TUTORIAL ON HOW TO CREATE REACT APP AND INTEGRATE IT WITH AWS API GATEWAY

INSTRUCTOR
CLASS

Introduction:

In this tutorial we are going to learn how to create a react web client and integrate it with aws api gateway. So we use aws amplify to deploy our react app and we create an aws lambda function using python programming language to integrate with api gateway.

AWS Amplify

AWS Amplify is a set of purpose-built tools and features that lets frontend web and mobile developers quickly and easily build full-stack applications on AWS.

AWS Lambda

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers

AWS Api Gateway

Aws api gateway is a fully managed service which could be used to create restful apis.

Prerequisites:

- Nodejs
- Python
- Aws CLI

If you do not have Nodejs already installed in your environment follow the link here and install the Nodejs. We use npx to create our react app because it is the easiest way to create the app without worrying about internal setups done using webpack and babel.

You need python3 installed in your local machine if it is already there you need to download it from here and install it.

Pip will be available once you have installed python. After that you will need to install aws cli if you have not already configured the aws cli in your local machine.In a terminal try to paste below and install aws cli.

pip install awscli --upgrade --user

After that you need to configure the aws then past below command and provide correct values which are related to your aws account

aws configure

You will get below to add the data. For your user which is used to login to aws console must have an access key and secret. You have to remember the secret value you have entered when you are creating the access key. Please login to aws console and search for IAM and open IAM console navigate to your user and you will find the plase to create a key if you have not already done it.Please follow this guide to get more details.

AWS Access Key ID [None]: ***PASTE ACCESS KEY ID HERE***

AWS Secret Access Key [None]: ***PASTE SECRET ACCESS KEY HERE***

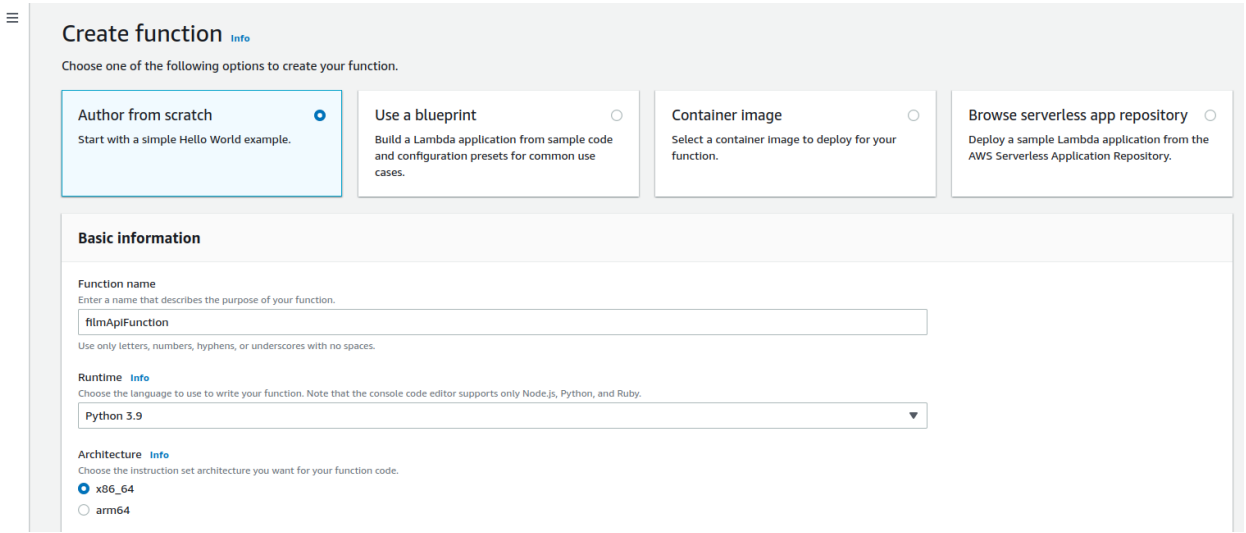
Default region name [None]: ***TYPE YOUR PREFERRED REGION***

Default output format [None]: json

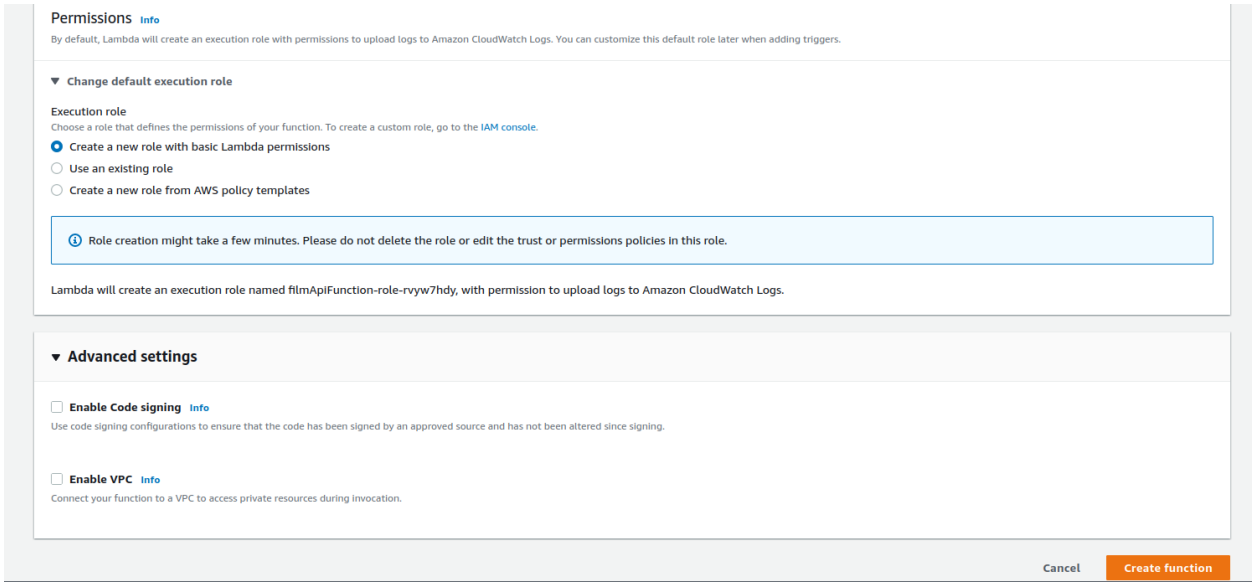
All good upto now right.Ok let's start creating our first restful api first.

Creation of rest api:

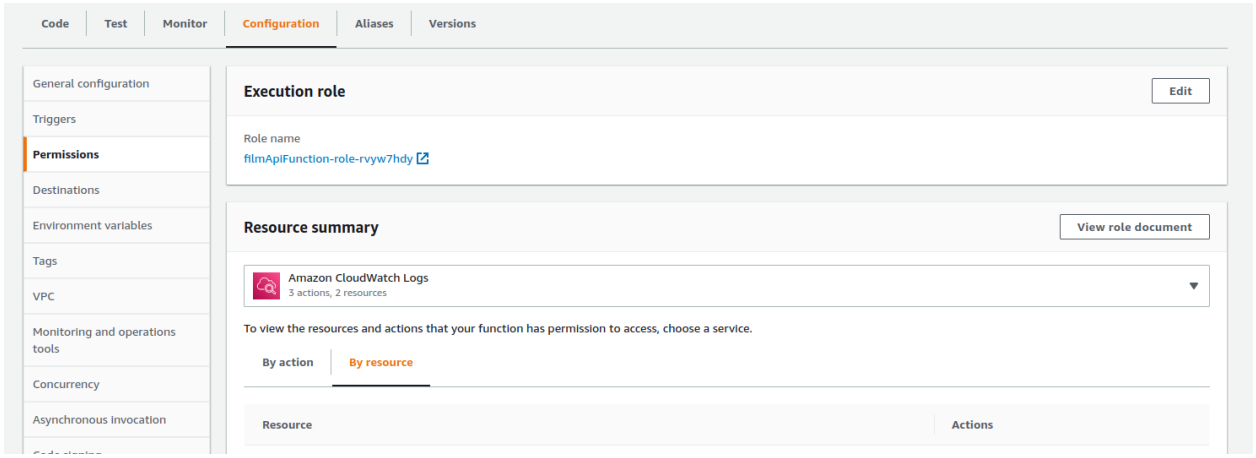
As mentioned in the introduction we use aws lambda function to create backend functionality of our api. So login to aws console and search for aws lambda and navigate to aws lambda. Then click on create button to create a lambda function.



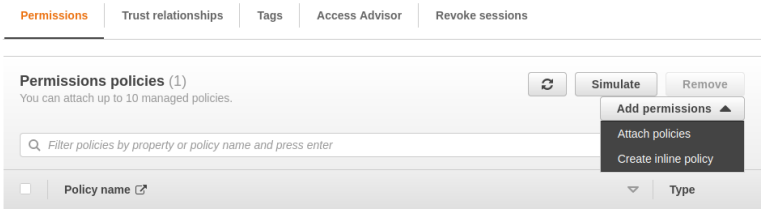
Add a function name and runtime in my case I have selected python3.9.In permissions choose create new role with basic lambda permission.



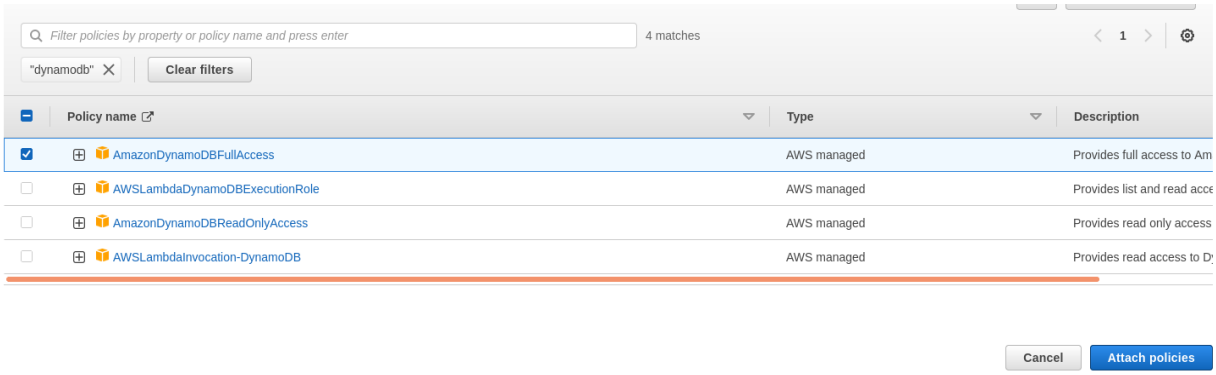
After that click on the create function button. Ok now we have created the aws lambda function and we need to access the dynamodb from our lambda function for that we need to setup permission with the role we got created for the lambda.Let's navigate to the configuration tab and select permission part in left pane of lambda console and navigate to the role.



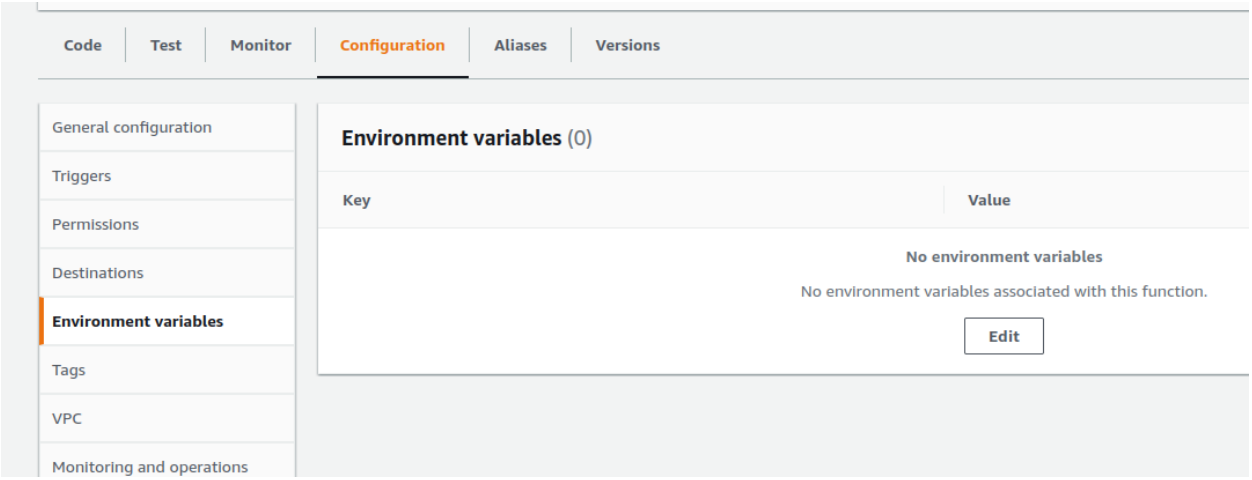
One you click on the role name it will navigate to the IAM console for the role.There under the permission tab you have a button to Add permission. Click on it and select attach policies from the drop down.



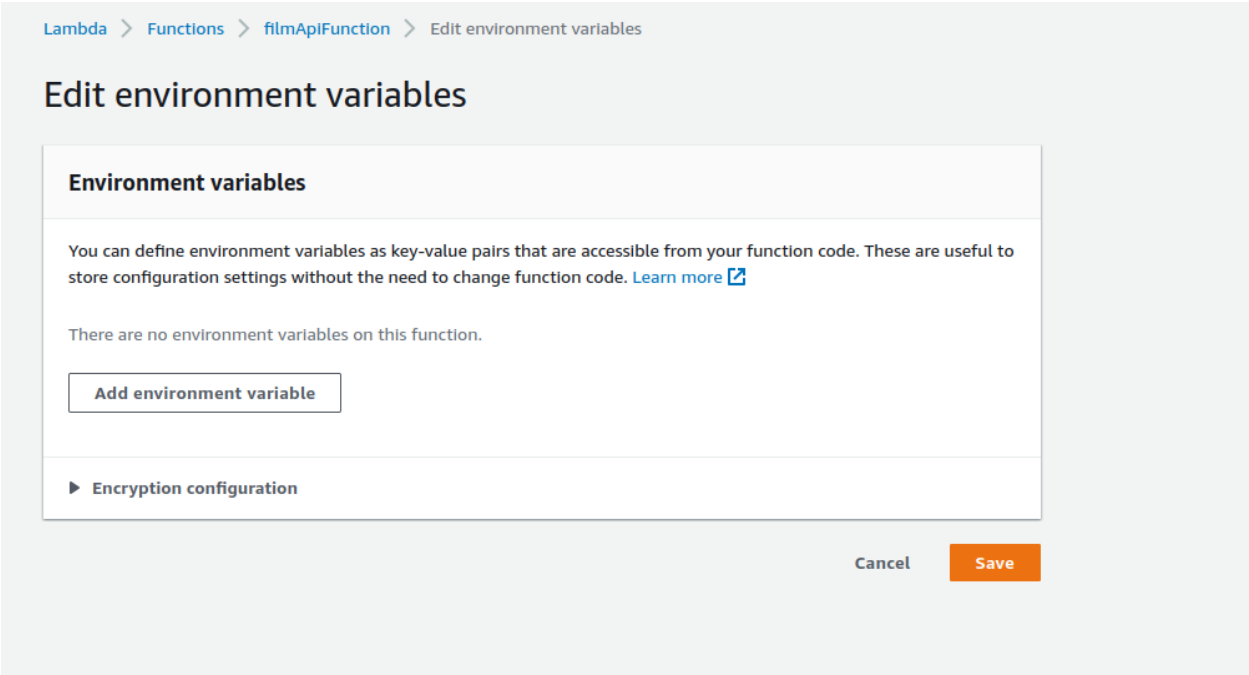
Then search for the dynamodb and select the full access permission set amd click on attach policies.



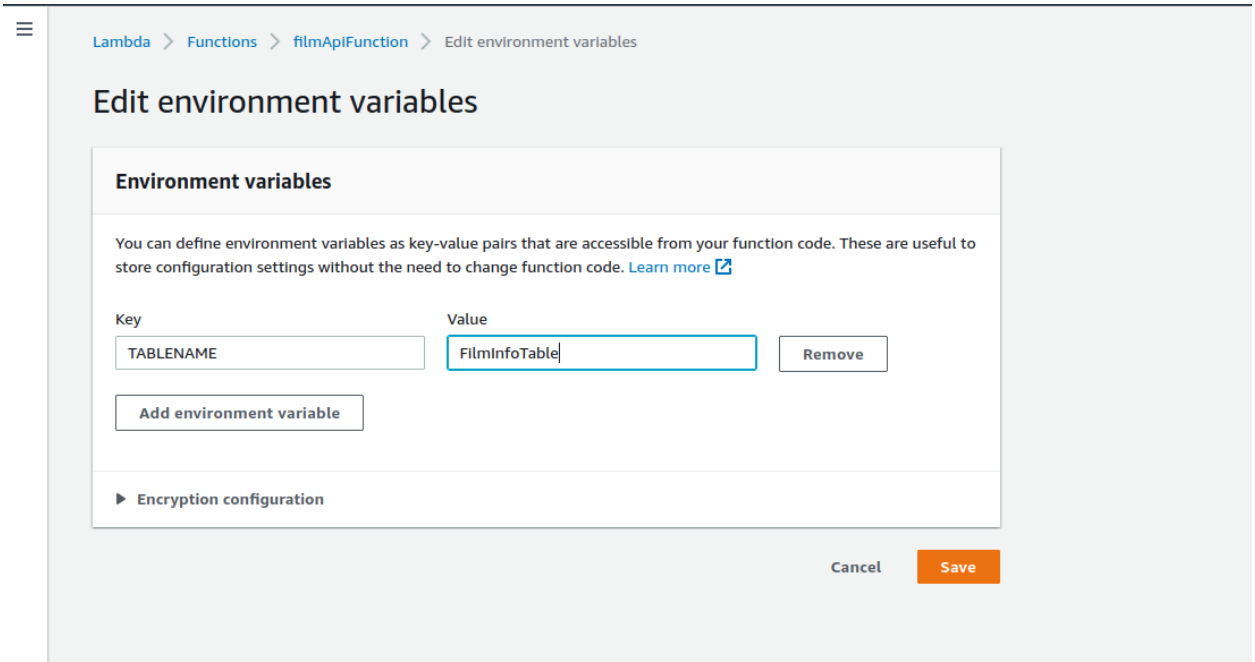
Ok well done we have successfully attached the policy.Let's go back to lambda console and add an environment variable to the lambda. In here we are going to pass the dynamodb table name as an env variable.
Under configurations select the environment variable in the left pane.



After that click on the edit button to add a variable.And it will show below window there you should click on the Add environment variable button.



Add the variable named TABLENAME with your db name as the value and click on the save button.



Ok we have added the table name but did not create it let’s create our dynamoDB table.

Create a json file named db-table.json and add this content in it.

```
{
   "TableName": "FilmInfoTable",
   "KeySchema": [
       { "AttributeName": "Director", "KeyType": "HASH" },
       { "AttributeName": "FilmTitle", "KeyType": "RANGE" }
   ],
   "AttributeDefinitions": [
       { "AttributeName": "Director", "AttributeType": "S" },
       { "AttributeName": "FilmTitle", "AttributeType": "S" }
   ],
   "ProvisionedThroughput": {
       "ReadCapacityUnits": 5,
       "WriteCapacityUnits": 5
   }
}
```

Here we are creating a dynamodb table named FilmInfoTable and it's partition key is Director and the sort key is the FilmTitle

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

Sort key

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

More about the provisioned information can be found here.

Let’s create the table and add the data to the table now.

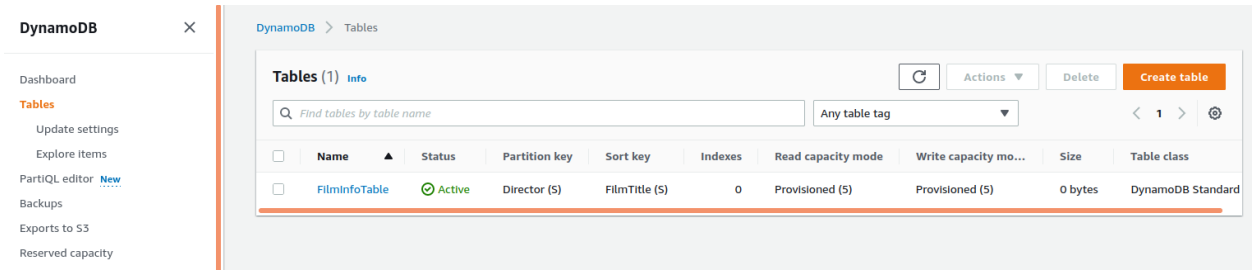
Run the below command in the terminal where your json file exists.

aws dynamodb create-table --cli-input-json file://db-table.json

You will see a output like this:


```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Director",
                "AttributeType": "S"
            },
            {
                "AttributeName": "FilmTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "FilmInfoTable",
        "KeySchema": [
            {
                "AttributeName": "Director",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "FilmTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": 1648446951.308,
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:eu-west-1:057186357501:table/FilmInfoTable",
        "TableId": "edcc9375-36f5-49ac-bd4c-7dde8e425649"
    }
```


You can navigate to aws console and search for the dynamodb and go dynamodb and see the table which you created.



Let’s add some data to this table now. Create a json file named test-data.json and add the below content in it.

```
{
   "FilmInfoTable": [{
           "PutRequest": {
               "Item": {
                   "Director": {
                       "S": "ADAM"
                   },
                   "FilmTitle": {
                       "S": "Sleep less"
                   }
               }
           }
       },
       {
           "PutRequest": {
               "Item": {
                   "Director": {
                       "S": "Stefan"
                   },
                   "FilmTitle": {
                       "S": "Paint ball"
```

```
                    }

                }

            }

        },

        {

            "PutRequest": {

                "Item": {

                    "Director": {

                        "S": "Lisa"

                    },

                    "FilmTitle": {

                        "S": "Fedup"

                    }

                }

            }

        }

    ]

}
```

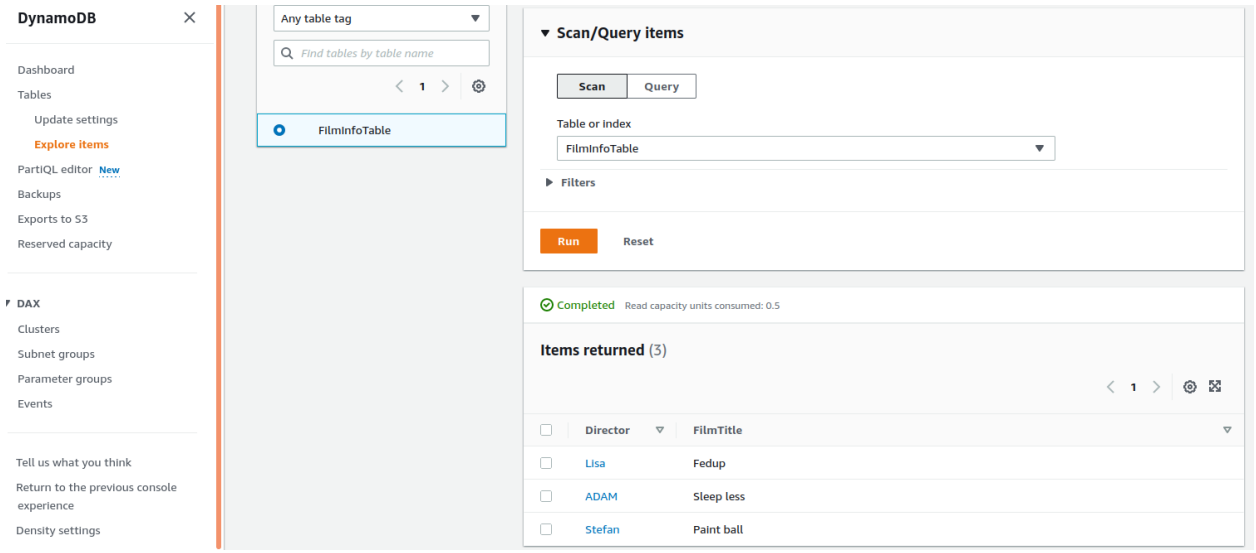
Now execute below command.

aws dynamodb batch-write-item --request-items file://test-data.json

You should see a result like this

{"UnprocessedItems": {}}

If you explore the items in the newly created table you will see the inserted data.

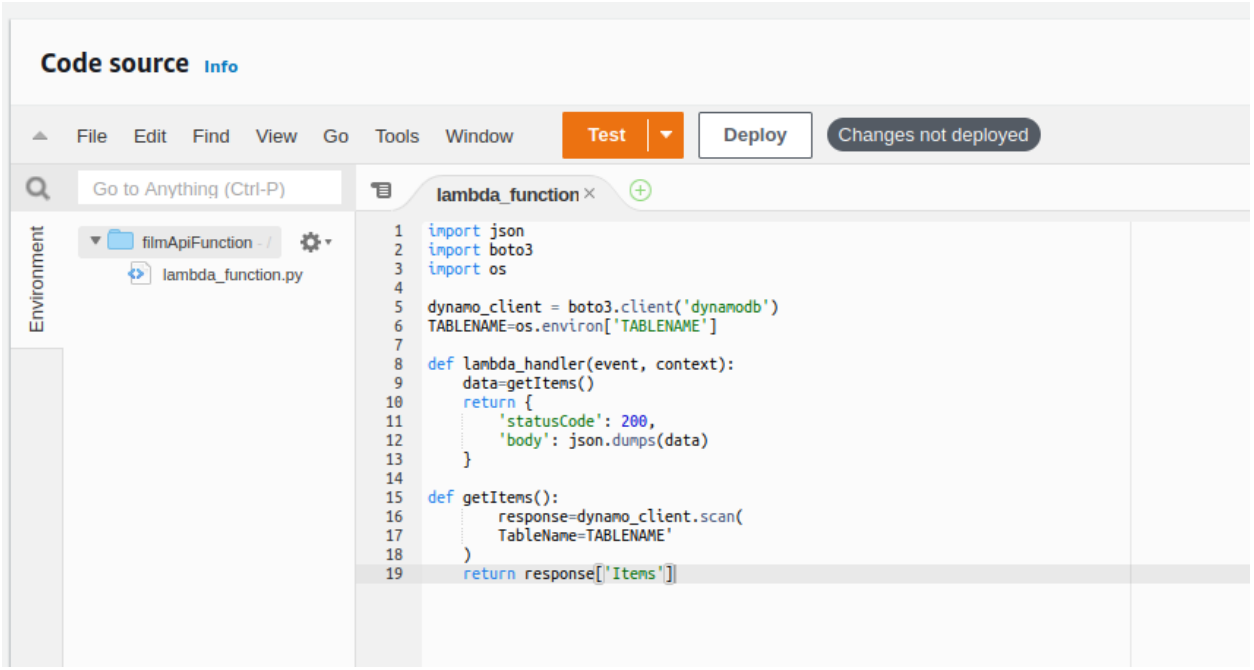


That's all about the configuration.Let's start coding our function. Navigate back to the code tab and add the below code to your function.

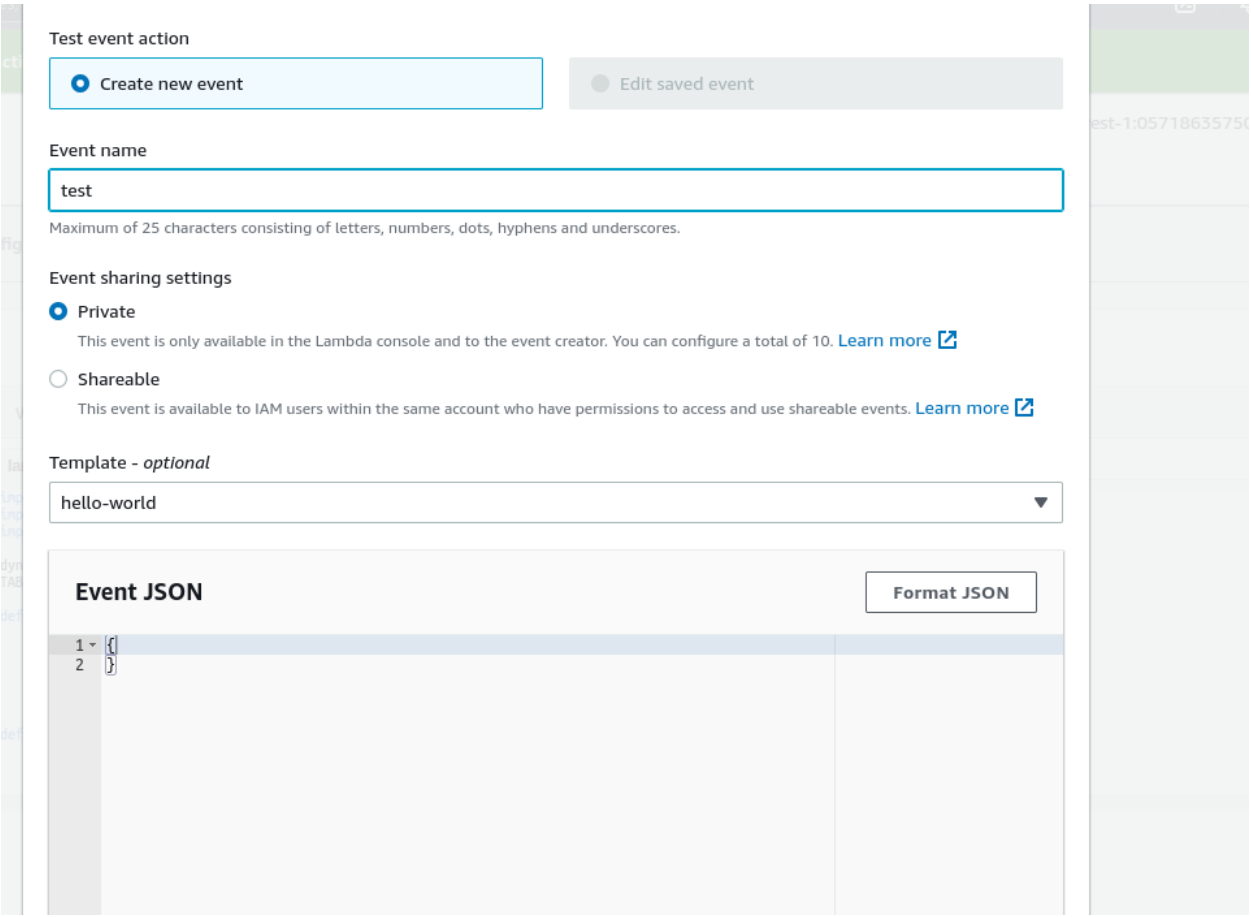
```
import json
import boto3
import os

dynamo_client = boto3.client('dynamodb')
TABLENAME=os.environ['TABLENAME']

def lambda_handler(event, context):
        data=getItems()
        return {
```

```
        'statusCode': 200,
        'body': json.dumps(data)
        }

def getItems():
        response=dynamo_client.scan(
        TableName=TABLENAME
        )
        return response['Items']
```

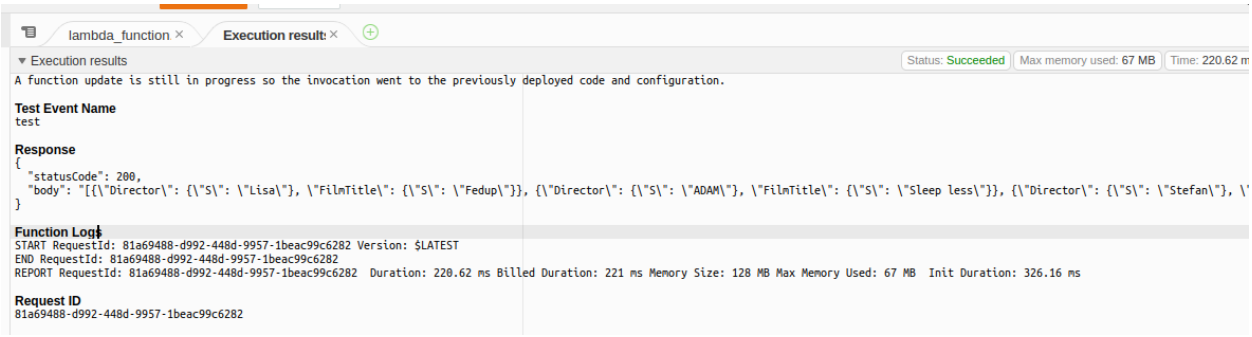
Now you have to click on the deploy button.



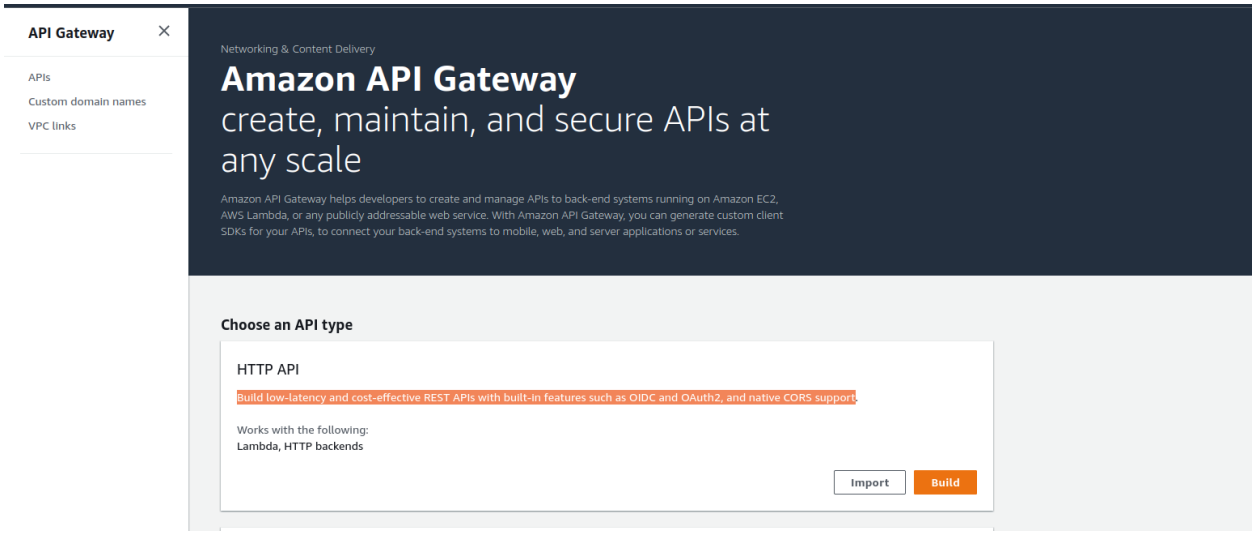
Once you deploy your code we can test our code click on test button and add below configurations as in the image and click on save.



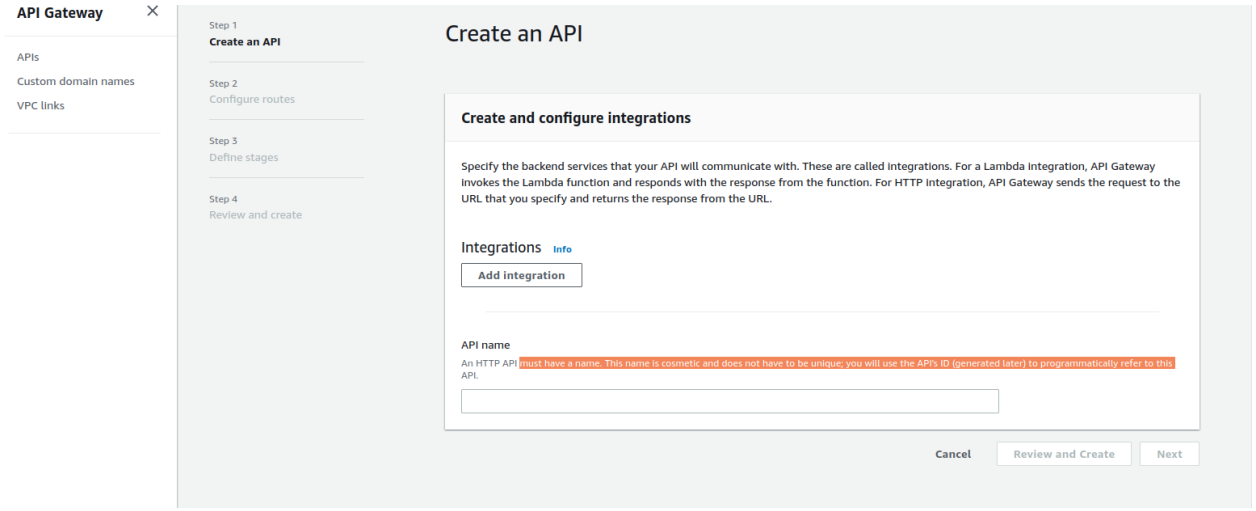
After that again click on the test button.So you will see the result of the function execution as below.



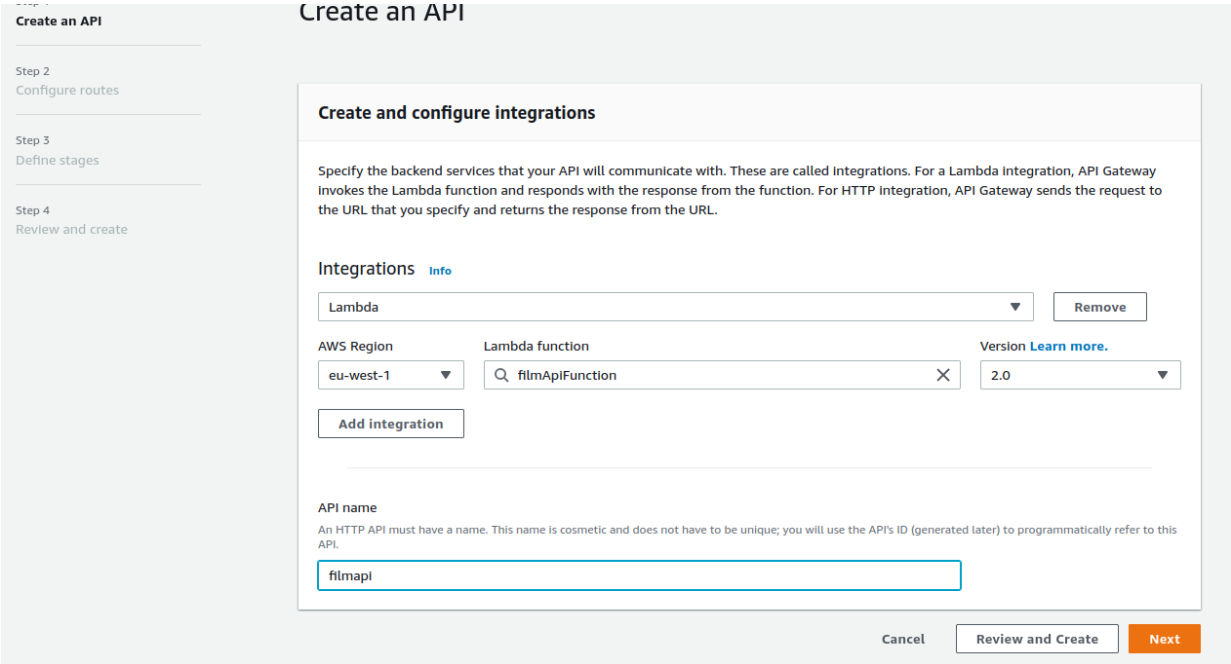
Ok we can see the results were returned correctly. Then let's create the api gateway to integrate with lambda.Search for the apigateway and navigate to api gateway in aws console. And click on create api gateway.There you can use HTTP Api and click on build button.



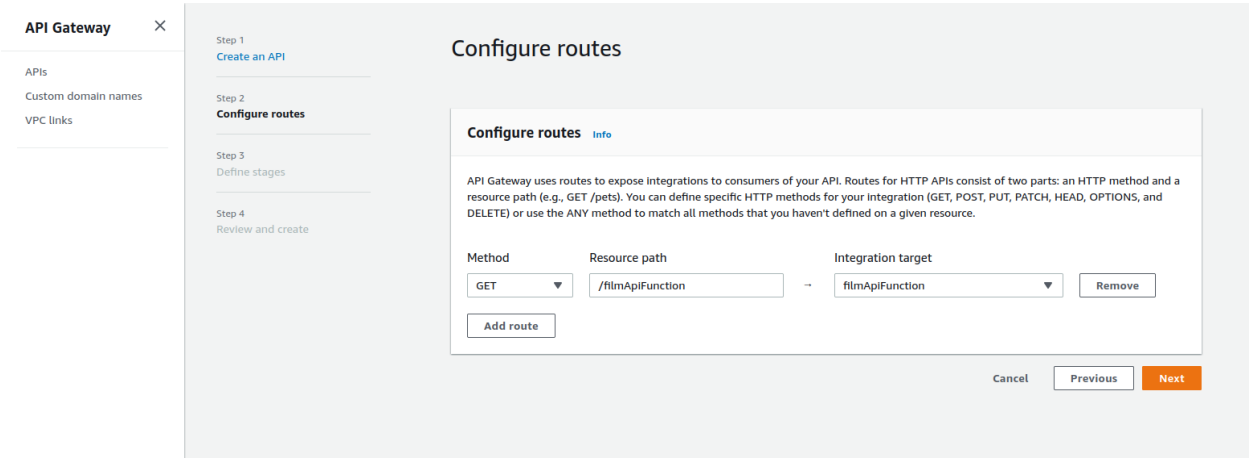
Once you click on build you will see the window below.



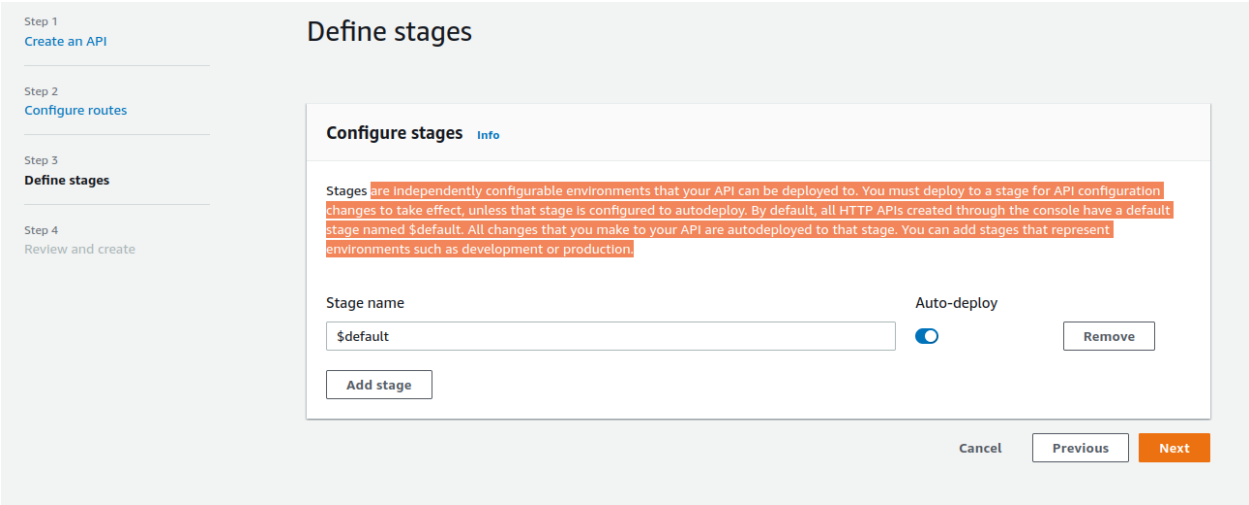
Click on add integration then and add the details as below.Here you select the lambda function you have already created in previous step.



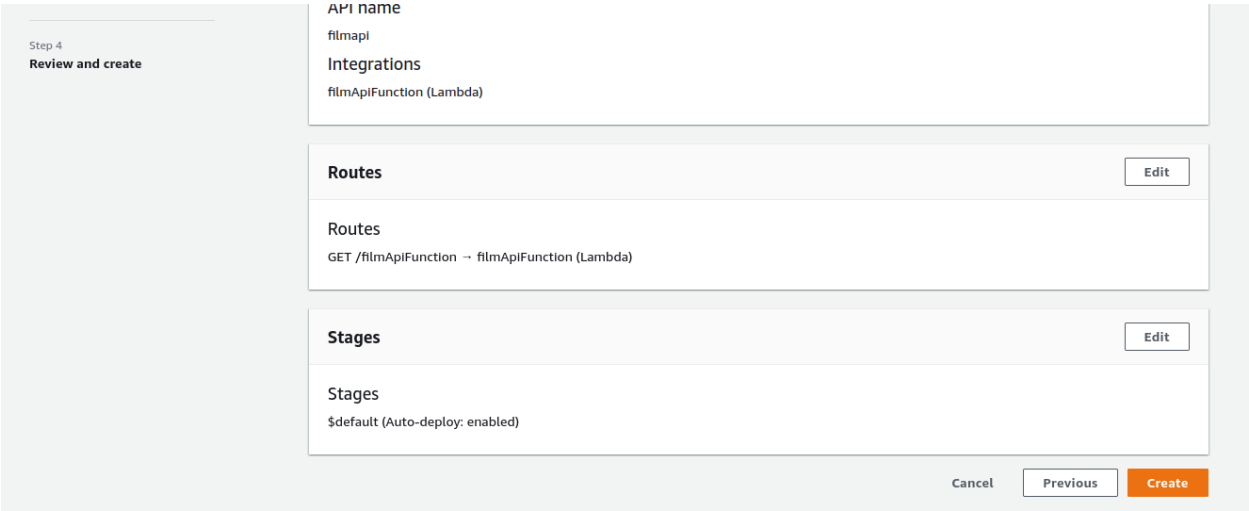
Then click next.



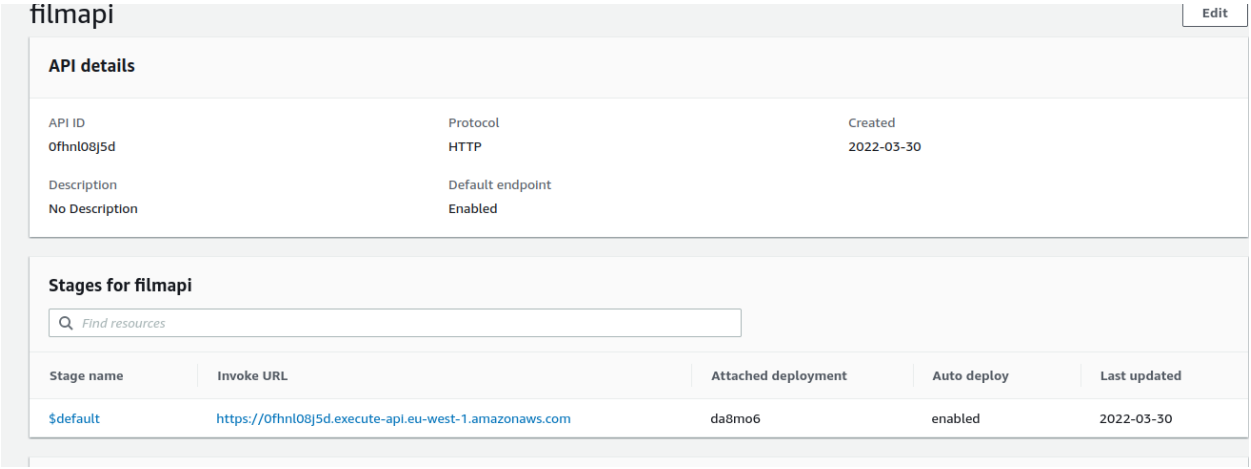
Add the route method as Get and click next.



Keep the default on stage and click next.

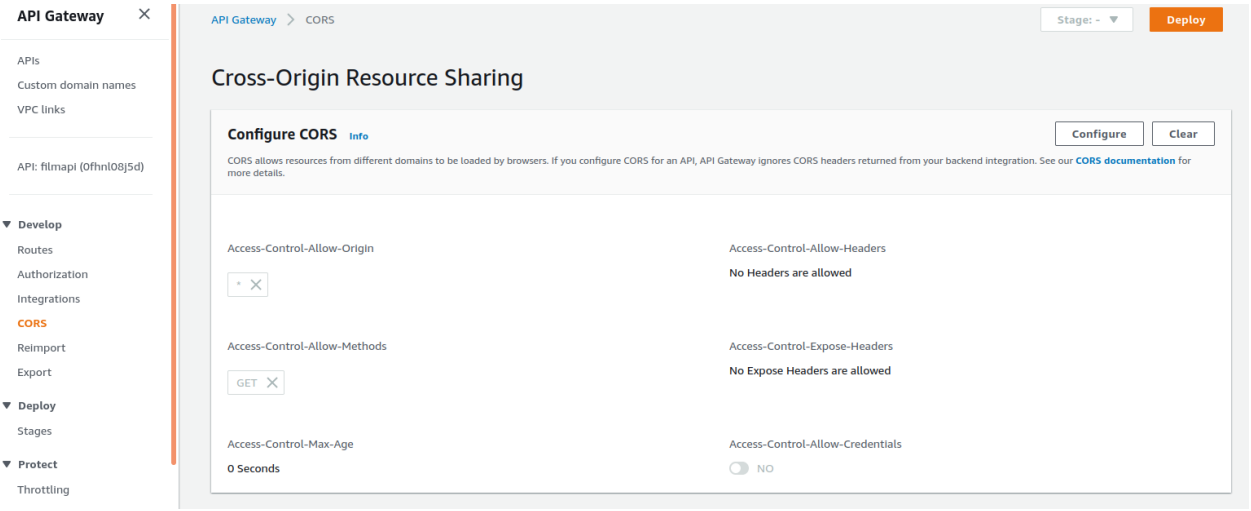


Now click on create to create api gateway.



Here you will see the invoke url . You can now test the api with the route you created. So the link will look like this.
<invoke url>/filmApiFunction

Now our api is up and running we can see the results.But still we need to enable cors to access the api from react app so in apigateway navigate to cors part and add below configuration.



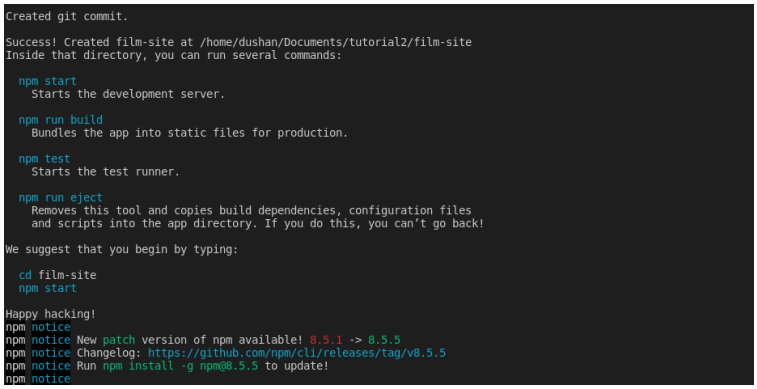
Let’s create our react app as the next step.

Creation of react app:

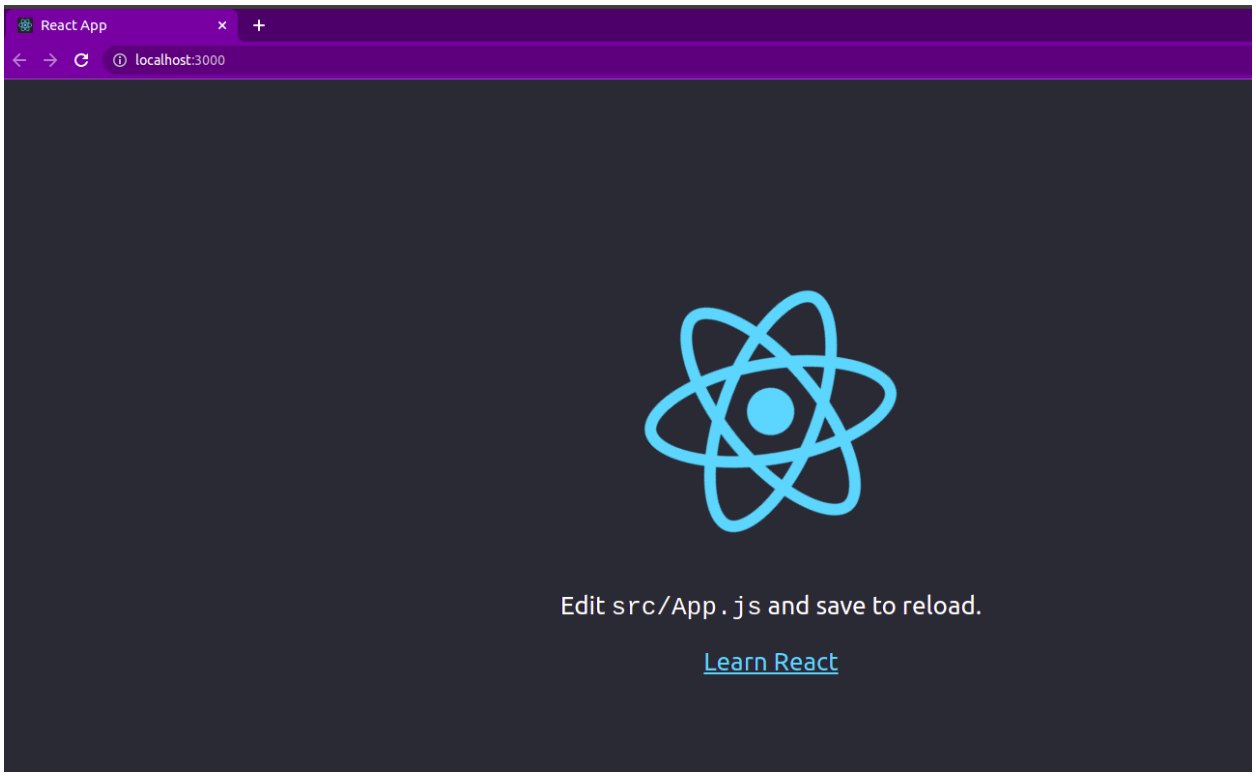
Open a terminal in the machine in a preferred location and type below command to create a react app.

npx create-react-app film-site

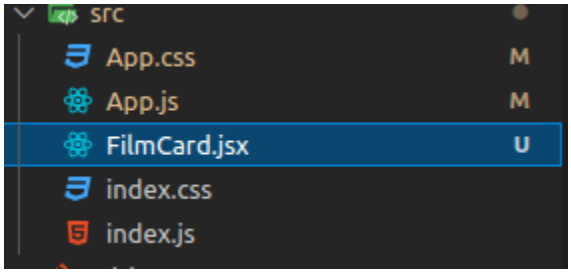
You will see a result like this at the end.



Now navigate to file-site in terminal by **cd film-site** and in there run **npm start** and see the result.If you see below result the app created working properly in your local environment.



Then let's clean the src folder a bit and start creating files.I have created a new file named FilmCard.jsx. And removed several files from example see below folder structure.



Now Let's edit our App.js file to visualize our data. We use the userEffect hook to fetch the data from our api gateway when the web is initialized. Then we display the information on a card.

```
import React from 'react';
import './App.css';
import { useState, useEffect } from 'react'
import FilmCard from './FilmCard';

const                                API_URL                                =
'https://0fhnl08j5d.execute-api.eu-west-1.amazonaws.com/filmApiFunction';

function App() {
   const movie1={"Director": {"S": "Lisa"}, "FilmTitle": {"S": "Fedup"}}
   const [films, setFilms] = useState([])
   const movieList = async() => {
       const response = await fetch(API_URL);
       const data = await response.json();
       setFilms(data)
       console.log(data);
   };
   useEffect(() => {
       movieList();
   }, []);

   return (
     <div className = "App">
       <h1> Movie Flix </h1>
       {
         films?.length > 0
         ? (
```

```
          <div className='container'>
            {films.map((film) => (
              <FilmCard movie={film}/>
            ))}
          </div>
        ):
        (
          <div>
            <p>Movies not found</p>
          </div>
        )
        }

      </div>
    );
}

export default App;
```

I have created a separate component to handle data of the card open newly created FilmCard.jsx file and add below content.

```
import React from "react"

const FilmCard = ({movie}) => {
 return (
   <div className='movie'>
     <p>{movie['Director']['S']}</p>
     <div>
       <img src='https://via.placeholder.com/400'></img>
     </div>
     <div>
       <h3>{movie['FilmTitle']['S']}</h3>
     </div>
   </div>
 );
}

export default FilmCard;
```

Then let's add some css to our project using app.css.

```css
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
      animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

.movie {
  width: 310px;
  height: 460px;
  margin: 1.5rem;
  position: relative;
  border-radius: 12px;
  overflow: hidden;
  border: none;
  transition: all 0.4s cubic-bezier(0.175, 0.885, 0, 1);
  box-shadow: 0px 13px 10px -7px rgba(0, 0, 0, 0.1);
```

```
}

.movie div:nth-of-type(1) {
   width: 100%;
   height: 100%;
}

.movie div:nth-of-type(1) img {
   height: 100%;
   width: 100%;
}

.movie div:nth-of-type(2) {
   z-index: 2;
   background-color: #343739;
   padding: 16px 24px 24px 24px;
   position: absolute;
   bottom: 0;
   right: 0;
   left: 0;
   font-family: "Roboto Slab", serif;
   color: #f9d3b4;
}

.container {
   width: 100%;
   margin-top: 3rem;
   display: flex;
   justify-content: center;
   align-items: center;
   flex-wrap: wrap;
}

.movie div:nth-of-type(3) {
   z-index: 2;
   background-color: #343739;
   padding: 16px 24px 24px 24px;
   position: absolute;
   bottom: 0;
   right: 0;
```

```
    left: 0;
}

@keyframes App-logo-spin {
    from {
        transform: rotate(0deg);
    }
    to {
        transform: rotate(360deg);
    }
}
```

Ok now we need to change the header of our app.Navigate to public folder and edit the index.html file> head> title to ‘Movie Flix’

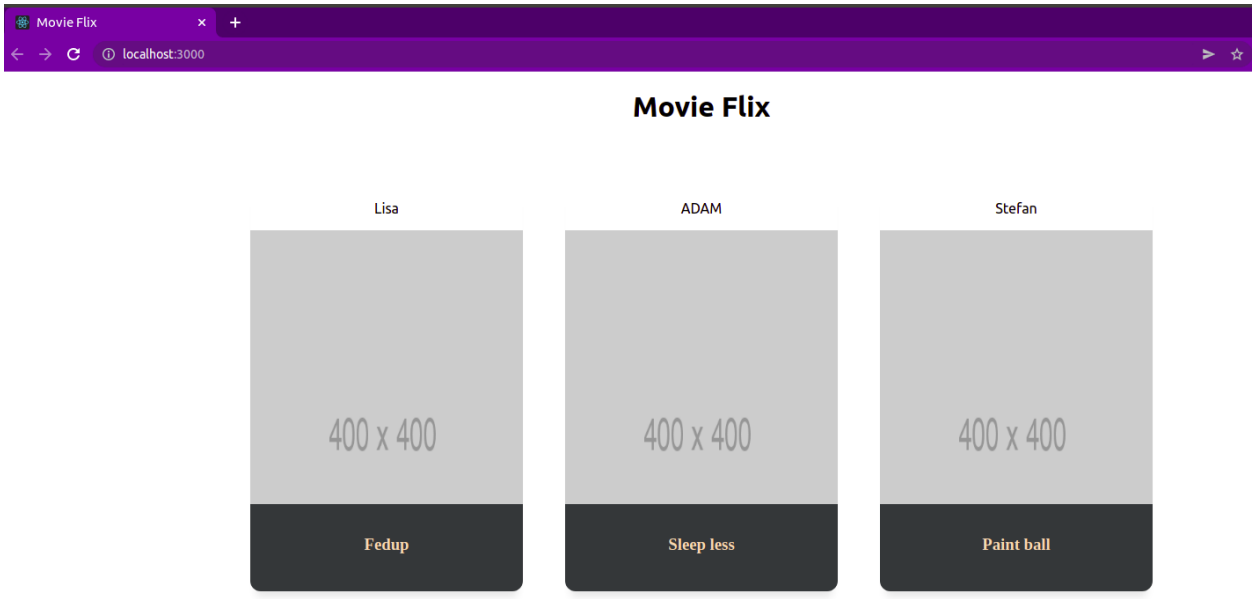
```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="utf-8" />
   <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
   <meta name="viewport" content="width=device-width, initial-scale=1" />
   <meta name="theme-color" content="#000000" />
   <meta
     name="description"
     content="Web site created using create-react-app"
   />
   <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
   <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
   <title>Movie Flix</title>
 </head>
 <body>
   <noscript>You need to enable JavaScript to run this app.</noscript>
   <div id="root"></div>

 </body>
</html>
```

Our Index.js file looks like this.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
```

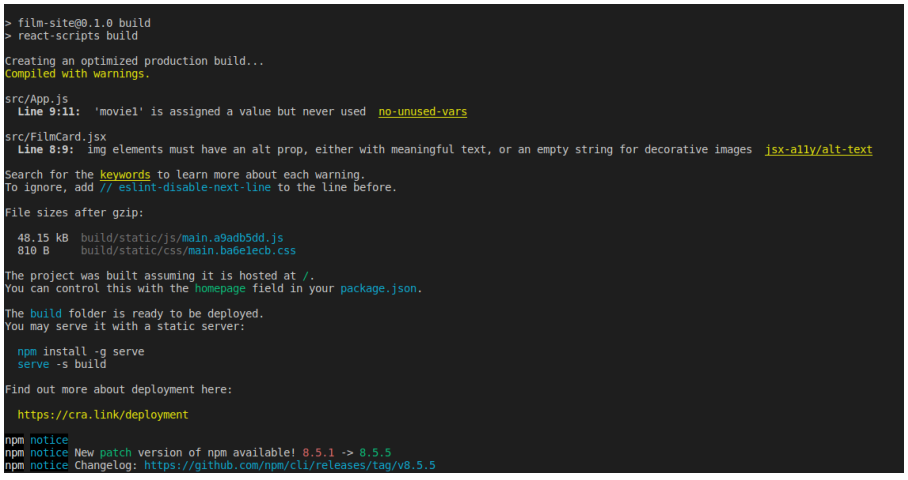
```
import App from './App';
ReactDOM.render(
<React.StrictMode>
   <App />
</React.StrictMode>, document.getElementById('root') );
```

Time to see our changes in the local environment.In terminal type npm start



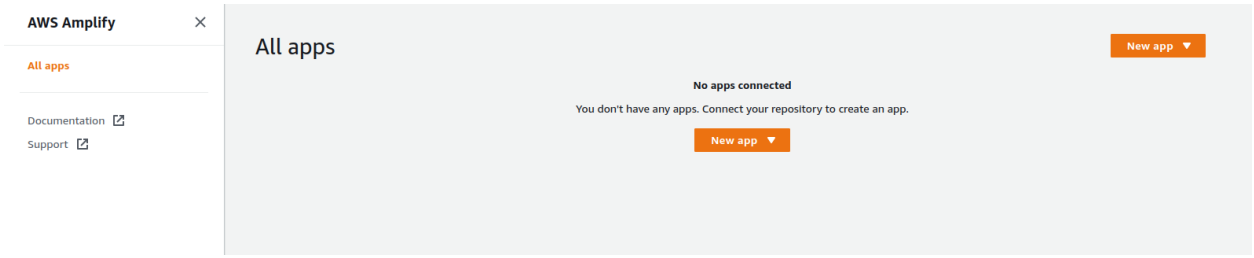
Ok now let's build our project using below command
npm run build

You will see a result like this and a build folder will get created in the root directory.

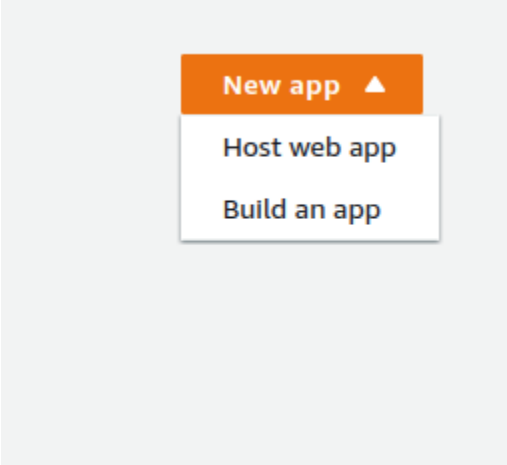


Deploy React app to aws amplify:

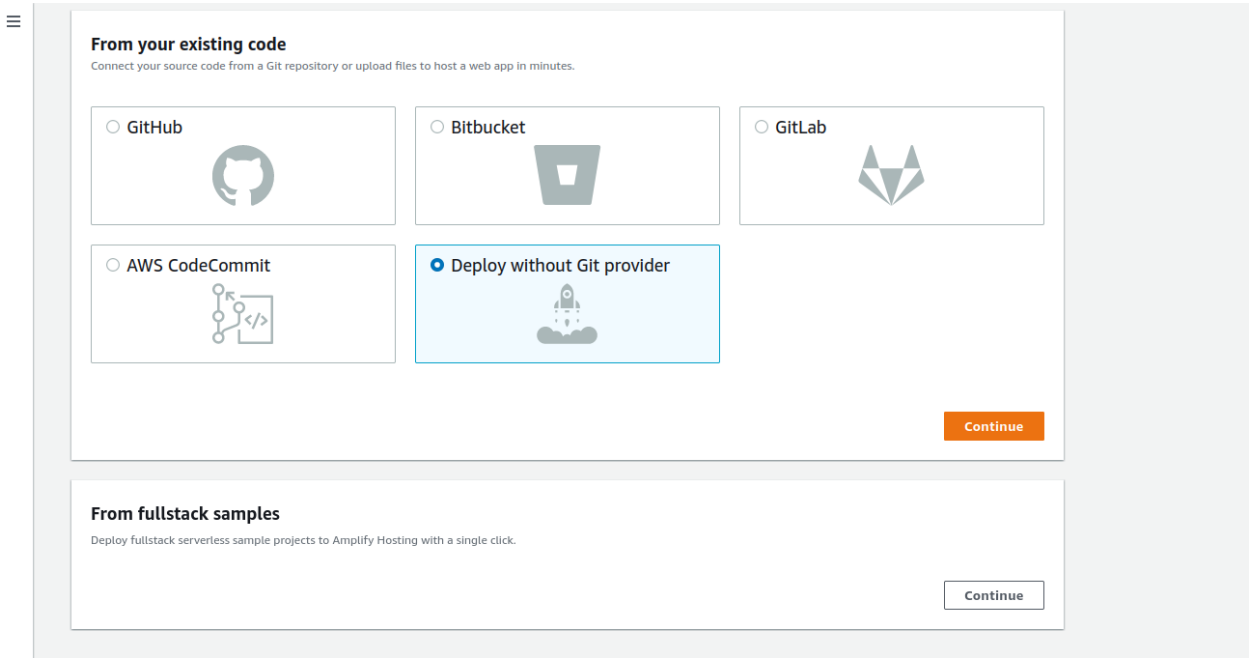
In aws console navigate to aws amplify.



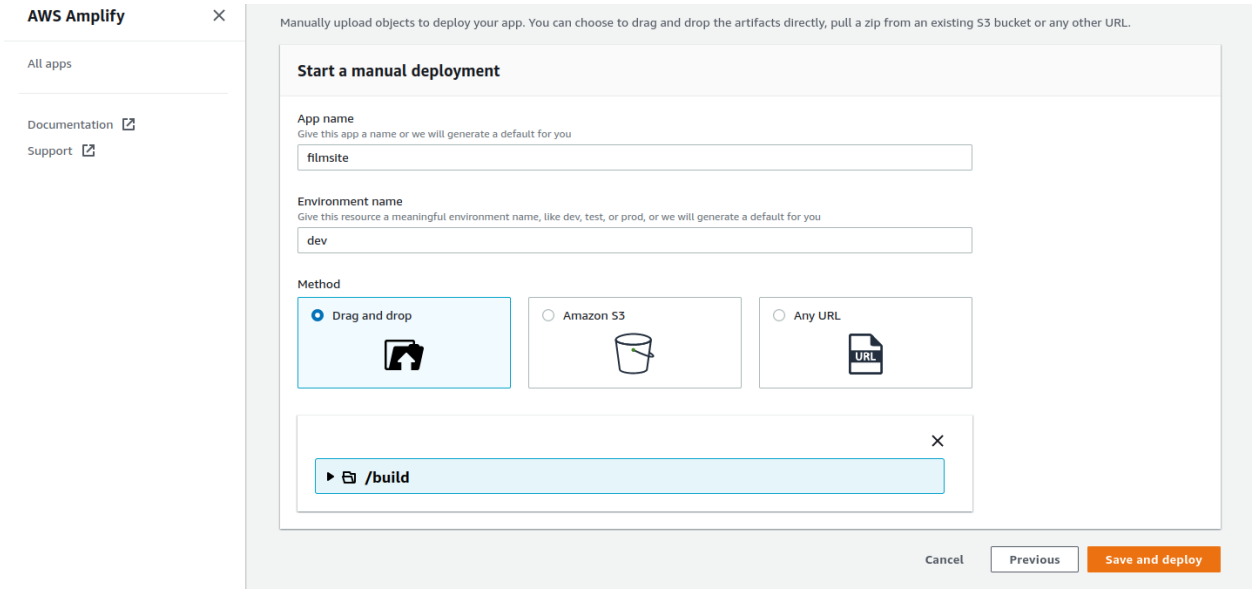
Then click on create new app button and select Host web app from drop down.



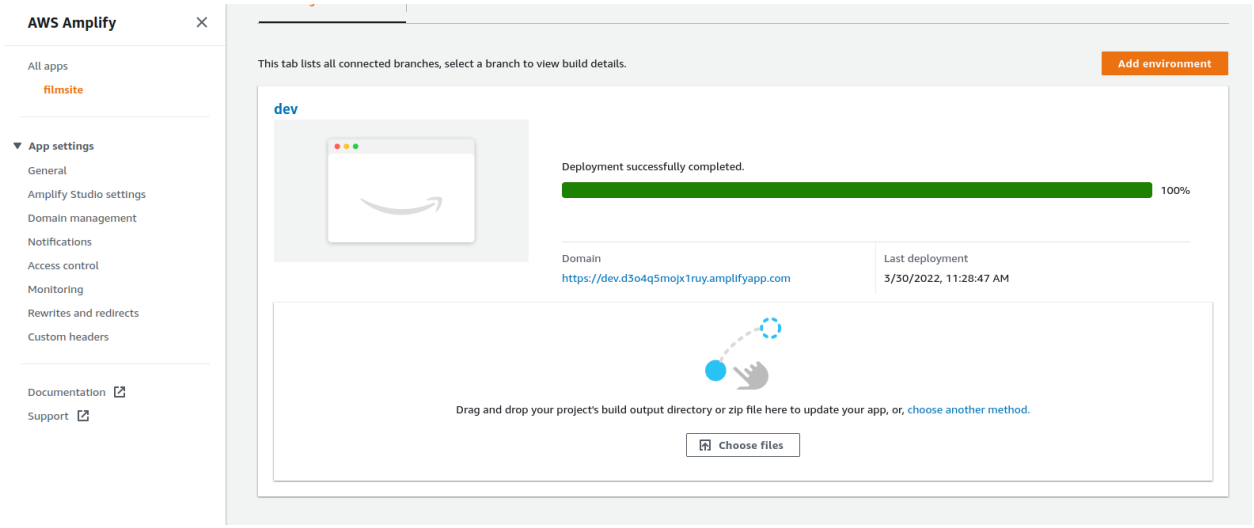
In here I'm planning to upload the build folder straightforwardly to amplify so I'm selecting Deploy without git provider.



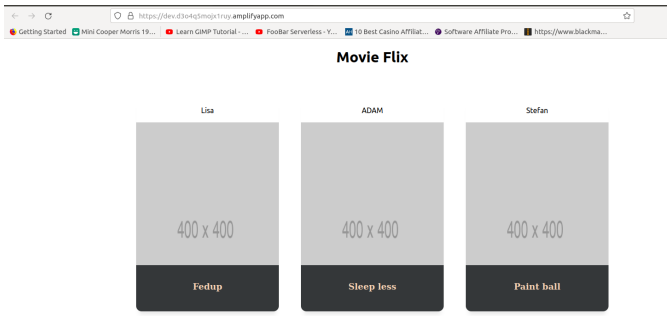
In the next window add an app name and an environment name as you wish and drag and drop the build folder of your react project.



Once you upload click on save and deploy button.



Once the deployment is successful. You can navigate to your web page using the domain provided.



Ok We are done with our React sample app deployment in aws amplify.

THE END.