School of Information Technology

NANYANG POLYTECHNIC

Guide to setting up ci/cd with amplify, codecommit and cypress

Table of Contents

[Setting up Cypress in Local Project Folder 3](#_Toc49116672)

[Downloading and Setting up Cypress 3](#_Toc49116673)

[Writing the first test case 4](#_Toc49116674)

[Running the first test case 5](#_Toc49116675)

[Setting up CodeCommit in AWS 8](#_Toc49116676)

[Setting up Amplify using CodeCommit 14](#_Toc49116677)

# Setting up Cypress in Local Project Folder

## Downloading and Setting up Cypress

For reference, here is the official documentation:

<https://docs.cypress.io/guides/getting-started/installing-cypress.html#System-requirements>

Assuming you already have the following:

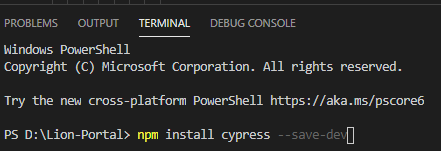
Project folder with codes inside

Initialised folder with ‘npm init’

Visual Studios Code with project folder loaded and ready to run (start on localhost)

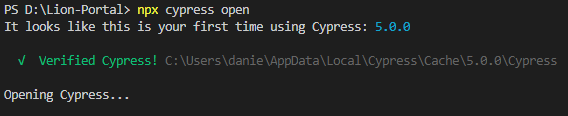
In Visual Studios Code, open the terminal and enter the following command:

**npm install cypress --save-dev**

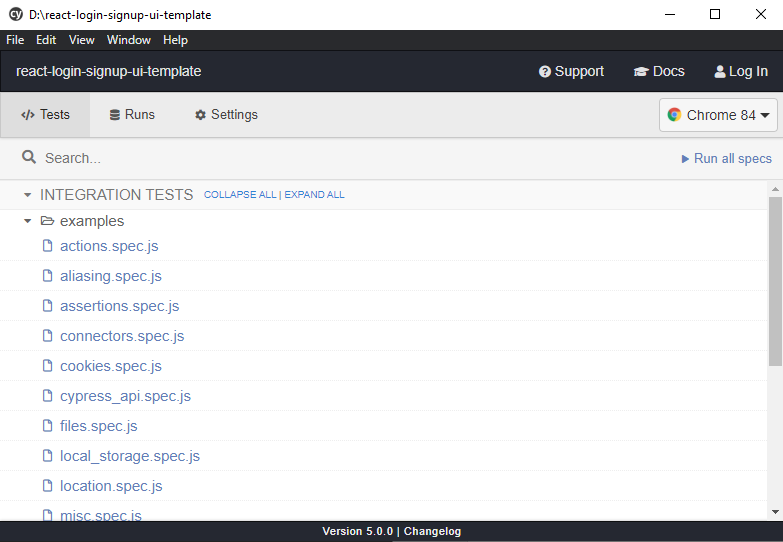


Then enter the following:

**npx cypress open**

****

And then the Cypress browser should open with a lot of random files inside.

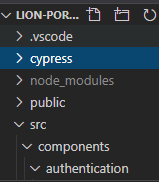


Each of these files are test cases, they are prewritten examples for you to refer to when building your own test cases.

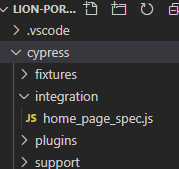
That’s all for setting up Cypress.

## Writing the first test case

Now go to your project folder and you should see a folder named ‘cypress’ inside.



Open it and navigate to the ‘integration’ folder, delete the ‘examples’ folder and create a new file inside called home\_page\_spec.js.



## Running the first test case

Go to home\_page\_spec.js and type in the following code

describe('My First Test', () => {

it('Does not do much!', () => {

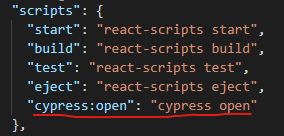
expect(true).to.equal(true)

})

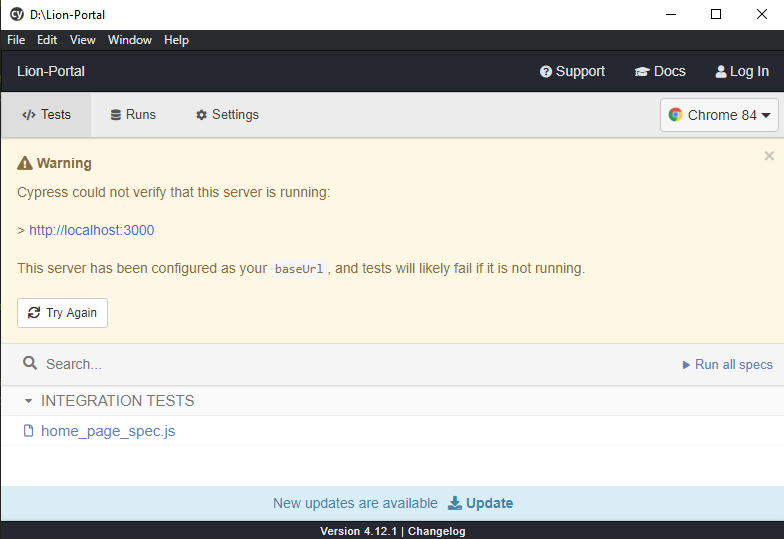
})

Save the file, then go to package.json and add the following line in your scripts.

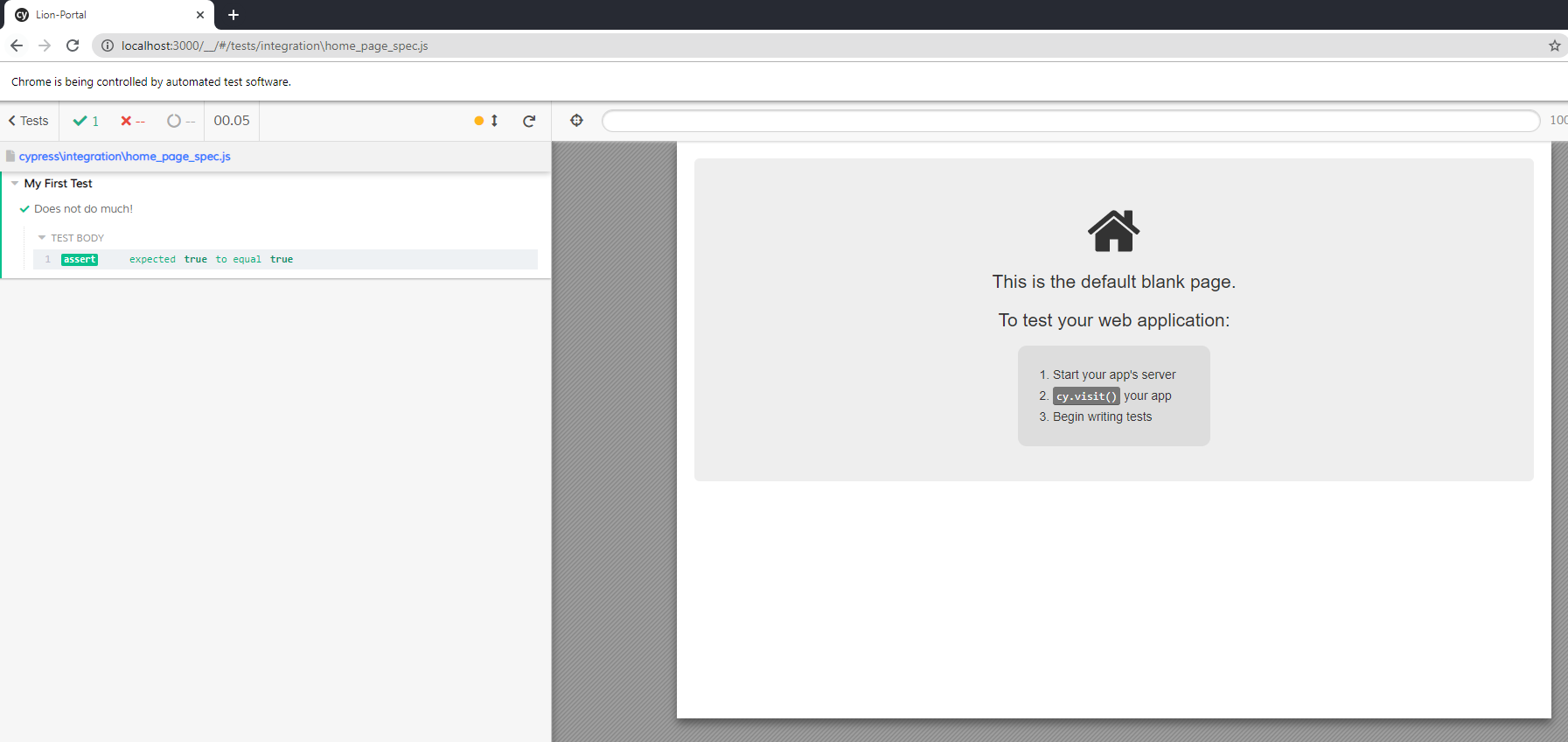
**“cypress:open”: “cypress open”**



Start the project and once it has loaded, run the cypress:open script.



This screen will appear, click the home\_page\_spec.js link there under integration tests.



Google Chrome should open with this window. Now that you have written your first test, try adding another test to link to the localhost webpage.

Clear the contents of the home\_page\_spec.js file and replace it with the code below:

**describe('The Home Page', () => {**

**it('successfully loads', () => {**

**cy.visit('/') // change URL to match your dev URL**

**})**

**})**

**describe('Logging in with incorrect credentials', () => {**

**it('fails to log in', () => {**

**cy.get('#username')**

**.type('abc')**

**cy.get('#password')**

**.type('1234');**

**cy.get('.login-btn')**

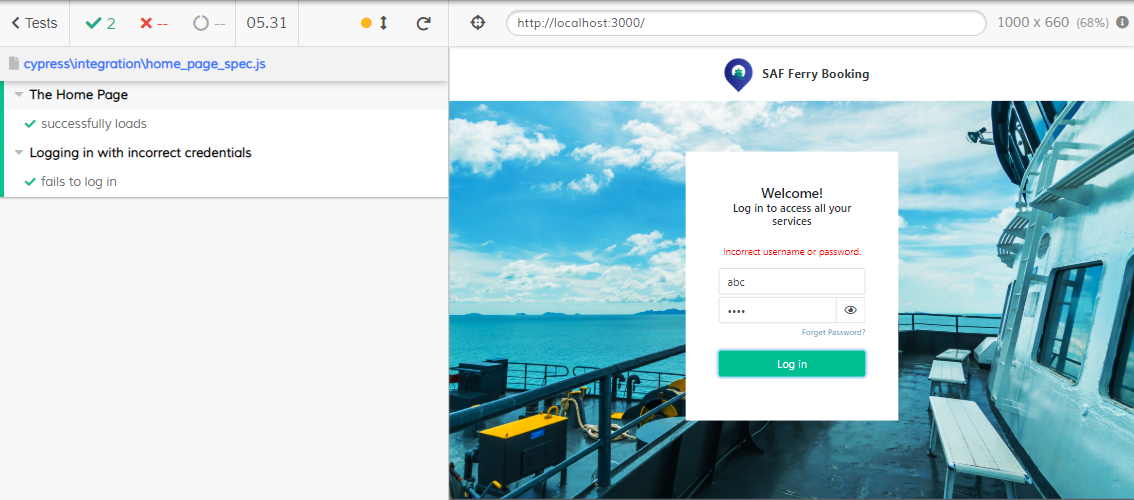
**.click()**

**})**

**})**

The code above should access the localhost website, and type in ‘abc’ and ‘1234’ as the username and password respectively, by finding the input fields by Id. Then it should find the submit button using the .login-btn class and then click on it. All this simulates an actual user accessing the website.

After changing the file contents and saving it, you should notice that the Cypress window in Chrome should automatically reflect the changes and start running the test cases newly written.



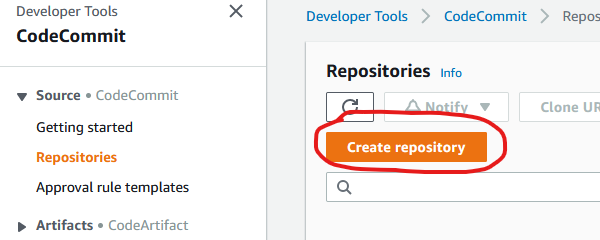
More guidance as to how to write test cases can be found at:

<https://docs.cypress.io/guides/overview/why-cypress.html#In-a-nutshell>

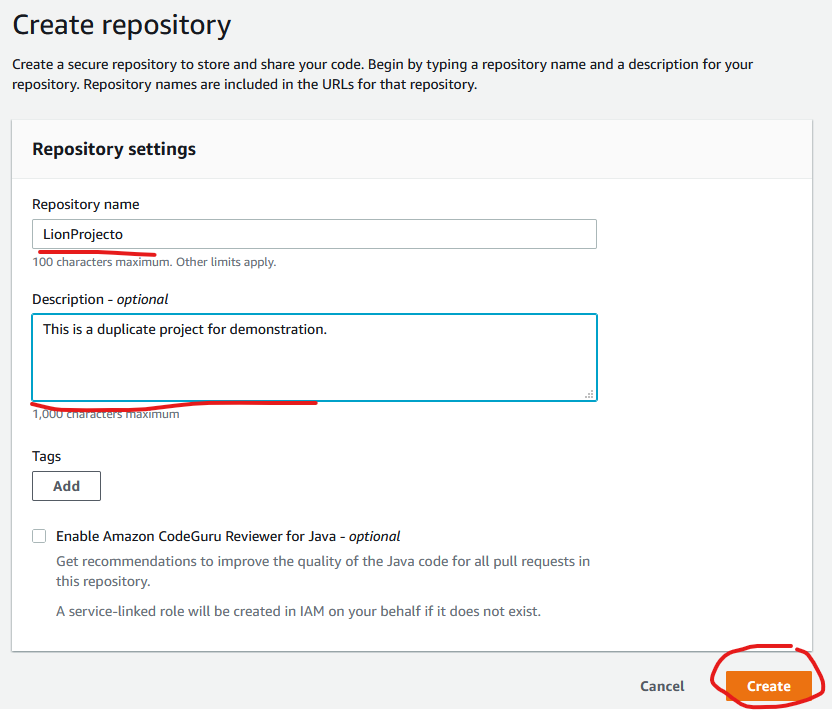
# Setting up CodeCommit in AWS

This section will address how to push codes onto CodeCommit, assuming the user has a project folder stored locally on their own machine and Git and Visual Studio Code installed as well.

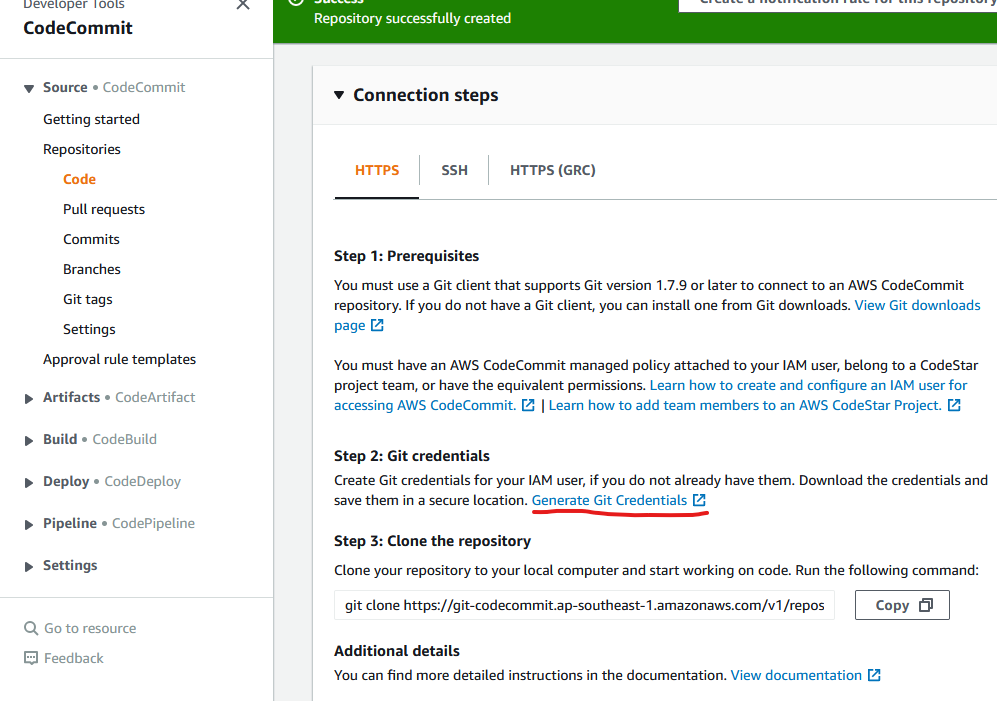
In AWS console, go to CodeCommit and create a new repository



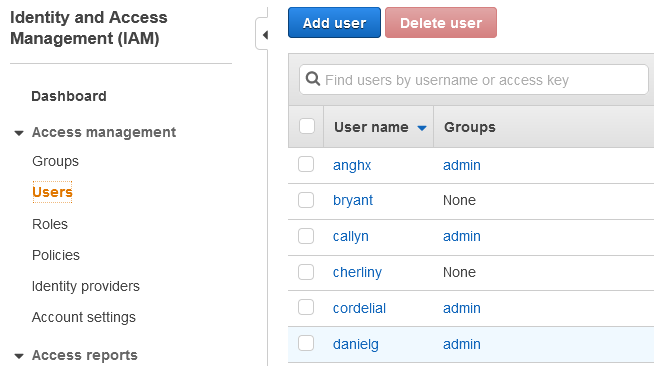
Give the repository a name and a description and click create



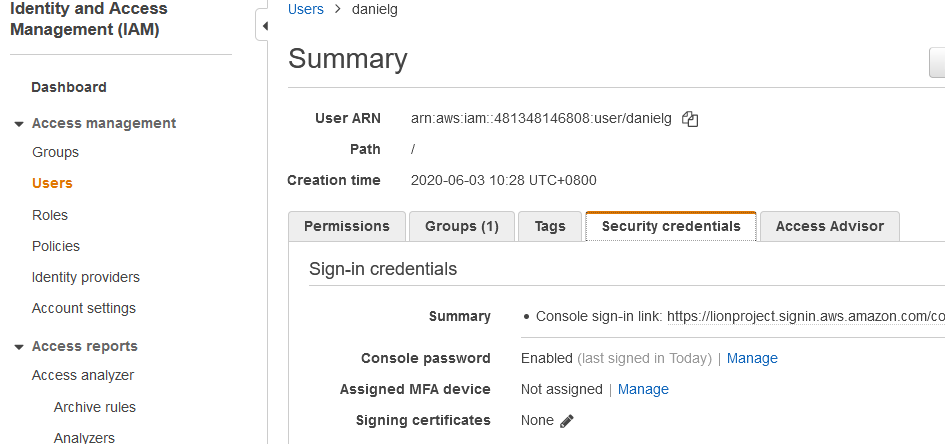
Download a set of git credentials by reading the link “Generate Git Credentials”

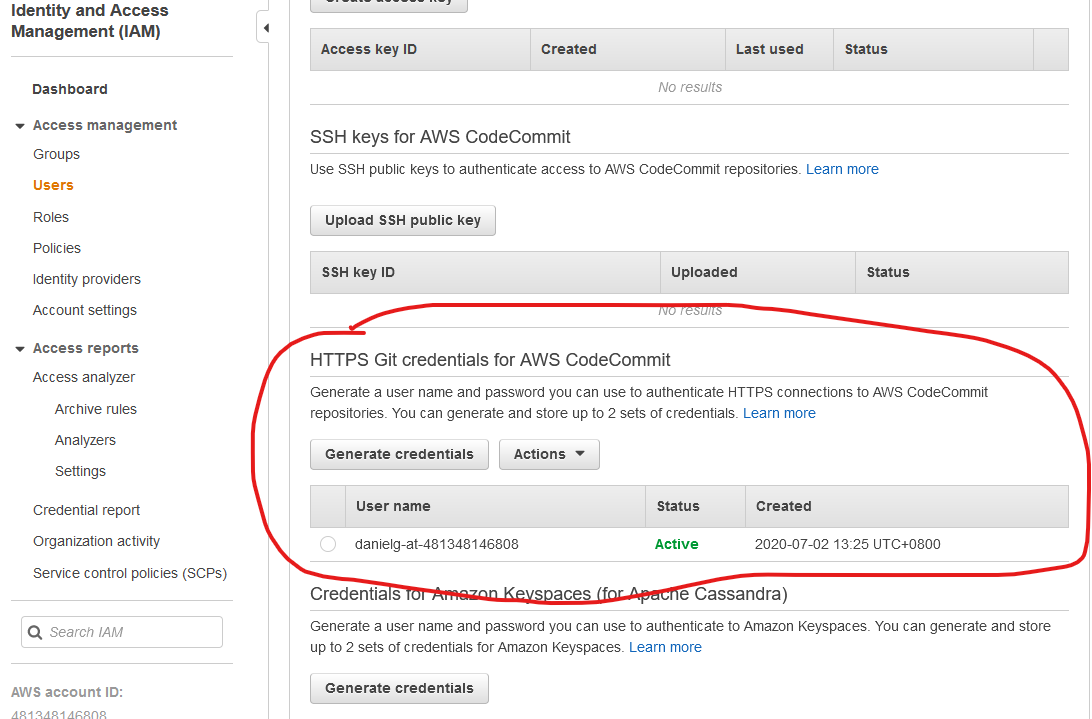


According to the link, the user has to go to IAM and access the user that he/she wants to configure for CodeCommit access, in this case we will use “danielg”.

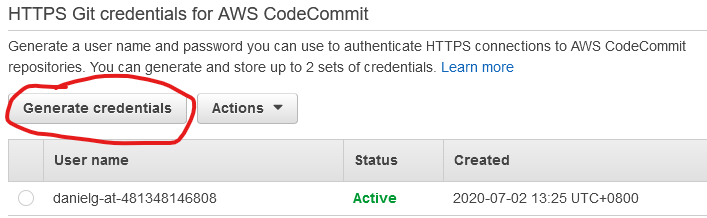


Go to Security credentials, and scroll down till you see “HTTPS Git Credentials for AWS CodeCommit”

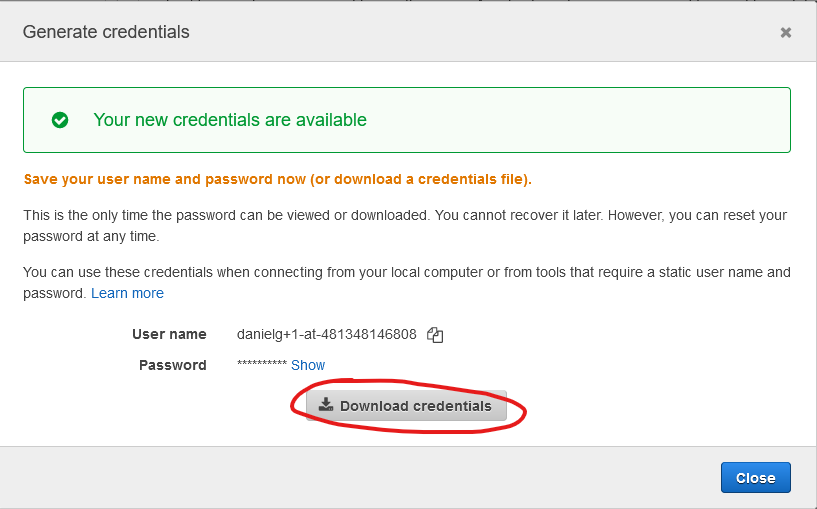




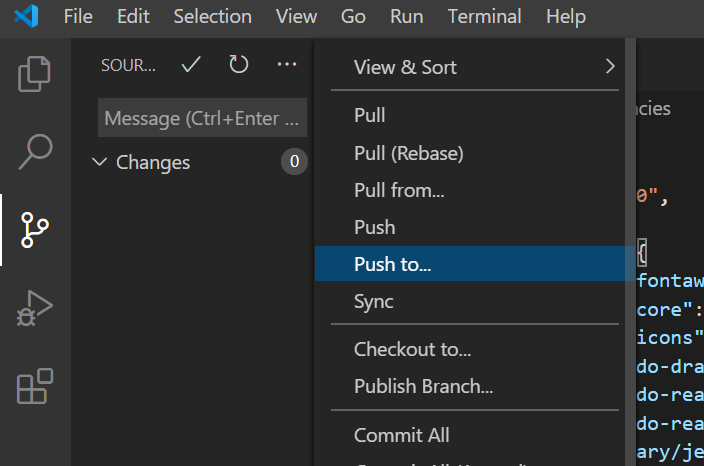
Click on “Generate credentials”



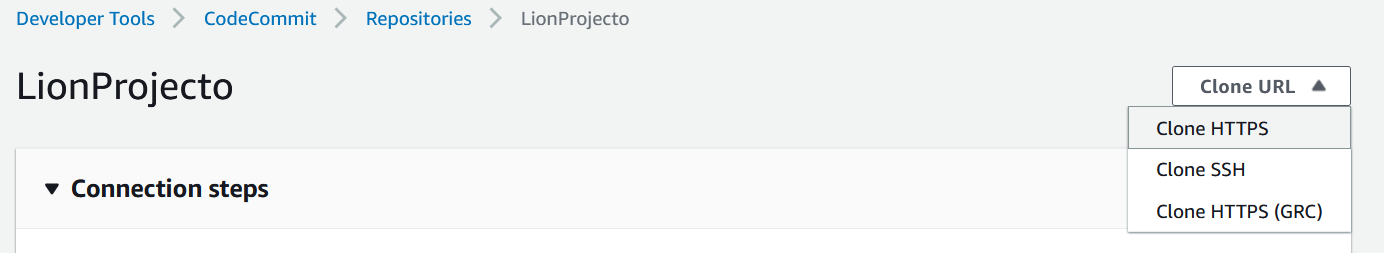
Then download the credentials given



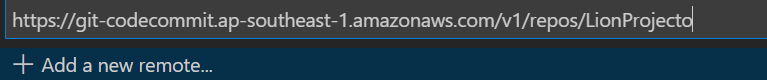
Open up the project folder in Visual Studios Code, go to Source Control > Views and more Actions > click Push to...



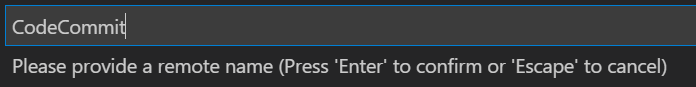
Go to CodeCommit, at the top right hand corner, click Clone URL, and select Clone HTTPS



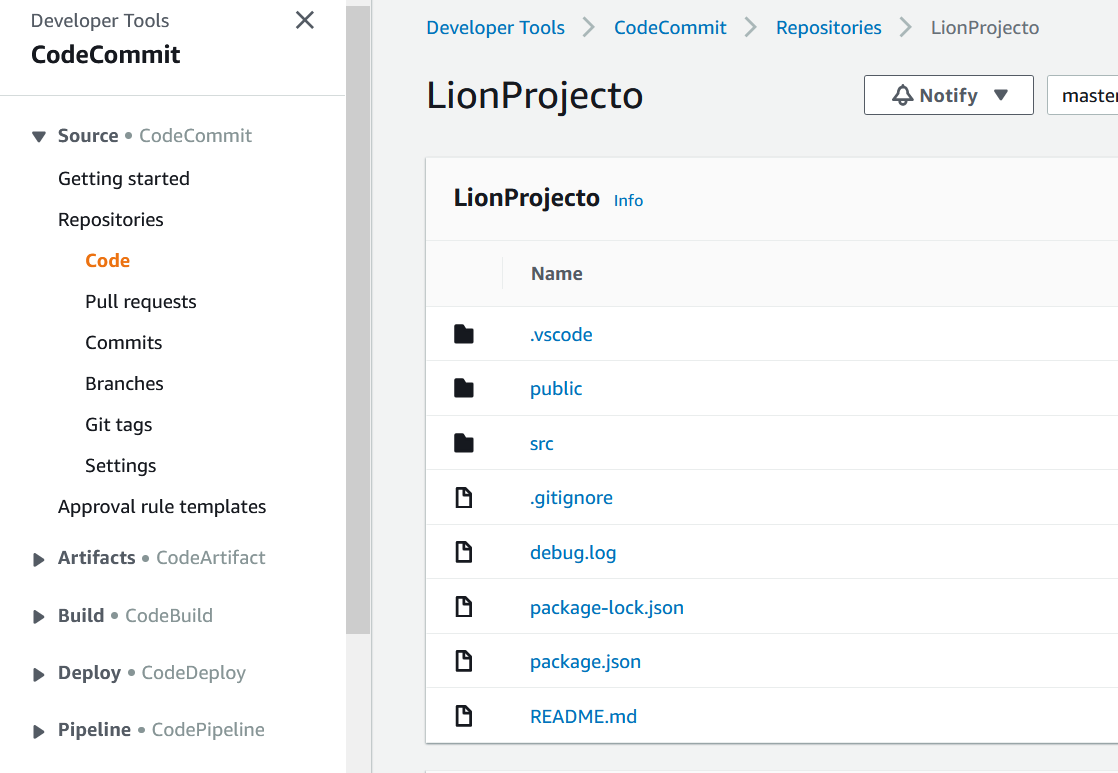
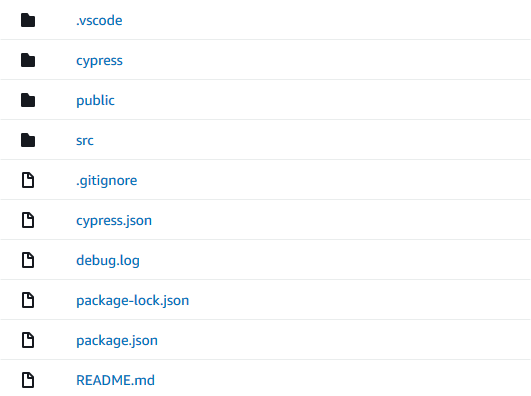
Paste URL in Visual Studios Code, then click Add a new remote



Give it a name, then click Enter



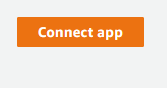
Now if you go back to CodeCommit, you should see all the codes inside



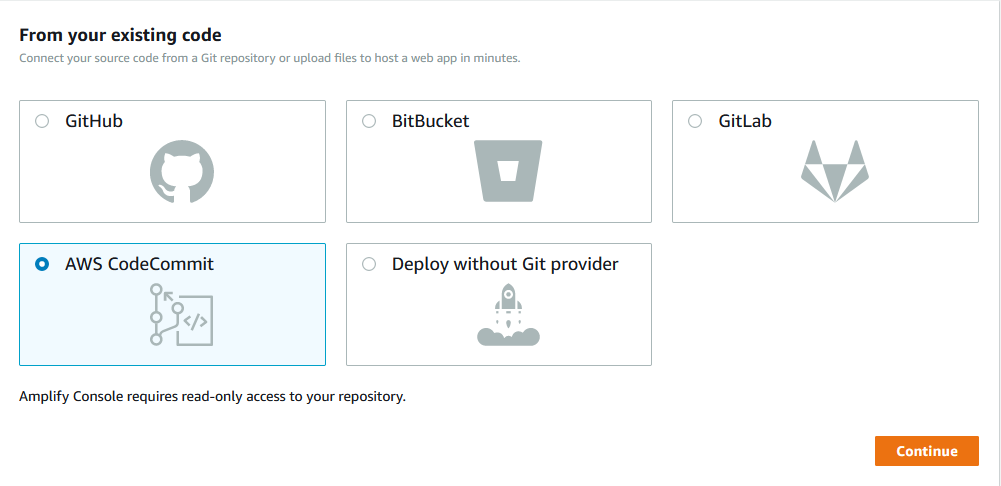
# Setting up Amplify using CodeCommit

This section describes how to host the website on CodeCommit using AWS Amplify.

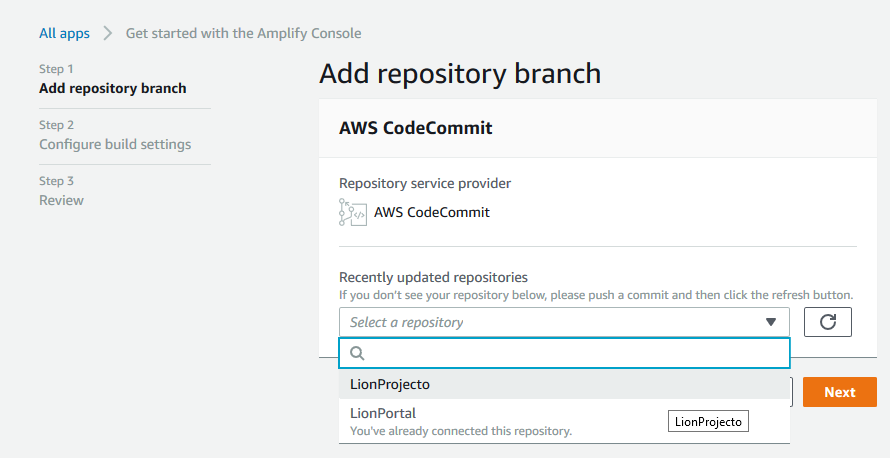
Go to Amplify on AWs and click this button



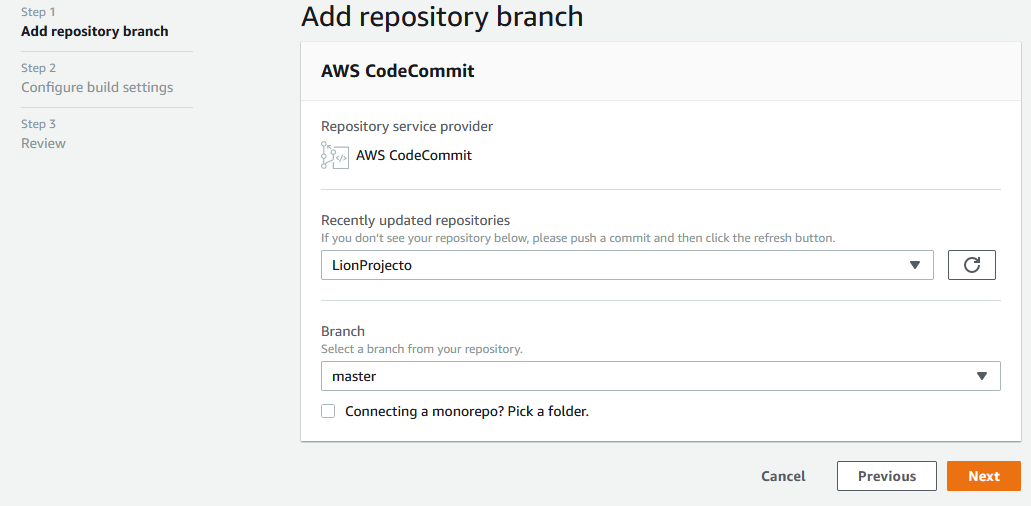
Select CodeCommit and click Continue



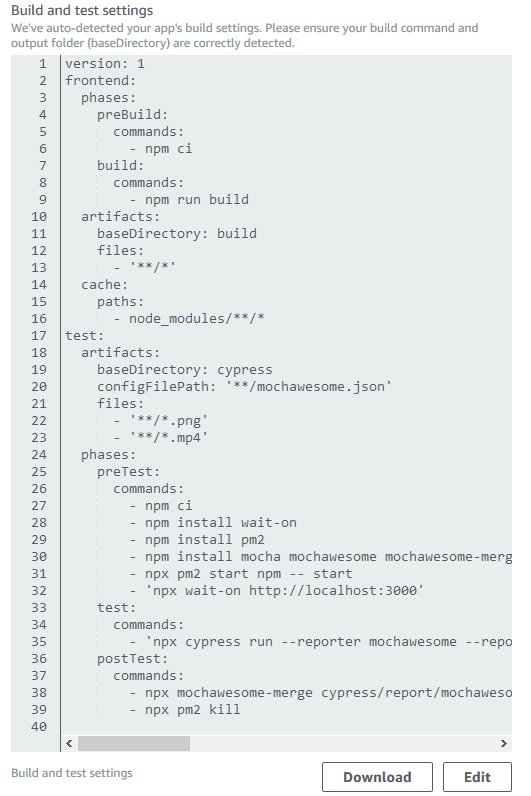
Under select a repository, select the one we just created



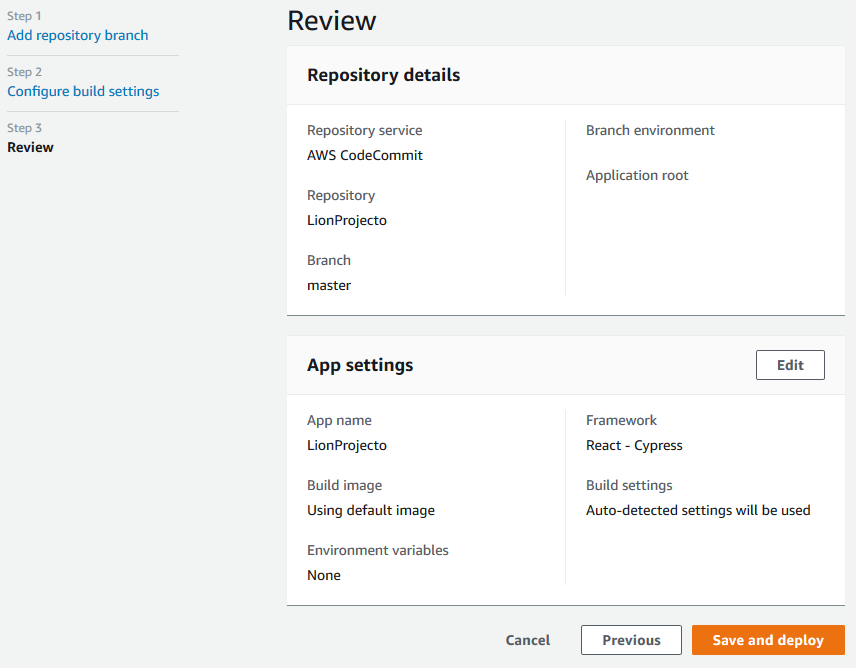
Leave the branch as master and click Next



Leave the build settings as default and click Next

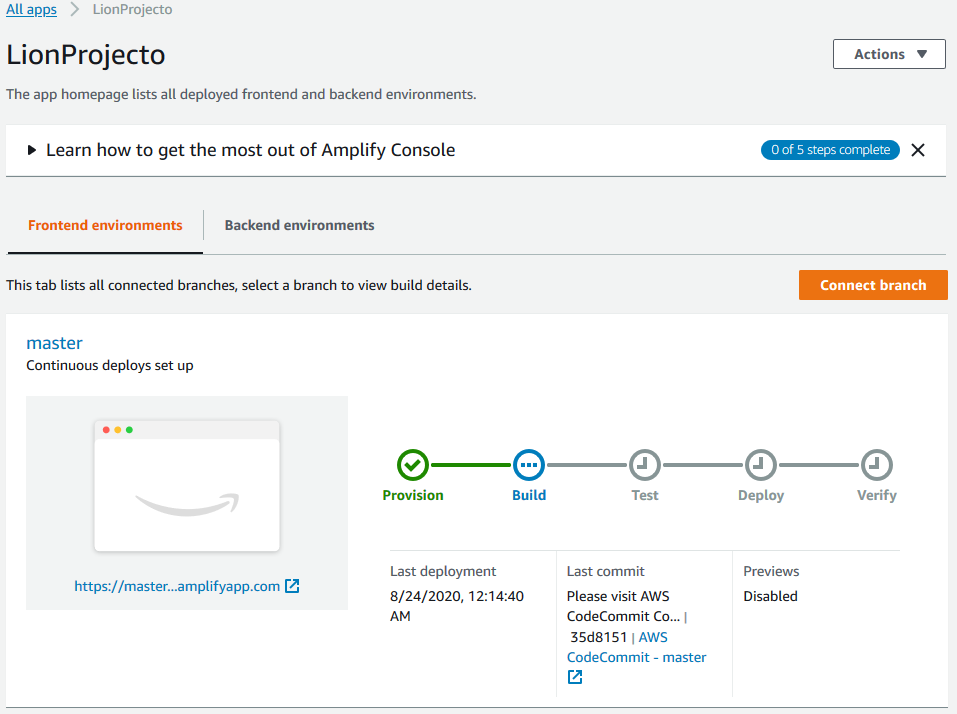


Leave the review as default as well and click Save and deploy



Wait for Amplify to finish building and deploying the app and the website should be up and running, on top of this, any changes being pushed to CodeCommit will trigger the whole build and deploy process again, which means Amplify has CI/CD capabilities automatically implemented. The whole process takes roughly 5~10 minutes.

The Cypress testing framework is also automatically configured and Amplify will run the tests before deploying that version. If any of the tests fails, Amplify will not deploy that version.



You can view the test’s results by clicking the Test icon, then clicking View log.

