Create and Deploy a Basic Blog Using AWS Amplify and React

AWS Amplify is a development platform that provides a set of tools and services to help front-end web and mobile developers build scalable and secure cloud-powered applications. It enables developers to quickly configure and deploy cloud resources, set up authentication and authorization, and integrate with other AWS services, all from a single unified dashboard.

With Amplify, developers can focus on building great user experiences while AWS handles the underlying infrastructure.

Goals:

Through this tutorial, you will be able to create your first Amplify project, add authentication to it, configure and use DynamoDB database, and add a custom domain to your application. At the end, you will deploy your app with AWS Amplify.

Part 1: Configure Amplify CLI

The Amplify Command Line Interface (CLI) is a tool that allows you to create, configure, and manage AWS Amplify projects and services from the command line.

1. To install the Amplify CLI, run the following command:

```
C:\Windows\system32\cmd.exe

C:\Users\Hira>npm install -g @aws-amplify/cli

added 26 packages, and audited 27 packages in 2m

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

2. Configure Amplify by running the following command:

```
C:\Windows\system32\cmd.exe

C:\Users\Hira>amplify configure
Follow these steps to set up access to your AWS account:

Sign in to your AWS administrator account:
https://console.aws.amazon.com/
Press Enter to continue

Specify the AWS Region
? region:  us-east-1
Follow the instructions at
https://docs.amplify.aws/cli/start/install/#configure-the-amplify-cli

to complete the user creation in the AWS console
https://console.aws.amazon.com/iamv2/home#/users/create
Press Enter to continue

Enter the access key of the newly created user:
? accessKeyId:  ********************
? secretAccessKey:  ****************************************
This would update/create the AWS Profile in your local machine
? Profile Name:  hira.dev

Successfully set up the new user.
```

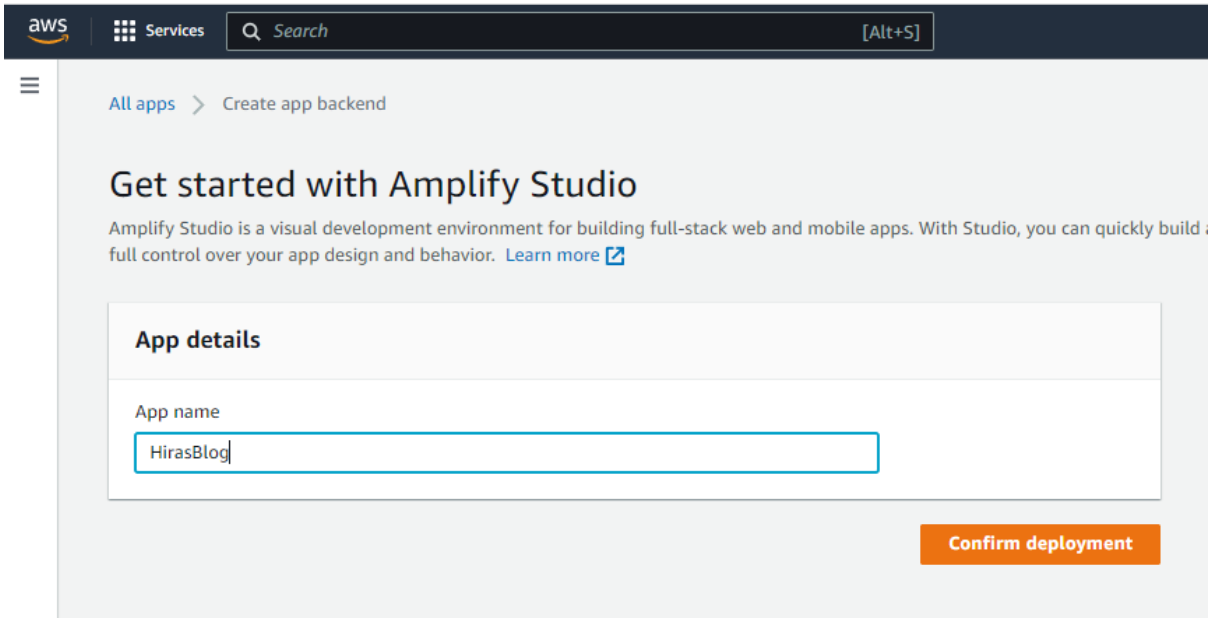
Amplify configure will open a browser and ask you to sign into the AWS Console.

Once you're signed in, Amplify CLI will ask you to create an IAM user. Create a user with AdministratorAccess to your account to provision AWS resources. Once the user is created, Amplify CLI will ask you to provide the accessKeyId and the secretAccessKey to connect Amplify CLI with your newly created IAM user.

After that, you will see the "Successfully set up the new user." Message.

Part 2: Create React App

Create an app for your application in the AWS Amplify Console, where you can view and manage your Amplify projects.



Part 3: Install and consume AWS Amplify Libraries

You need to install the Amplify JavaScript library aws-amplify and Amplify UI library for React @aws-amplify/ui-react (contains the React UI components).

1. Run the following command to install them:

```
D:\CloudComputing\amplify-app>npm install aws-amplify @aws-amplify/ui-react

up to date, audited 2654 packages in 8s

263 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

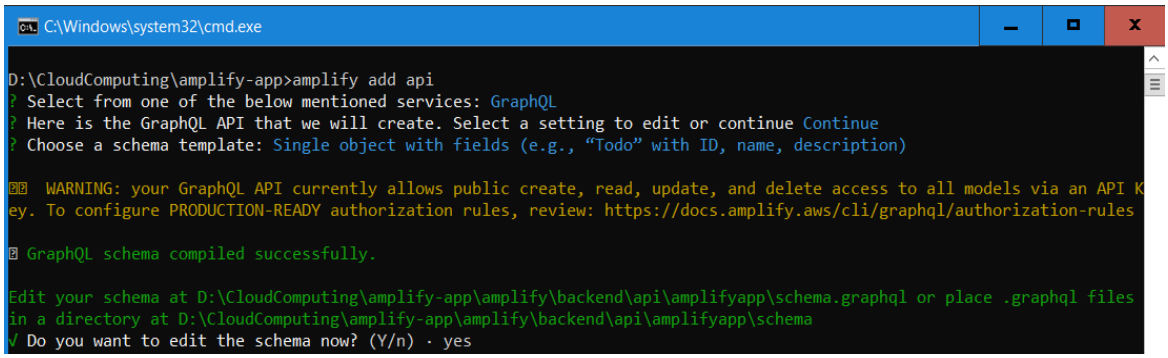
2. Now, to make your front end use these libraries, open the src/index.js file and replace its entire content with the following code to initialize your Amplify libraries:

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

import "@aws-amplify/ui-react/styles.css"; // Ensure React UI
libraries are styled correctly
import { Amplify } from 'aws-amplify'
import awsconfig from './aws-exports'
Amplify.configure(awsconfig) // Configures the Amplify libraries
with the cloud backend set up via the Amplify CLI

const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
reportWebVitals();
```

Part 4: Add a GraphQL API with Amplify

You are now going to add an API to the application. Amplify uses AWS AppSync and Amazon DynamoDB to power GraphQL APIs. AppSync is a managed GraphQL service that will take care of our API, and Amazon DynamoDB is a NoSQL database that will store the data our API will use.

1. To add the API, run the following command:



If you entered Yes to edit the schema now, your default editor should open to the file needed for the next section.

Part 5: Create a GraphQL Schema

A GraphQL schema is a representation of an object and its fields. You need to define your GraphQL schema, and Amplify will create the required DynamoDB table, and configure GraphQL to handle the reads, writes, updates and deletes for you.

1. Open the amplify/backend/api/amplifyapp/schema.graphql file and replace the contents with the following:

```
type BlogPost @model @auth(rules: [{ allow: owner }]) {
  id: ID!
  title: String!
  text: String!
}
```

Go back to command line and continue the execution. Once successful, you will see this message.

```
√ Do you want to edit the schema now? (Y/n) · yes
Edit the file in your editor: D:\CloudComputing\amplify-app\amplify\backend\api\amplifyapp\schema.graphql
ⓘ Successfully added resource amplifyapp locally

ⓘ Some next steps:
"amplify push" will build all your local backend resources and provision it in the cloud
"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and prov
ision it in the cloud


√ Successfully pulled backend environment dev from the cloud.

    Current Environment: dev

| Category | Resource name | Operation | Provider plugin |
|---|---|---|---|
| Api | amplifyapp | Create | awscloudformation |
```

Part 6: Deploy Application

You are now ready to deploy your Amplify web application.

1. Run the following command:

```
C:\Windows\system32\cmd.exe

D:\CloudComputing\amplify-app>amplify push
√ Successfully pulled backend environment dev from the cloud.

    Current Environment: dev

| Category | Resource name | Operation | Provider plugin |
|---|---|---|---|
| Api | amplifyapp | Create | awscloudformation |

√ Are you sure you want to continue? (Y/n) · yes
Cognito UserPool configuration
Using service: Cognito, provided by: awscloudformation

 The current configured provider is Amazon Cognito.

 Do you want to use the default authentication and security configuration? Default configuration
 Warning: you will not be able to edit these selections.
 How do you want users to be able to sign in? Username
 Do you want to configure advanced settings? No, I am done.
ⓘ Successfully added auth resource amplifyapp043416a2 locally
```

Accept all the default values to configure auth with a simple username and password combination. After confirming the auth settings, select the default values to the follow up questions when prompted.

Part 7: Update Front End to Use API

To use the new API and auth backend you just deployed, update the src/App.js file by replacing the contents with the following code:

```
import './App.css';
import { createBlogPost, deleteBlogPost } from './graphql/mutations'
import { listBlogPosts } from './graphql/queries'
import { withAuthenticator, Button, Text, Flex, Heading } from
"@aws-amplify/ui-react";
```

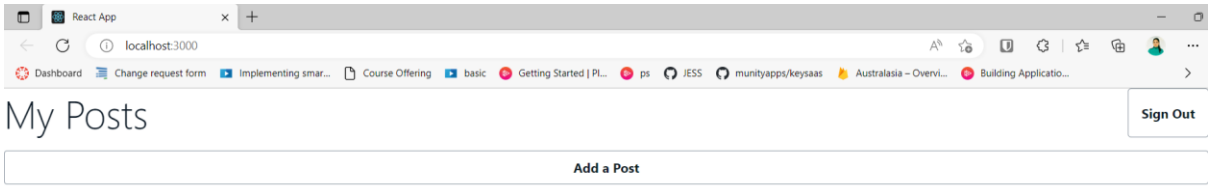
```
import { useCallback, useEffect, useState } from 'react';
import { API } from 'aws-amplify';
function App({ signOut }) {
  const [blogPosts, setblogPosts] = useState([])

  const fetchBlogPosts = useCallback(async () => {
    const result = await API.graphql({
      query: listBlogPosts,
      authMode: 'AMAZON_COGNITO_USER_POOLS'
    })
    setblogPosts(result.data.listBlogPosts.items)
  }, [setblogPosts])

  const handleCreateBlogPost = useCallback(async () => {
    await API.graphql({
      query: createBlogPost,
      variables: { input: { title: window.prompt("New Post Title"),
text: window.prompt("New Post Text") } },
      authMode: 'AMAZON_COGNITO_USER_POOLS'
    })
    fetchBlogPosts()
  }, [fetchBlogPosts])

  const handleDeleteBlogPosts = useCallback(async (id) => {
    await API.graphql({
      query: deleteBlogPost,
      variables: { input: { id: id } },
      authMode: 'AMAZON_COGNITO_USER_POOLS'
    })
    fetchBlogPosts()
  }, [fetchBlogPosts])
  useEffect(() => {
    fetchBlogPosts()
  }, [fetchBlogPosts])
  return (
    <Flex direction={"column"}>
      <Flex justifyContent={'space-between'}>
        <Heading level={1}>My Posts</Heading>
        <Button onClick={signOut}>Sign Out</Button>
      </Flex>
      {blogPosts.map(post => <Flex alignItems={'center'}>
        <Text>{post.title}</Text>
        <Text>{post.text}</Text>
        <Button onClick={() =>
handleDeleteBlogPosts(post.id)}>Remove</Button>
      </Flex>)}
      <Button onClick={handleCreateBlogPost}>Add a Post</Button>
    </Flex>
  );
}
export default withAuthenticator(App);
```

Part 8: Test your application

Now you can run your app locally to test it. Run the following command.

```
D:\CloudComputing\amplify-app>npm start

> amplify-app@0.1.0 start
> react-scripts start
```

You should be able to see the app running on localhost.



Part 9: Store project on GitHub

Create a private repo, and push the code you have created so far to that repo.

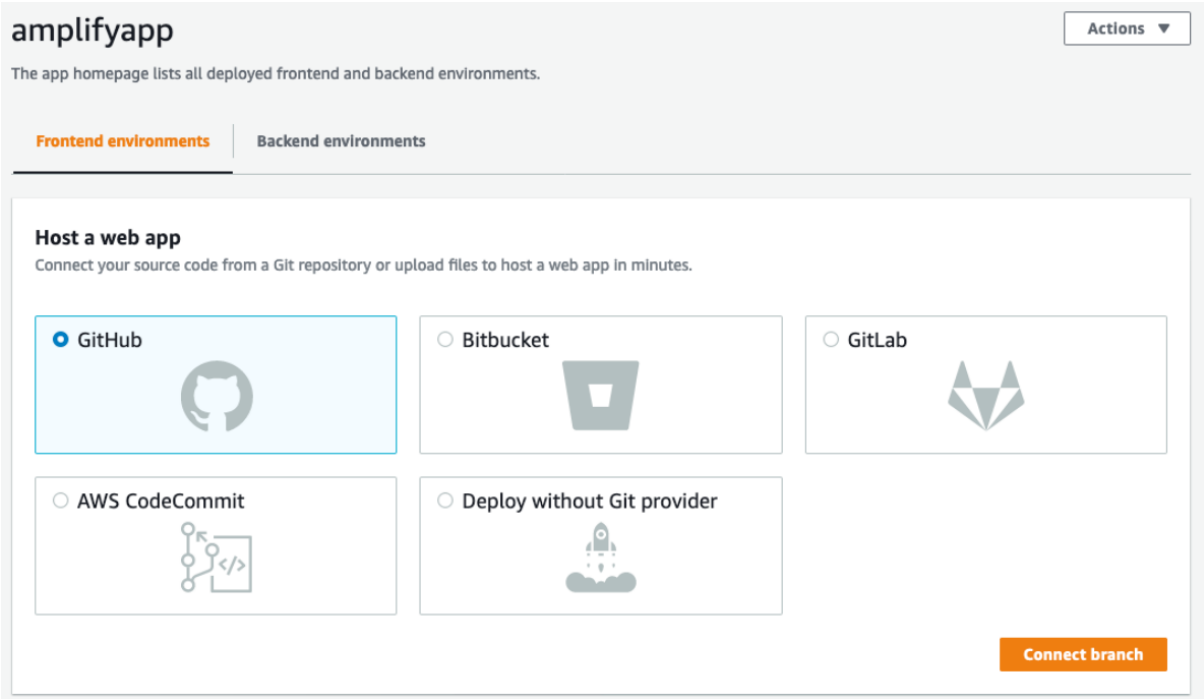
Part 10: Setup Continuous Deployment

To configure Amplify to deploy your code, you need to connect it with your GitHub account. This is done via the AWS Console as it needs to generate a GitHub token to access your private repo, and store it in your AWS account.

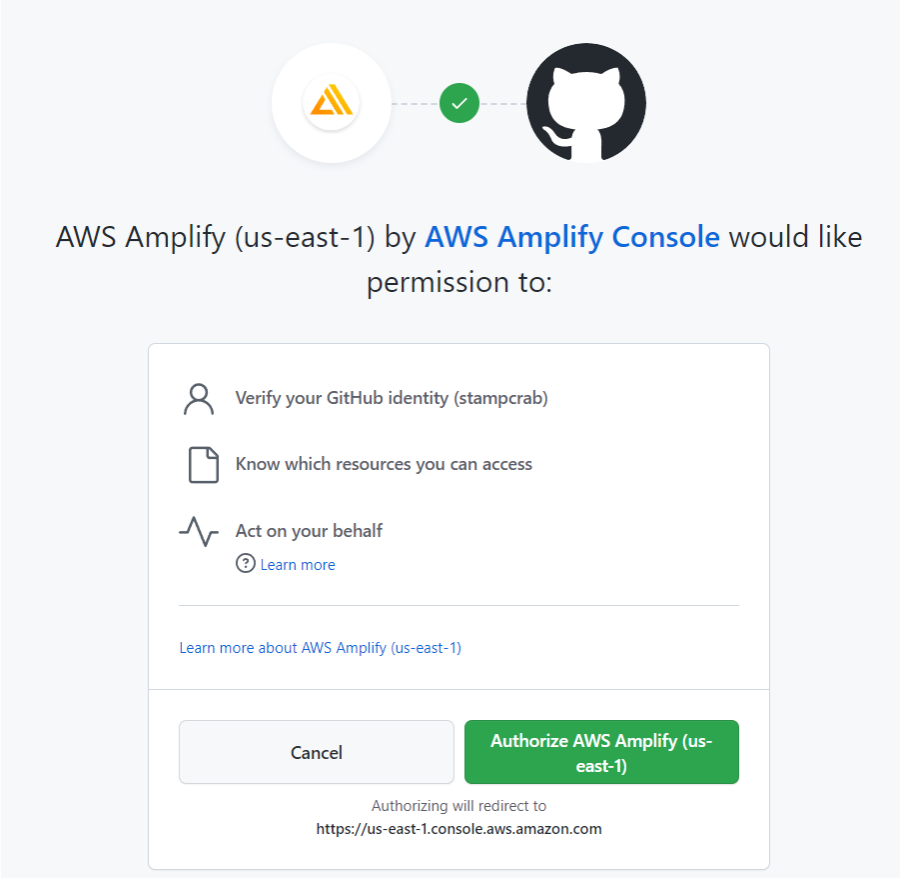
1. To do this, from the amplify-app directory, run amplify add hosting. Amplify will present a list of questions about the hosting, please select the options shown below:

```
amplify add hosting

? Select the plugin module to execute (Use arrow keys)
❯ Hosting with Amplify Console (Managed hosting with custom domains, Continuous deployment)
  Amazon CloudFront and S3
? Choose a type
❯ Continuous deployment (Git-based deployments)
  Manual deployment
  Learn more
```

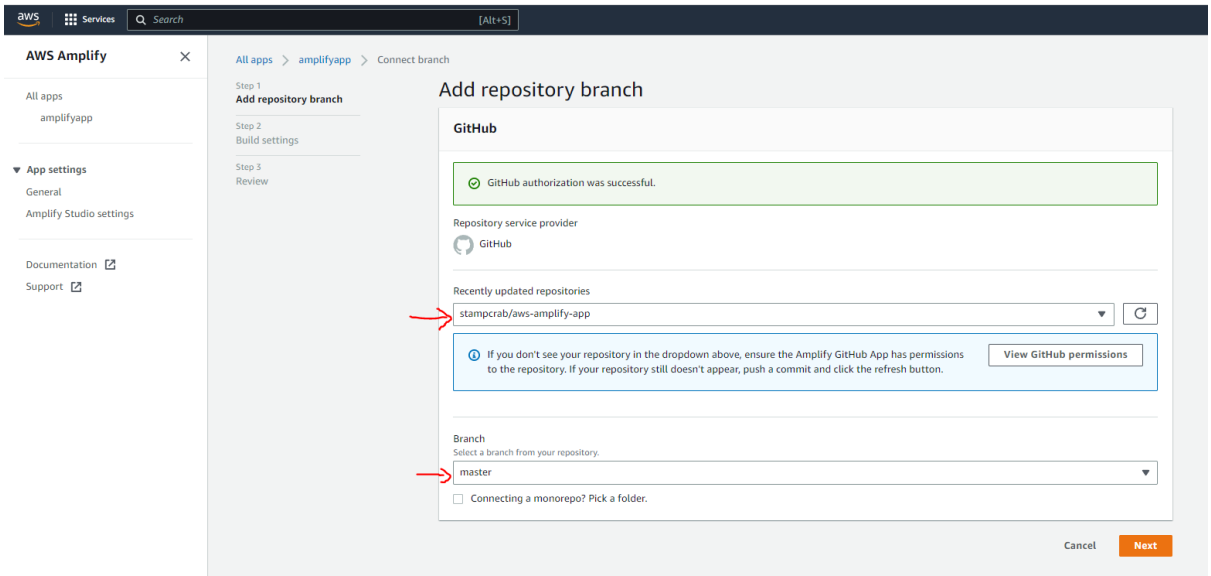
2. This will launch launch a new window in your browser, and open the Amplify Console for you to configure hosting on your project. From the app page on the Amplify Console, click on the Frontend environments tab, select GitHub and click on the Connect branch button.



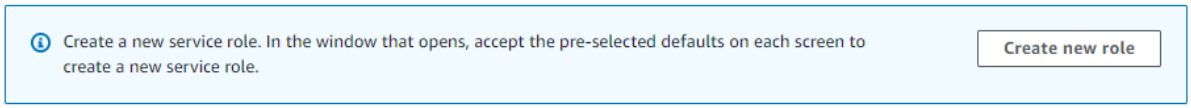
3. This will redirect you to GitHub. You now will give the Amplify Console access to your GitHub account so it can deploy the source code you are hosting there. To do so, click on the green Authorize aws-amplify-console button.



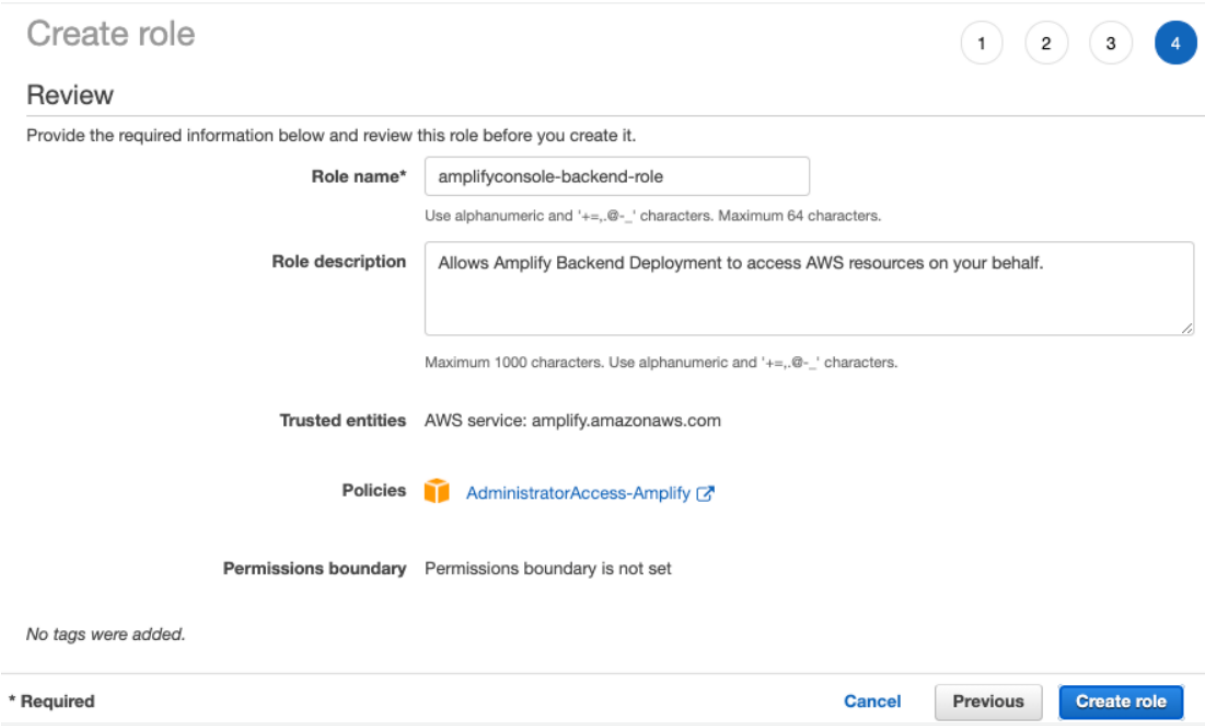
4. You will now connect your GitHub repo to your Amplify application. From the Add repository branch page, select the amplify-web-app repo and the main branch. Then, click on Next.



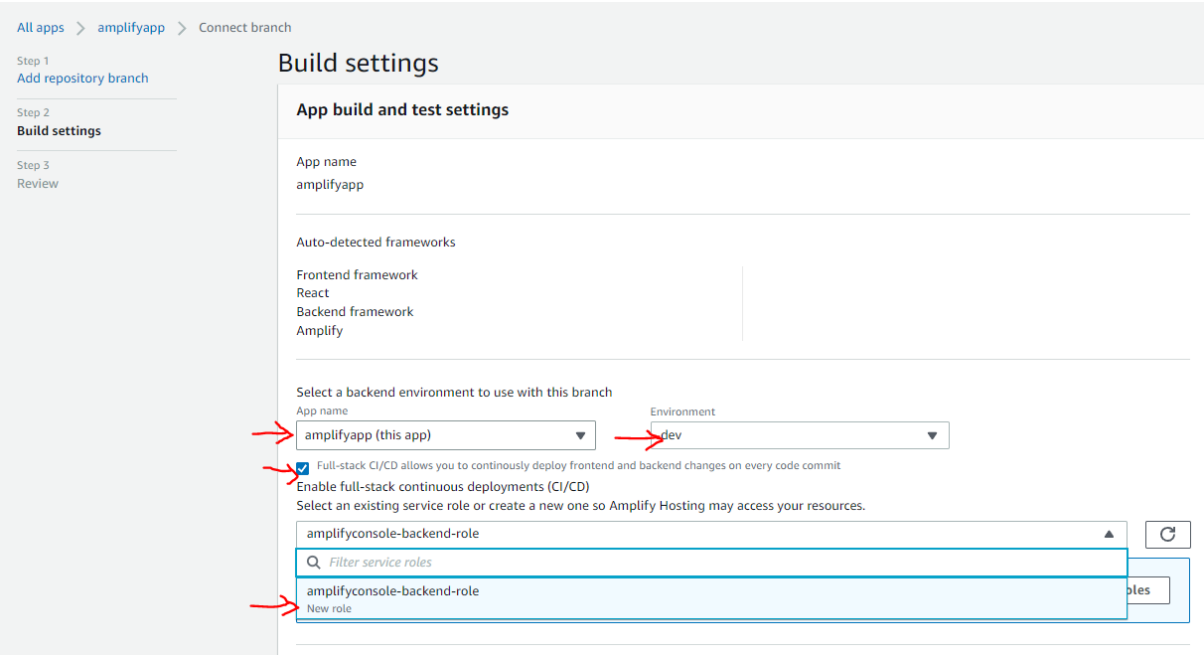
5. The next step is to configure the build settings for your application. On the Configure build settings page, click on the Create new role button inside the blue box.



6. click on Create Role:



7. Once created, you can close this tab in your browser, and return to the Configure build settings page, click on Refresh existing roles and then select the role you just created from the dropdown menu. Then, select dev from the Environment dropdown menu - this is the environment you created when configuring Amplify on your project after running amplify init. Leave the remaining settings with their default values, and click on Next.



8. Review the values you configured, and click on Save and deploy. Amplify will now start to automatically deploy your React application on source repository changes.