {
                    "schema": {
                      "type": "string"
                    },
                    "example": "keep-alive"
                  }
                }
              }
            },
            "content": {
              "application/json; charset=utf-8": {
                "schema": {
                  "$ref": "#/components/schemas/Healthcheck"
                },
                "example": {
                  "status": "ok",
                  "api": "soccer-player-api",
                  "time": 1577135007891
                }
              }
            }
          }
        },
        "deprecated": false
      }
    }
  },
  "components": {
    "schemas": {
      "Gethelp": {
        "title": "Gethelp",
        "required": ["message"],
        "type": "object",
        "properties": {
          "message": {
            "type": "string"
          }
        },
        "example": {
          "message": "Welcome to Soccer Player API!"
        }
      },
      "Healthcheck": {
        "title": "Healthcheck",
        "required": ["status", "api", "time"],
        "type": "object",
        "properties": {
          "status": {
            "type": "string"
          },
          "api": {
            "type": "string"
          },
          "time": {
            "type": "integer",
            "format": "int64"
          }
        },
        "example": {
          "status": "ok",
          "api": "soccer-player-api",
          "time": 1577135007891
        }
      }
    }
  },
  "tags": [
    {
      "name": "Info",
      "description": "Endpoints to get info about the API."
    },
    {
      "name": "Players",
      "description": "Endpoints to get and manage soccer players."
    }
  ]
}
```

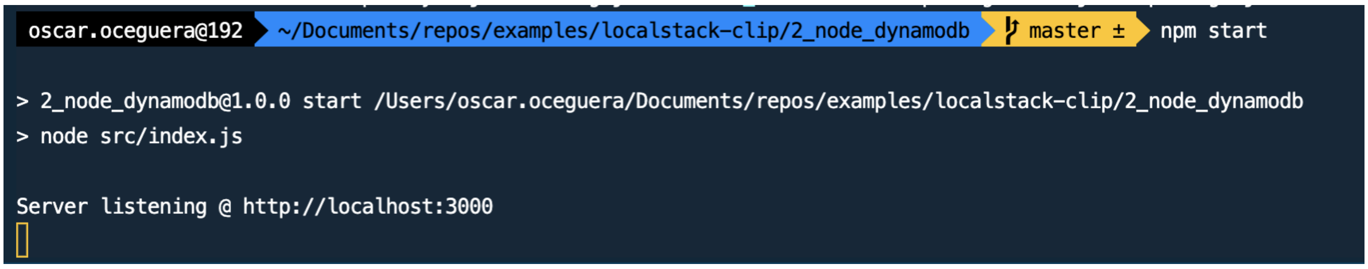
Listening the server `src/index.js`

```
const app = require("./app");
const PORT = 3000;

app.use((req, res) => {
  res.status(404);
  res.send({ error: "Not Found -  Error 404." });
});

app.listen(PORT, () =>
  console.log(`Server listening @ http://localhost:${PORT}`)
);
```

Modify the package.json script section

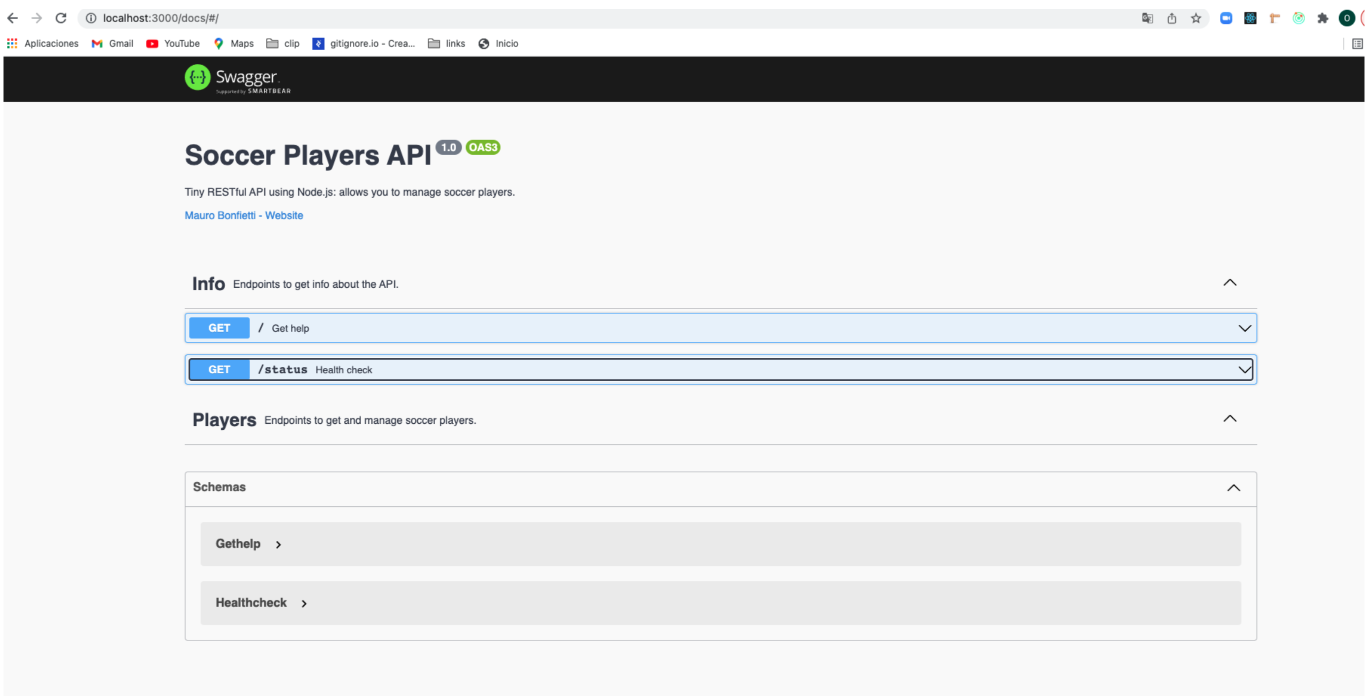
```
...
"scripts": {
        "start": "node src/index.js",
        "local": "nodemon --inspect=0.0.0.0:9229 src/index.js",
        "test": "jest --coverage"
    },
...
```

Start the server with `npm start`



```
oscar.oceguera@192  ~/Documents/repos/examples/localstack-clip/2_node_dynamodb  master ±  npm start

> 2_node_dynamodb@1.0.0 start /Users/oscar.oceguera/Documents/repos/examples/localstack-clip/2_node_dynamodb
> node src/index.js

Server listening @ http://localhost:3000
```

Consult documentation `http://localhost:3000/docs`



Soccer Players API 1.0 OAS3

Tiny RESTful API using Node.js: allows you to manage soccer players.

Mauro Bonfietti - Website

Info Endpoints to get info about the API.

GET / Get help

GET /status Health check

Players Endpoints to get and manage soccer players.

Schemas

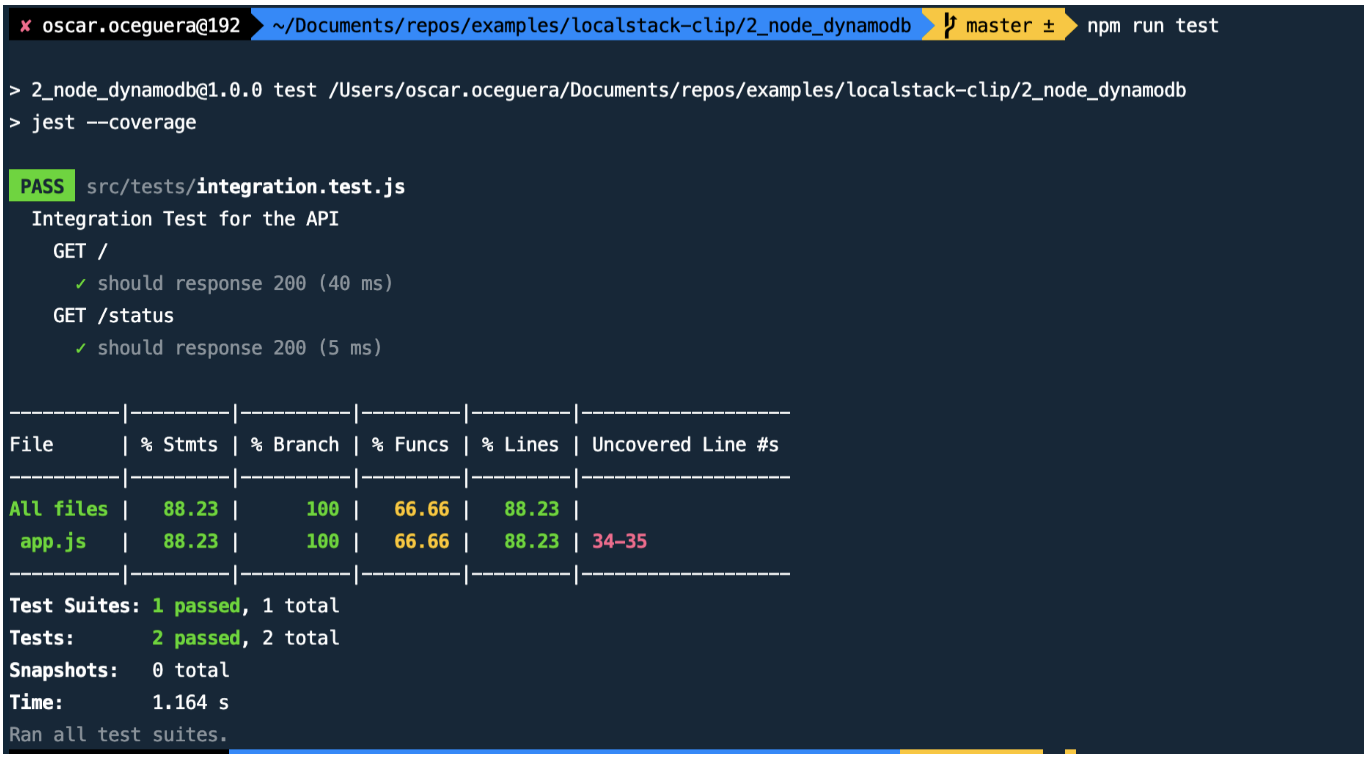
Gethelp >

Healthcheck >

Write some simple tests `src/tests/integration.test.js`

```
const request = require("supertest");
const app = require("../app");

describe("Integration Test for the API", () => {
  describe("GET /", () => {
    it("should response 200", async () => {
      const res = await request(app).get("/");
      expect(res.statusCode).toEqual(200);
      expect(res.body).toHaveProperty("message");
    });
  });

  describe("GET /status", () => {
    it("should response 200", async () => {
      const res = await request(app).get("/status");
      expect(res.statusCode).toEqual(200);
      expect(res.body).toHaveProperty("status", "api", "time");
    });
    });
  });
```

Run `npm test`



```
✘ oscar.oceguera@192  ~/Documents/repos/examples/localstack-clip/2_node_dynamodb  master ±  npm run test

> 2_node_dynamodb@1.0.0 test /Users/oscar.oceguera/Documents/repos/examples/localstack-clip/2_node_dynamodb
> jest --coverage

 PASS  src/tests/integration.test.js
  Integration Test for the API
    GET /
      ✓ should response 200 (40 ms)
    GET /status
      ✓ should response 200 (5 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   88.23 |      100 |   66.66 |   88.23 |
 app.js   |   88.23 |      100 |   66.66 |   88.23 | 34-35
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.164 s
Ran all test suites.
```