Hands-on tutorials

Build a Basic Web Application

Build a Basic Web Application: Hands-on tutorials

Table of Contents

Build a Basic Web Application

AWS experience	Beginner
Minimum time to complete	35 minutes
Cost to complete	Free Tier eligible
Services used	AWS Amplify AWS AppSync AWS Lambda Amazon DynamoDB
Last updated	July 12, 2024

Overview

In this tutorial, you will learn to create a simple full-stack web application using AWS Amplify. Throughout this tutorial, you will build and host a React application on AWS, use Amplify to add authentication, data, and a serverless function to capture the signed-up user's email and save it in the database. Then, you will implement a frontend for your app that integrates with your cloud resources.

What you will accomplish

In this tutorial, you will:

- **Host:** Build and deploy a React application on the AWS global content delivery network (CDN).
- **Authenticate:** Add authentication to your app to enable sign-in and sign-out functionality.
- **Database:** Integrate a real-time API, database, and a serverless function.
- **Function:** Implement a lambda function that is triggered when a user signs up to the App.

Prerequisites

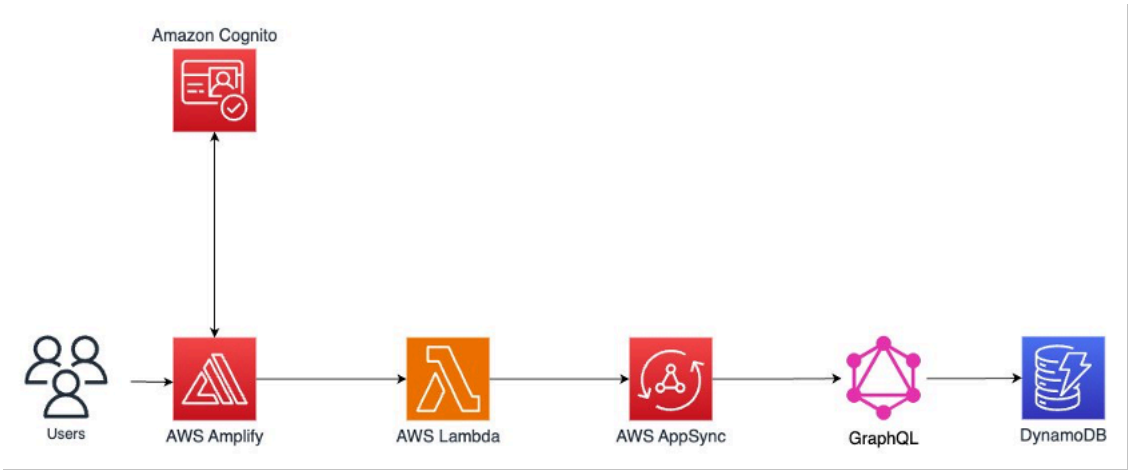
Before starting this tutorial, you will need:

- An **AWS account**: if you don't already have one follow the Setup Your Environment tutorial.
- **Configure** your **AWS profile** for local development.
- **Installed** on your environment: Nodejs and npm.
- **Familiarity** with Git and a Github account.

Application architecture

The following diagram provides a visual representation of the services used in this tutorial and how they are connected. This application uses AWS Amplify, GraphQL API, AWS Lambda, and Amazon DynamoDB.

As you go through the tutorial, you will learn about the services in detail and find resources that will help you get up to speed with them.



Tasks

This tutorial is divided into six tasks. You must complete each task in order before moving on to the next one.

1. Task 1: Create a Web App (5 minutes): Deploy static resources for your web application using the AWS Amplify Console.
2. Task 2: Build a Serverless Function (5 minutes): Build a serverless function using AWS Lambda.

3. Task 3: Create a Data Table (10 minutes): Persist data in an Amazon DynamoDB table.
4. Task 4: Link a Serverless Function to a Web App to Web App (5 minutes): Deploy your serverless function with API Gateway.
5. Task 5: Add Interactivity to Your Web App (5 minutes): Modify your web app to invoke your API.
6. Task 6: Clean Up Resources (5 minutes): Clean up resources used in this tutorial.

You will be building this web application using the AWS Management Console accessible directly from your browser.

Task 1: Create a Web App

Minimum time to complete	5 minutes
Services used	AWS Amplify
Requires	A GitHub account GitHub SSH connection Nodejs and npm
Get help	Troubleshooting Amplify How Amplify works

Overview

AWS Amplify offers a Git-based CI/CD workflow for building, deploying, and hosting single-page web applications or static sites with backends. When connected to a Git repository, Amplify determines the build settings for both the frontend framework and any configured backend resources, and automatically deploys updates with every code commit.

In this task, you will start by creating a new React application and pushing it to a GitHub repository. You will then connect the repository to AWS Amplify web hosting and deploy it to a globally available content delivery network (CDN) hosted on an **amplifyapp.com** domain.

Key concepts

React Application: React is a JavaScript library that enables developers to quickly build performant single-page applications.

Git: Git is a version control system that allows developers to store files, maintain and update relationships between files and directories, and track versions and changes to the files.

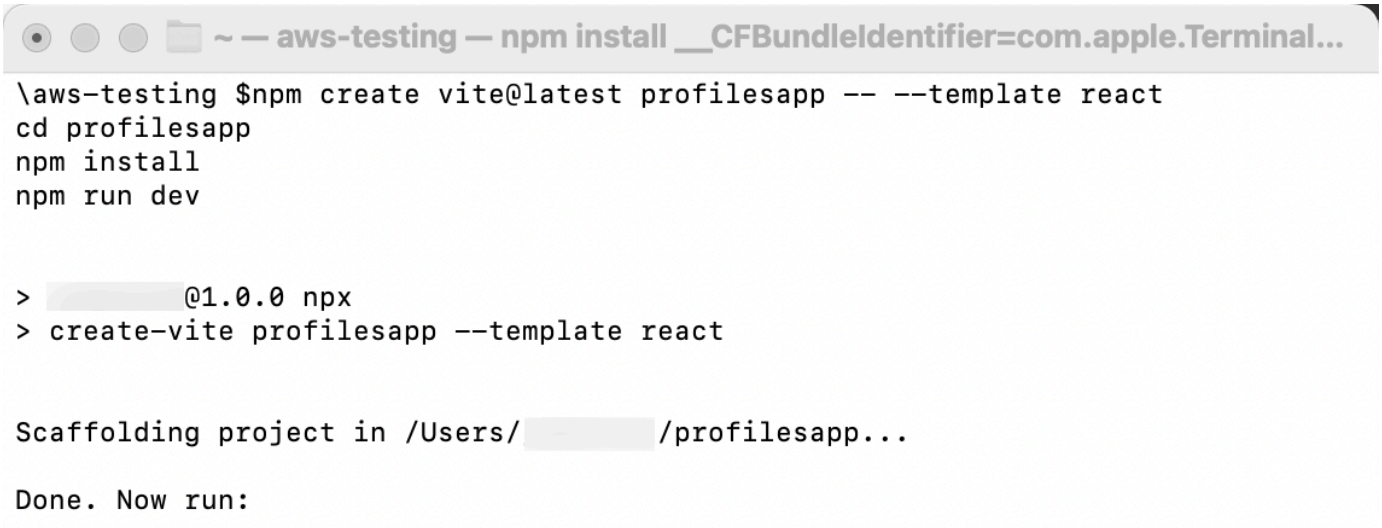
Implementation

Step 1: Create a new React application

1. Create the app

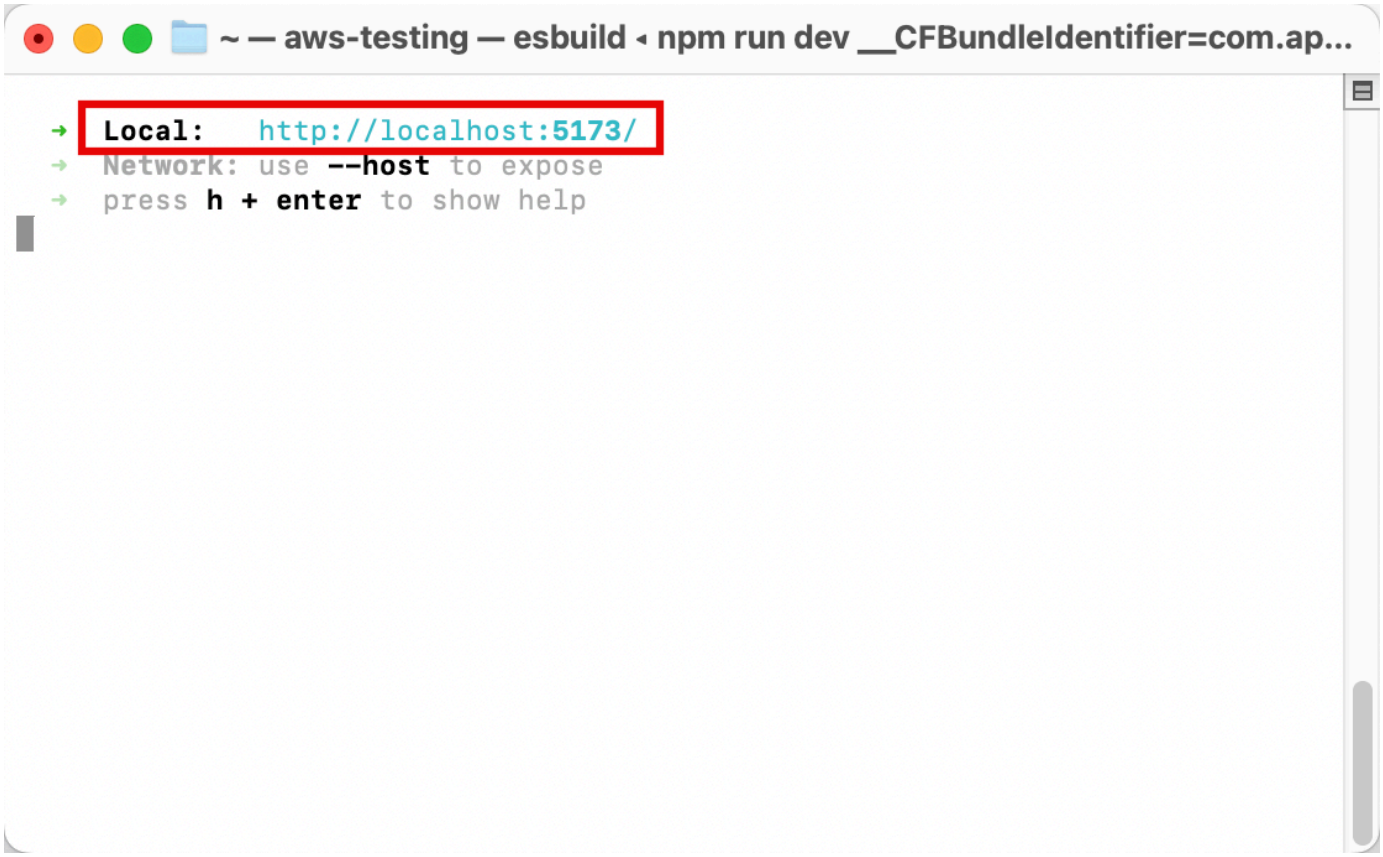
   In a new terminal or command line window, **run** the following command to use Vite to create a React application:

```
npm create vite@latest profilesapp -- --template react
cd profilesapp
npm install
npm run dev
```



2. View the app

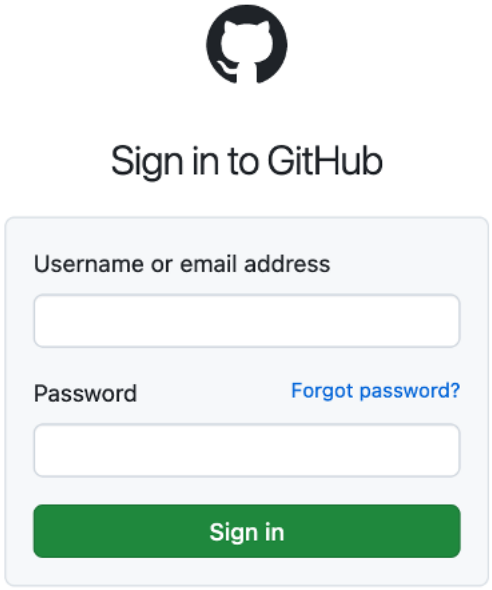
   In the terminal window, select and open the **Local link** to view the Vite + React application.



Step 2: Install the Amplify packages

1. Open GitHub

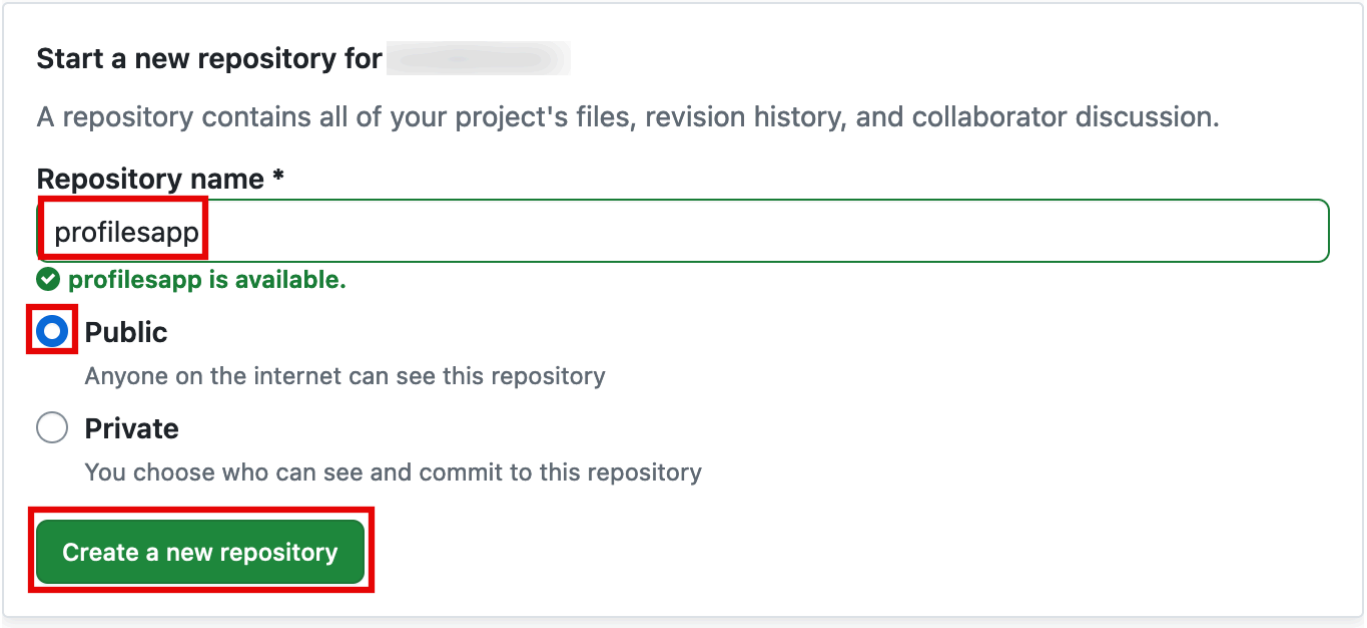
   Sign in to GitHub at https://github.com/.



2. Start repository

   In the Start a new repository section, make the following selections:

   - For **Repository name**, enter **profilesapp**, and choose the **Public** radio button.
   - Then select, **Create a new repository**.



3. Push the app to GitHub

Open a new terminal window, navigate to your projects root folder (profilesapp), and run the following commands to initialize a git and push of the application to the new GitHub repo:

> **Note**
>
> Replace the SSH GitHub URL in the command with your GitHub URL.

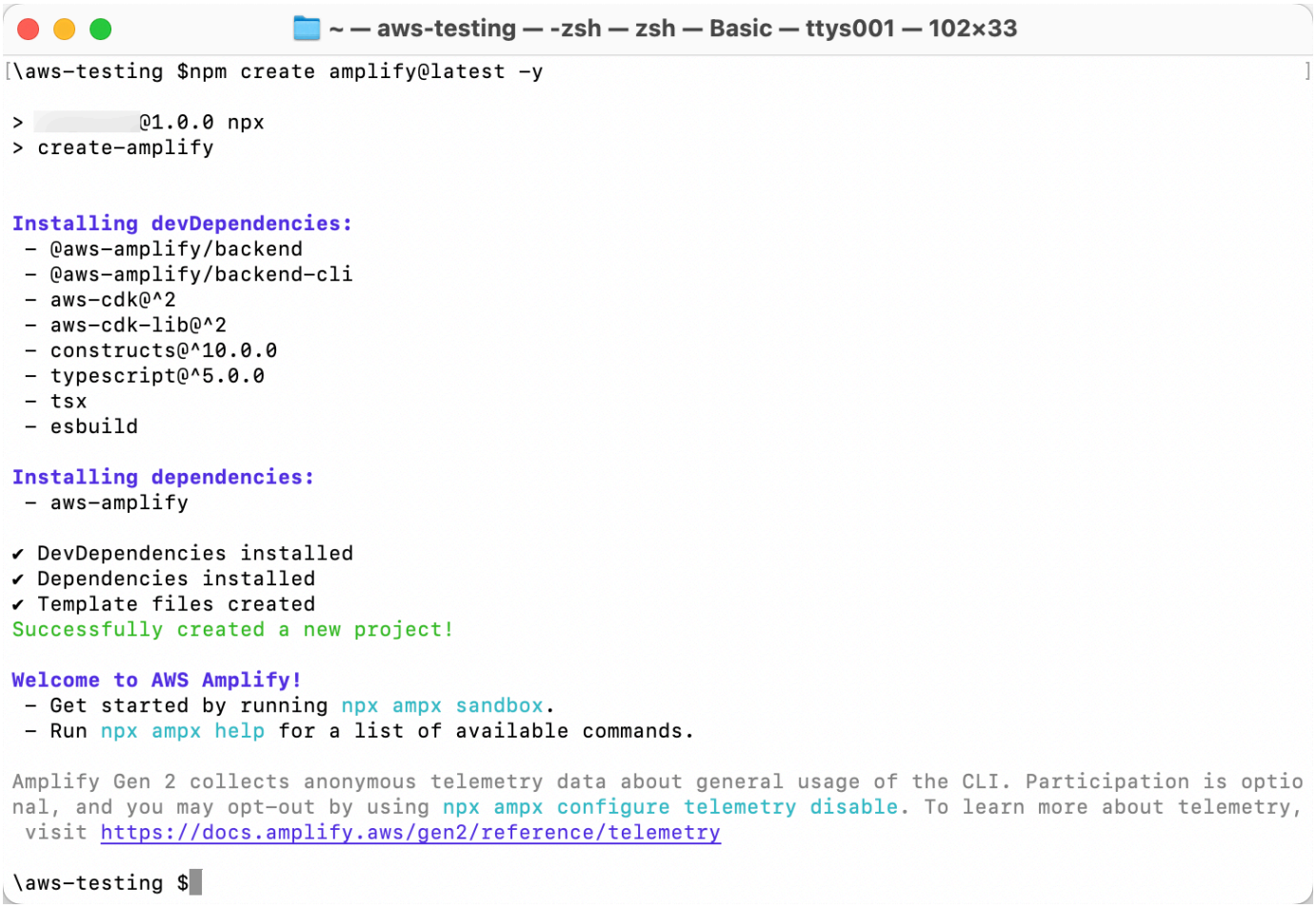
```
git init
git add .
git commit -m "first commit"
git remote add origin git@github.com:<your-username>/profilesapp.git
git branch -M main
git push -u origin main
```

Step 3: Initialize a GitHub repository

1. Create an Amplify project

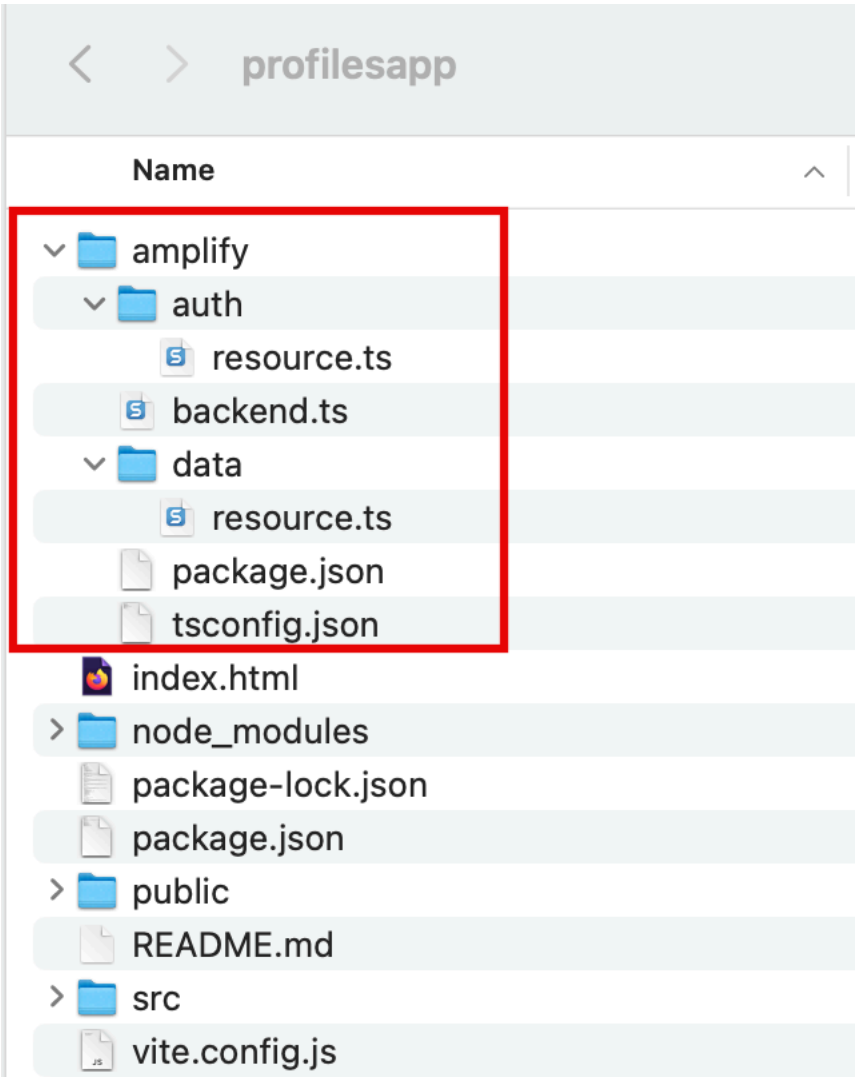
   Open a new terminal window, **navigate** to your app's root folder (**profilesapp**), and **run** the following command:

```
npm create amplify@latest -y
```



2. View the project directory

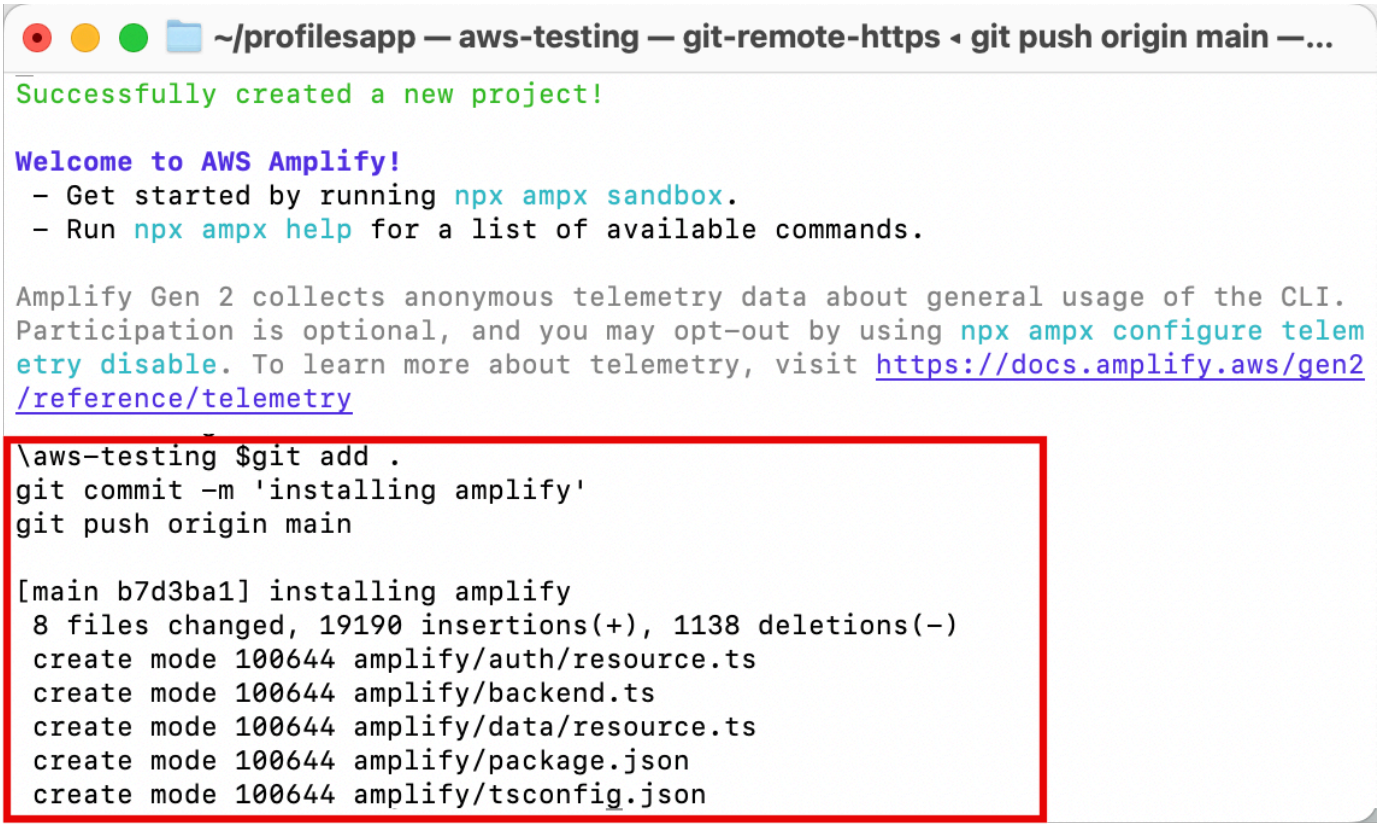
   Running the previous command will scaffold a lightweight Amplify project in the app's directory.



3. Push the changes to the repository

   In your terminal window, **run** the following command to push the changes to GitHub:

```
git add .
git commit -m 'installing amplify'
git push origin main
```



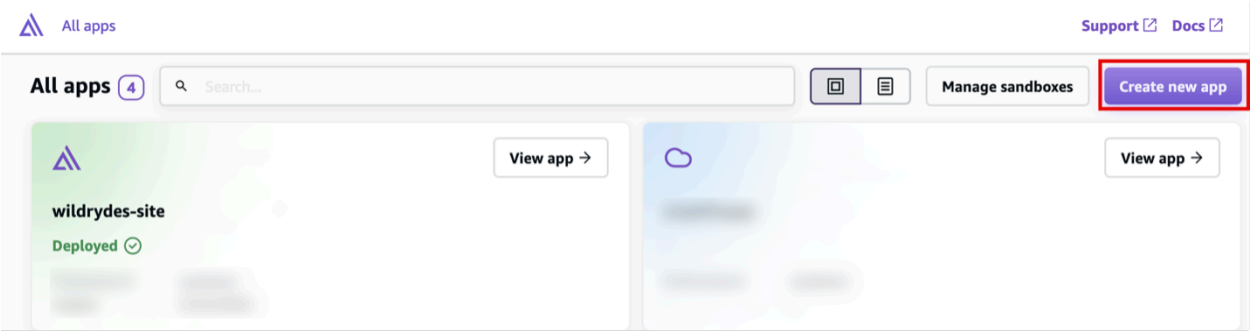
Step 4: Deploy your app with AWS Amplify

In this step, you will connect the GitHub repository you just created to AWS Amplify. This will enable you to build and deploy your app on AWS.

1. Create a new Amplify app

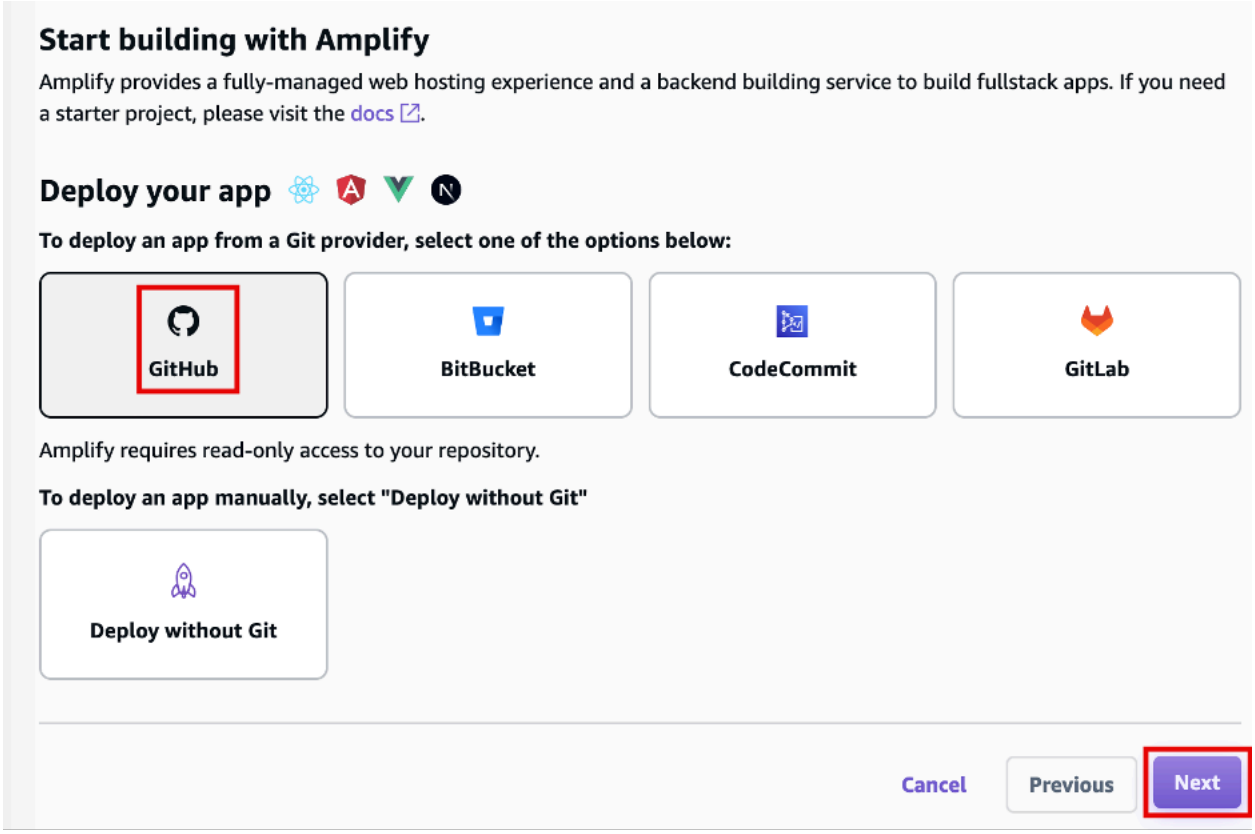
   Sign in to the AWS Management console in a new browser window, and open the AWS Amplify console at https://console.aws.amazon.com/amplify/apps.

   Choose **Create new app**.



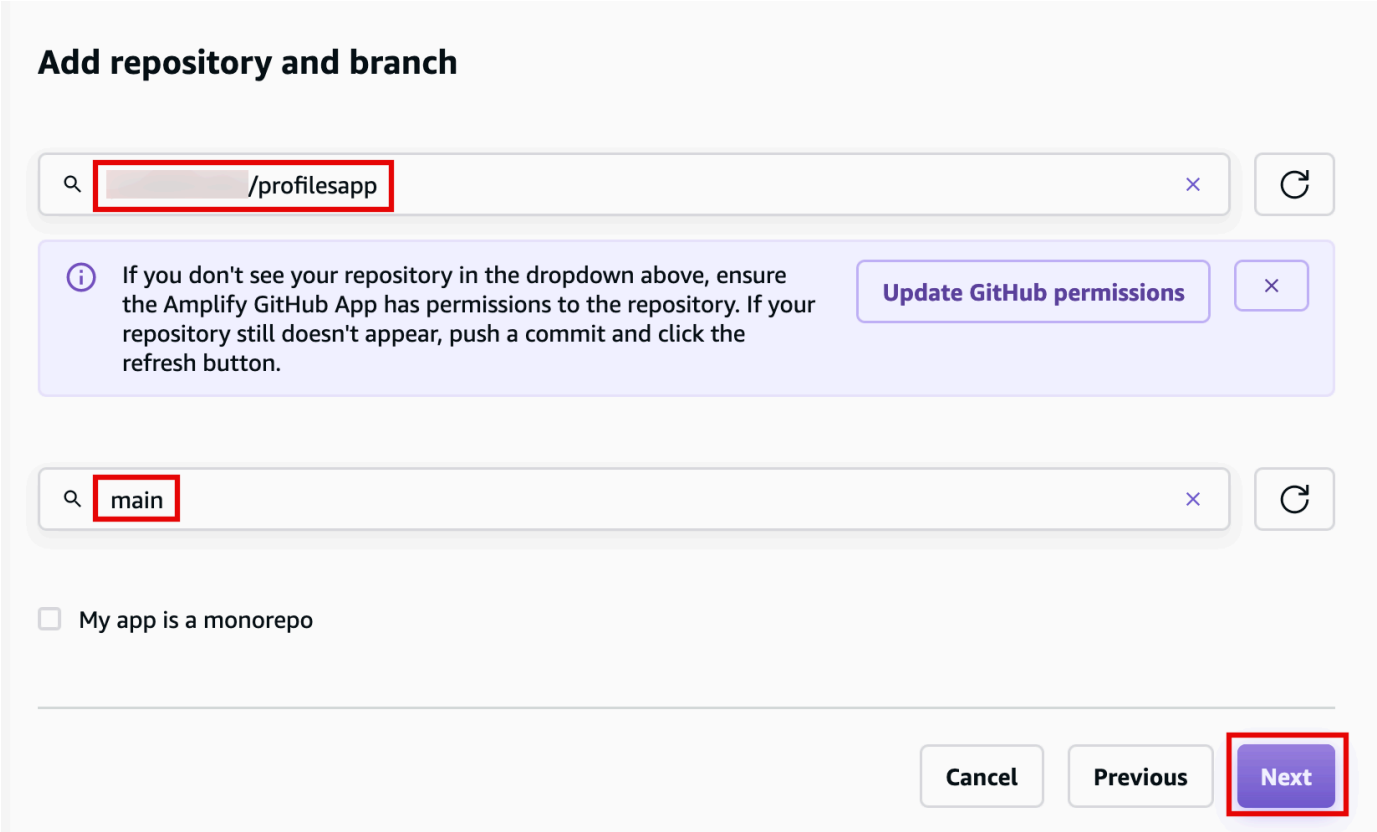
2. Choose GitHub as your Git provider

   On the **Start building with Amplify** page, for **Deploy your app**, select **GitHub**, and select **Next**.



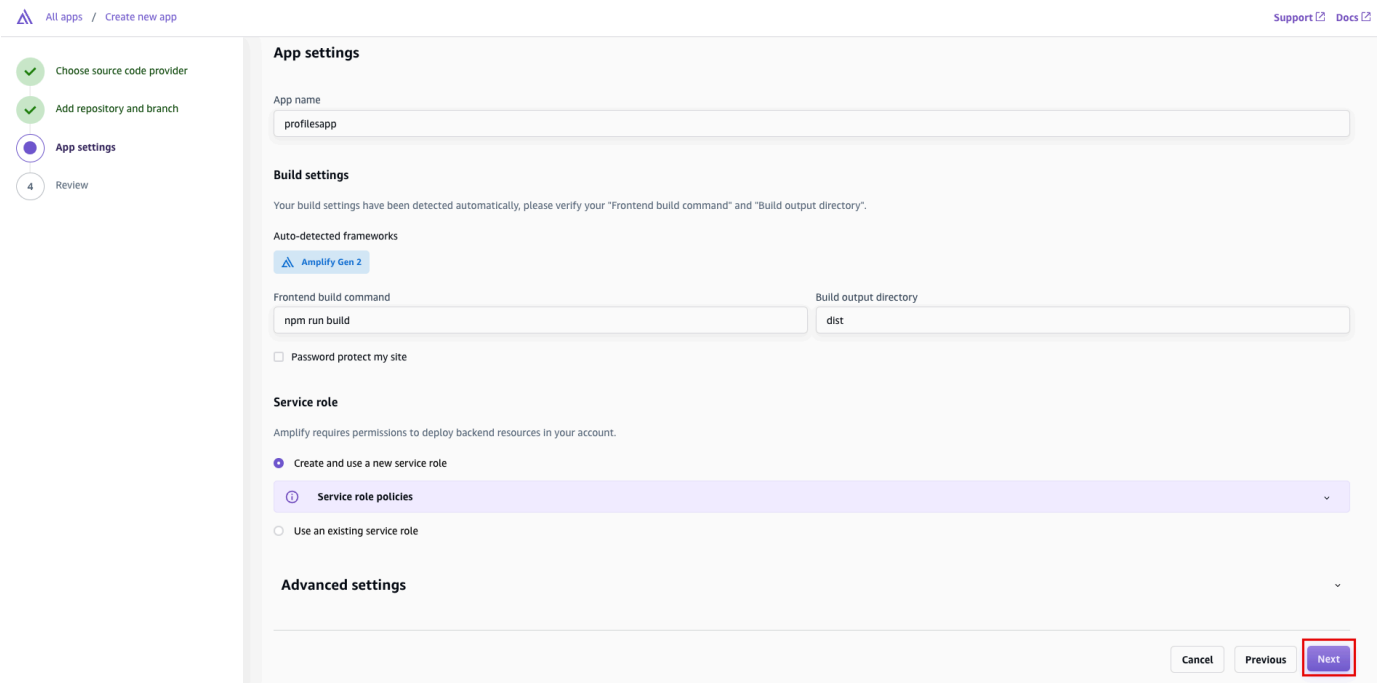
3. Add repository and branch

   When prompted, **authenticate** with GitHub. You will be automatically redirected back to the Amplify console. Choose the **repository** and **main branch** you created earlier. Then select **Next**.



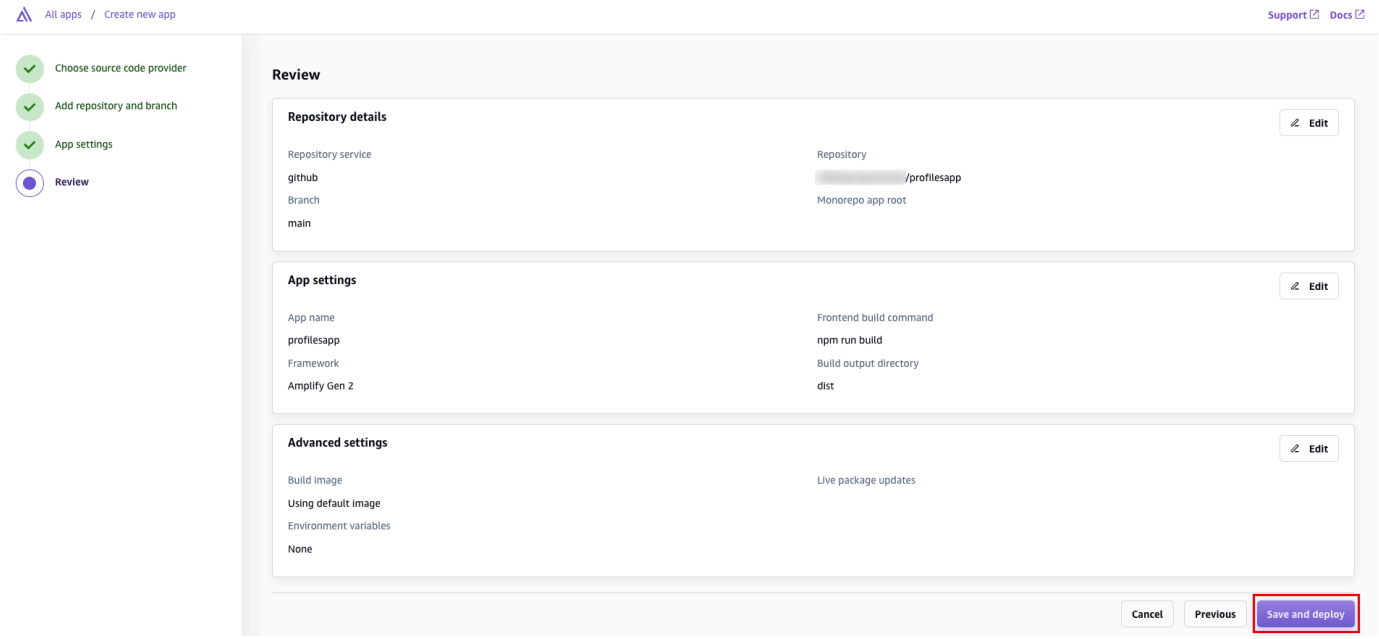
4. Review build settings

   Leave the default **build settings**, and select **Next**.



5. Deploy the app

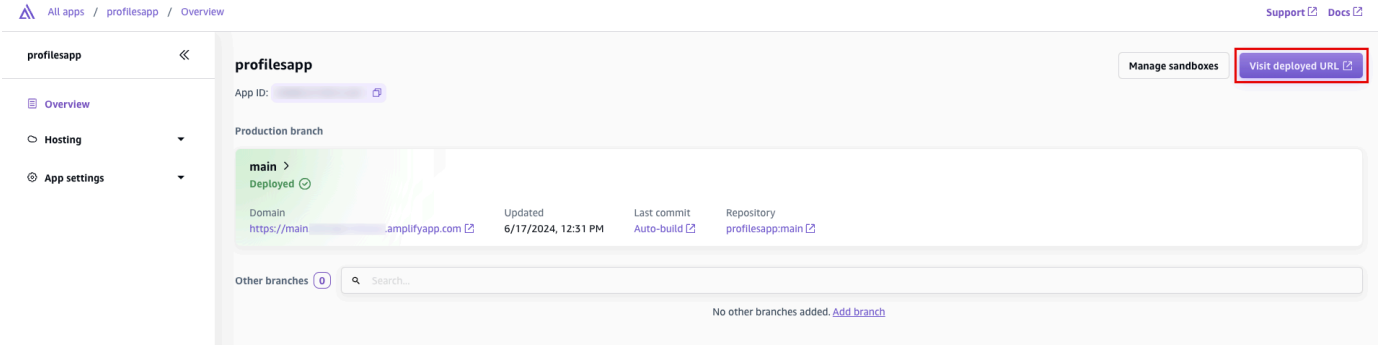
Review the inputs selected, and choose **Save and deploy**.



6. Monitor build status

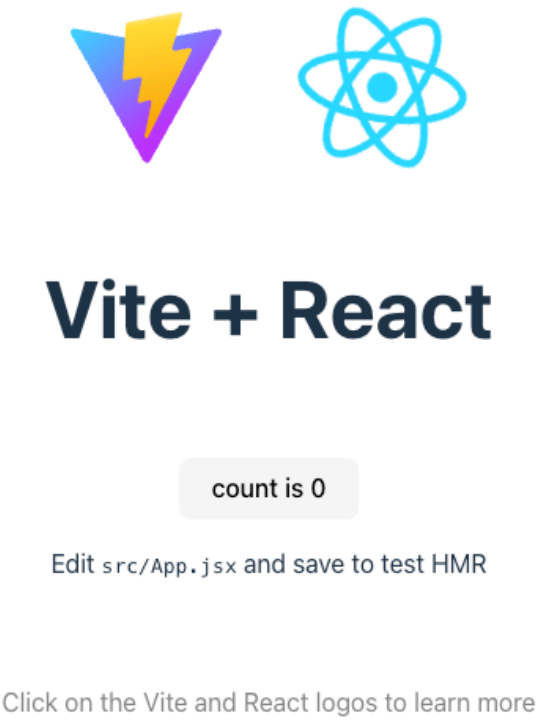
AWS Amplify will now **build** your source code and **deploy** your app at **https://...amplifyapp.com**, and on every git push your deployment instance will update.

It may take up to 5 minutes to deploy your app.



7. View the deployed app

Once the build completes, select the **Visit deployed URL** button to see your web app up and running live.



Conclusion

You have deployed a React application in the AWS Cloud by integrating with GitHub and using AWS Amplify. With AWS Amplify, you can continuously deploy your application in the Cloud and host it on a globally available CDN.

Task 2: Build a Serverless Function

Time to complete	5 minutes
Services used	AWS Lambda AWS Amplify
Requires	A text editor. Here are a few free ones: • Atom • Notepad++ • Sublime • Vim • Visual Studio Code
Get help	Learn more about functions

Overview

Now that you have a React web app, you will use AWS Amplify and AWS Lambda to configure a serverless function. This function is invoked after a signed-up user confirms their user account. AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, making it the fastest way to turn an idea into a modern, production, serverless applications.

Key concepts

Serverless function: Piece of code that will be executed by a compute service, on demand.

Implementation

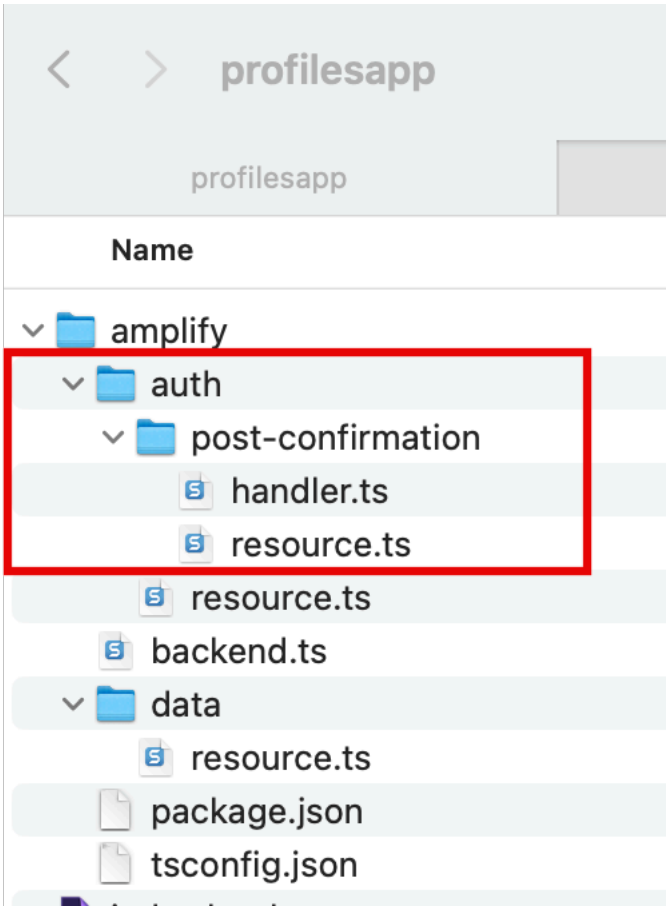
Step 1: Set up an Amplify Function

1. Create files

On your local machine, navigate to the **profilesapp/amplify/auth** folder.

Create a new folder inside the **amplify/auth** folder and name it **post-confirmation**.

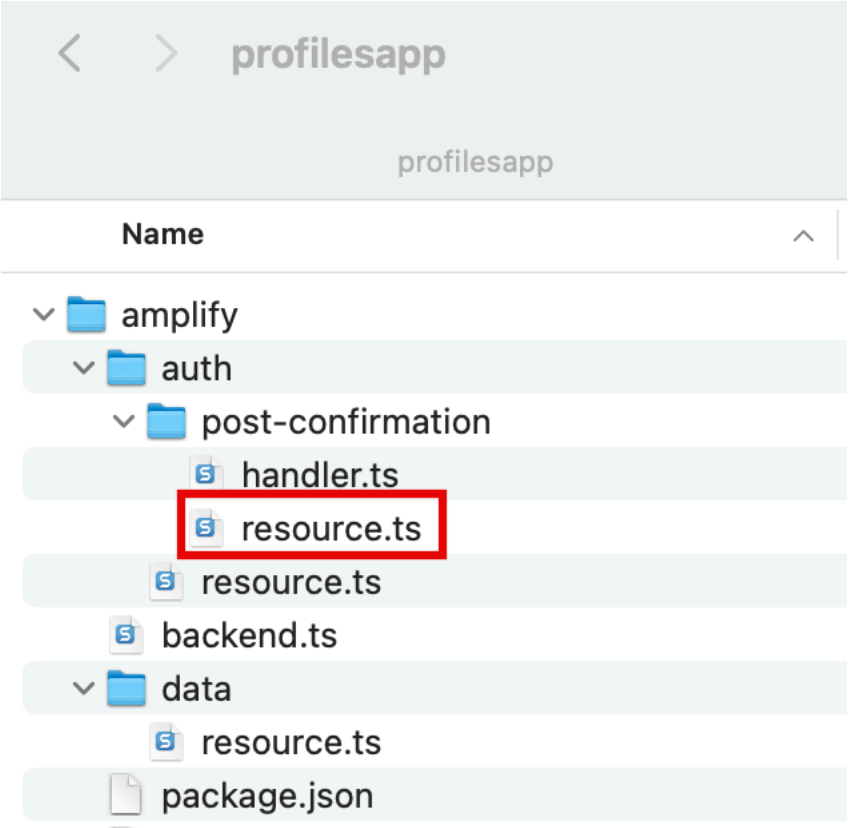
Create the files named **resource.ts** and **handler.ts** inside the folder.



2. Update the resources file

Update the **amplify/auth/post-confirmation/resource.ts** file with the following code to define the **postConfirmation** function. Then, **save** the file.

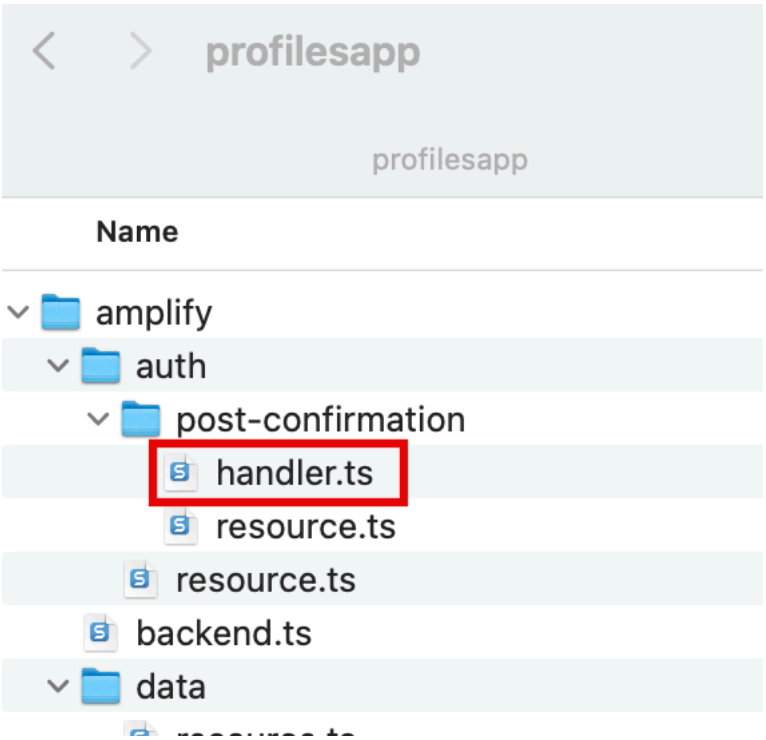
```
import { defineFunction } from '@aws-amplify/backend';
export const postConfirmation = defineFunction({
  name: 'post-confirmation',
});
```



3. Update the handler file

   Update the **amplify/auth/post-confirmation/handler.ts** file with the following code to define the function's handler. Then, **save** the file.

```
import type { PostConfirmationTriggerHandler } from "aws-lambda";
export const handler: PostConfirmationTriggerHandler = async (event) => {
  return event;
};
```



Conclusion

You have defined a Lambda function using Amplify.

Task 3: Create a Data Table

Time to complete	5 minutes
Services used	AWS Amplify AWS AppSync
Get help	Troubleshooting Amplify Learn about Data

Overview

In this task, you will create a data model to persist data using a GraphQL API and Amazon DynamoDB. Additionally, you will use AWS Identity and Access Management (IAM) authorization to securely give our services the required permissions to interact with each other. You will also allow the Lambda function you created in the previous task to use the GraphQL API to write to your newly created DynamoDB table using an IAM policy.

Key concepts

Amplify backend: Amplify Gen 2 uses a full stack TypeScript developer experience (DX) for defining backends. Simply author app requirements like data models, business logic, and auth rules in TypeScript. Amplify automatically configures the correct cloud resources and deploys them to per-developer cloud sandbox environments for fast, local iteration.

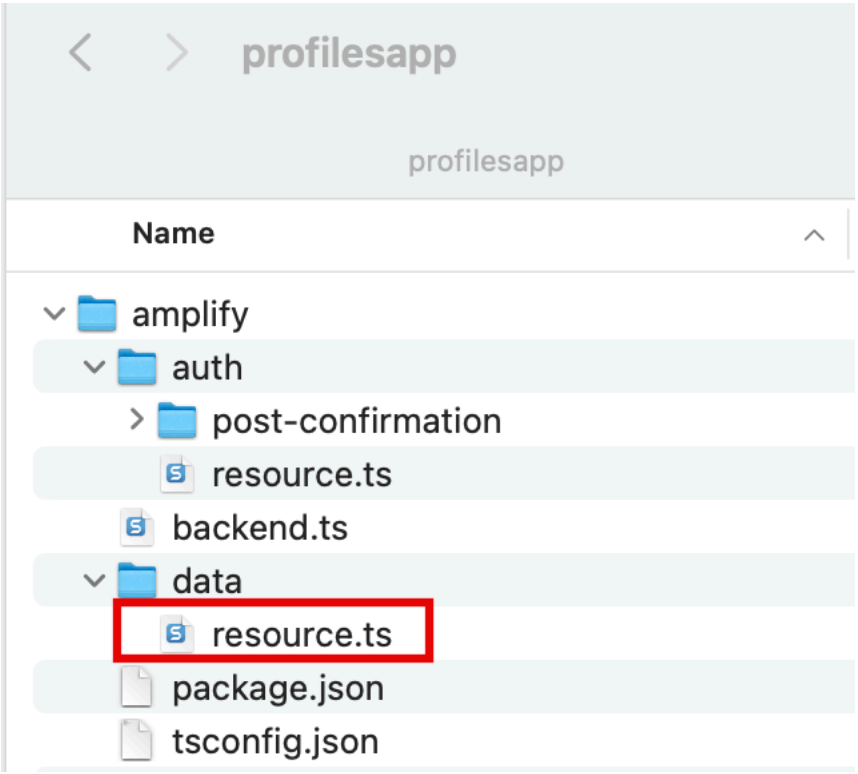
Implementation

Step 1: Set up Amplify Data

1. Update the resource file

   On your local machine, navigate to the **amplify/data/resource.ts** file and **update** it with the code in this file. Then, **save** the file.

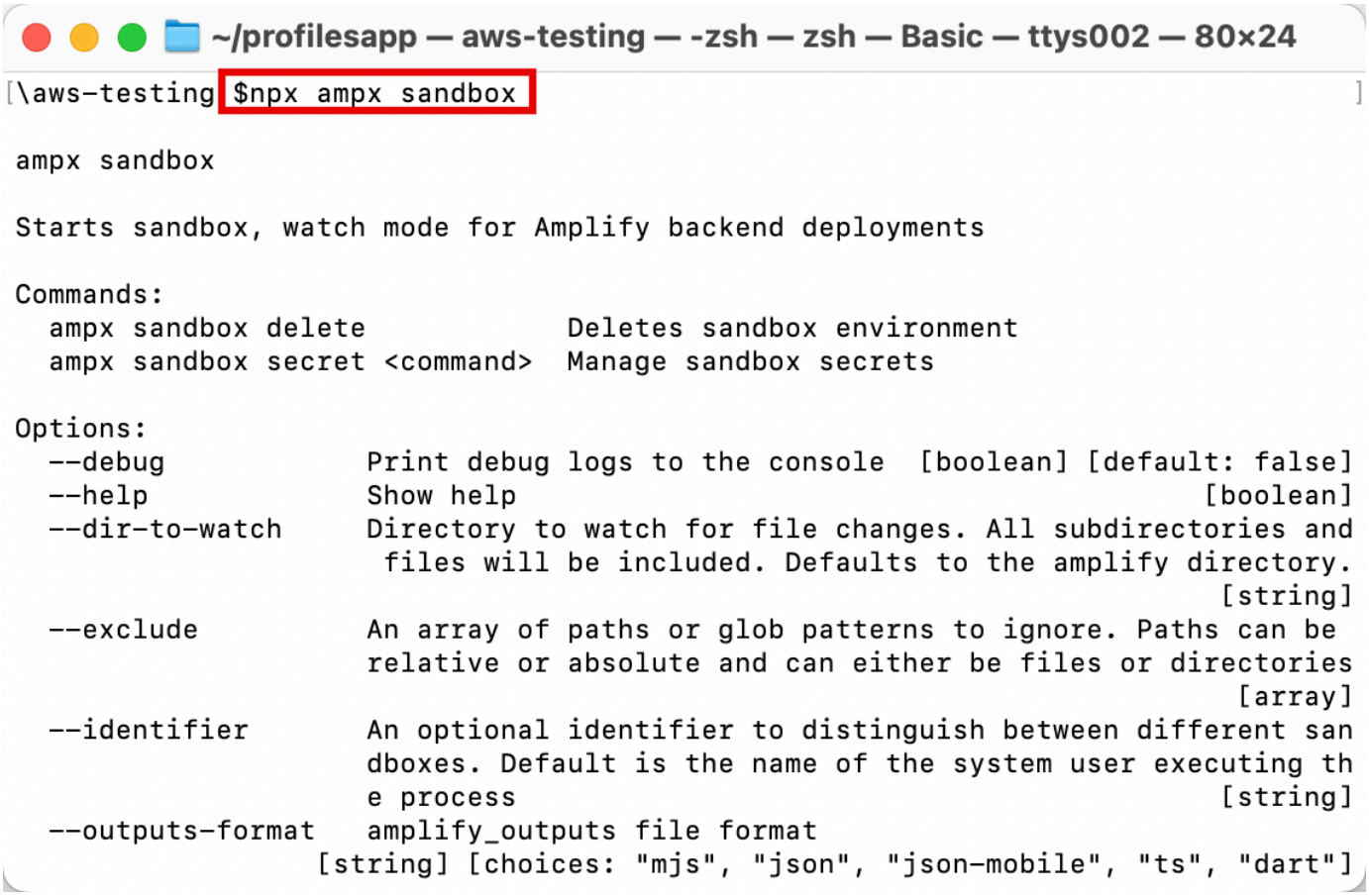
- This code will define the schema for the UserProfile data model using a per-owner authorization rule allow.owner() to restrict the expense record's access to the creator of the record
- It also uses the field profileOwner to track the ownership, and configures the authorization rule to allow the postConfirmation resource.
- Granting access to resources creates environment variables for your function, such as the GraphQL API endpoint.



2. Deploy sandbox

Open a new terminal window, **navigate** to your app's root folder (**profilesapp**), and **run** the following command to deploy cloud resources into an isolated development space so you can iterate fast.

```
npx ampx sandbox
```

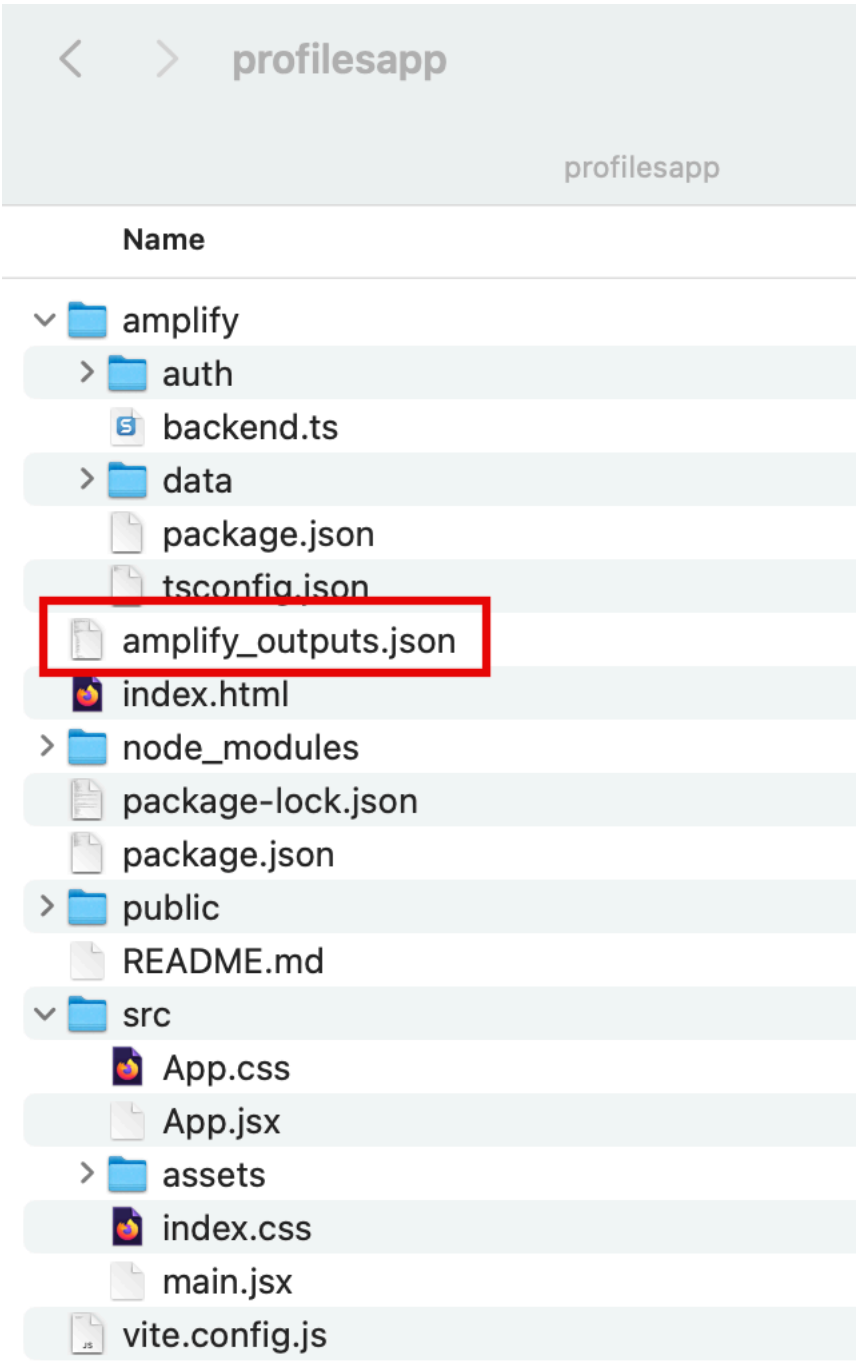


3. View confirmation message

   Once the cloud sandbox has been fully deployed, your terminal will display a **confirmation message**.

4. Verify outputs file is added to your project

   Verify that the **amplify_outputs.json** file was **generated and added** to your project.



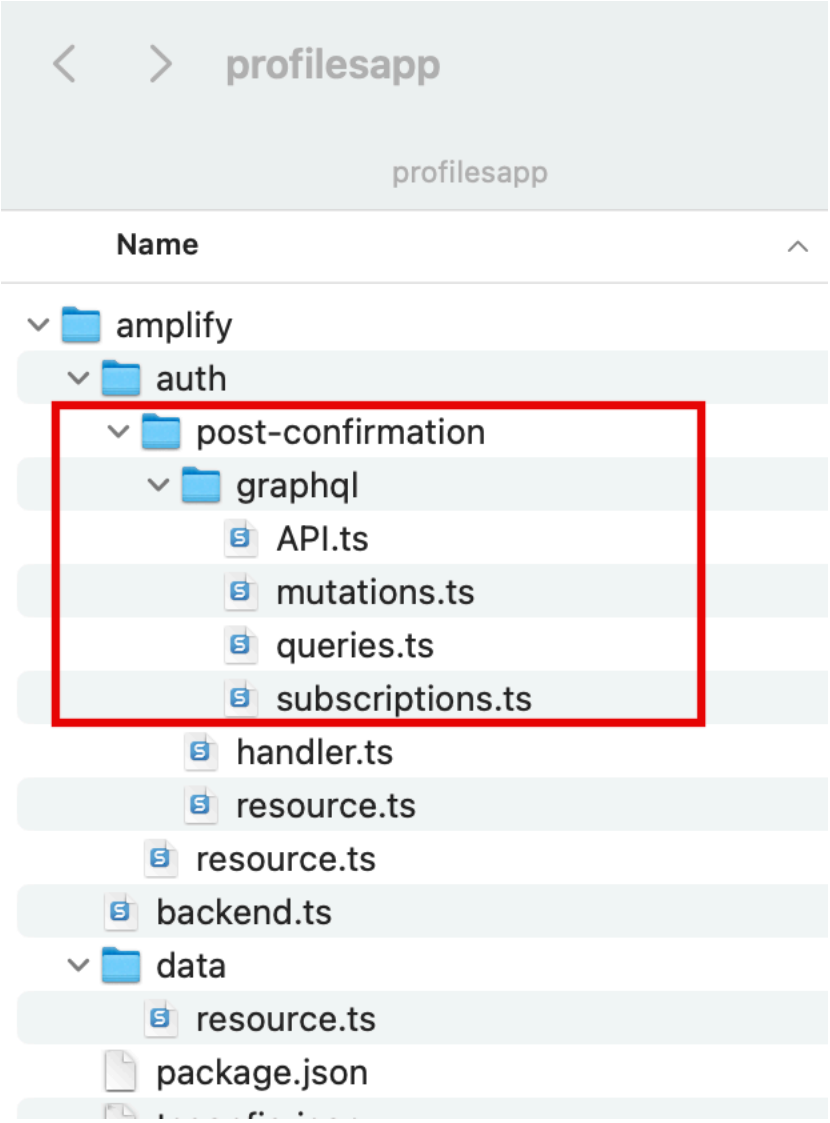
5. Generate GraphQL client code

   Open a new terminal window, **navigate** to your app's root folder(**profilesapp**), and **run** the following command to generate the GraphQL client code to call your data backend.

> **Note**
>
> You will need to **update** the following command to use the path to the **post-confirmation** folder that you created in the previous step. For example: **npx ampx generate graphql-client-code --out amplify/auth/post-confirmation/graphql.**

```
npx ampx generate graphql-client-code --out <path-to-post-confirmation-handler-dir>/graphql
```

Amplify will create the folder **amplify/auth/post-confirmation/graphql** where you will find the client code to connect to the GraphQL API.

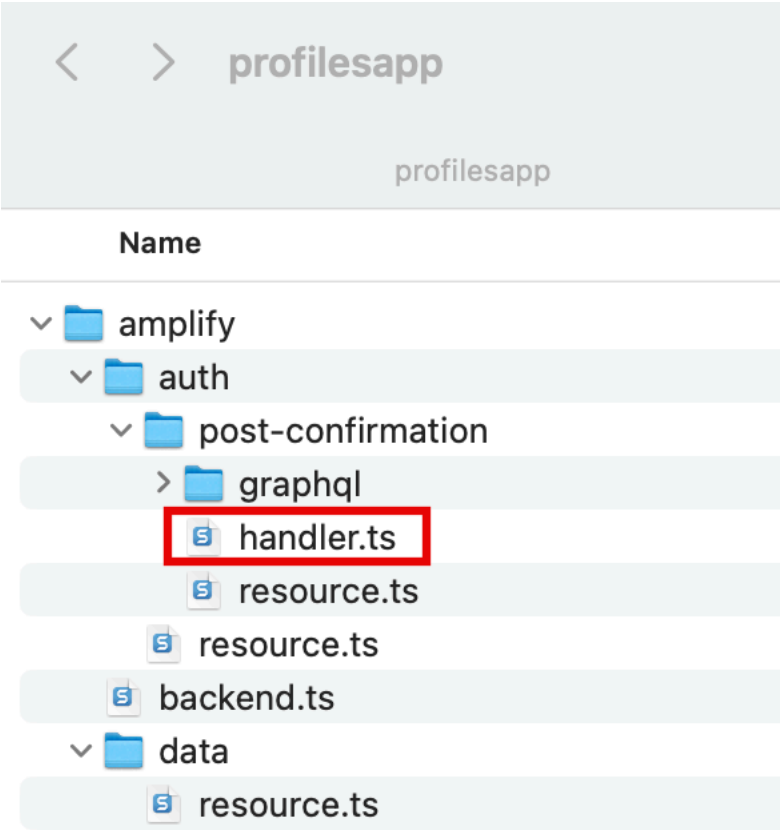


Step 2: Modify Lambda function to connect to the API

- Modify the handler file

  On your local machine, navigate to the **amplify/auth/post-confirmation/handler.ts** file and **replace** the code with the code in this file. Then, **save** the file.

  This code configures the Amplify using the env variables and sets the authorization to use IAM. It then generates a data client using the generateClient() function. Finally, it uses the data client to create a user profile by setting the email and owner based on the attributes of the signed-up user.



Conclusion

You have created a data table and configured a GraphQL API to persist data in an Amazon DynamoDB database. Then, you updated the Lambda function to use the API.

Task 4: Link a Serverless Function to a Web App

Minimum time to complete	10 minutes
Services used	AWS Amplify
Get help	Troubleshooting Amplify Learn about Auth

Overview

In this task, you will update the Amplify Auth resources to use the Lambda function you created in the previous module as an Amazon Cognito post confirmation invocation. When the user completes the sign up, the function will use the GraphQL API and capture the user's email into the DynamoDB table.

Key concepts

Lambda invocation: The type of event that will make a Lambda (serverless) function run. This can be another AWS service or an external input.

Implementation

Step 1: Set up Amplify Auth

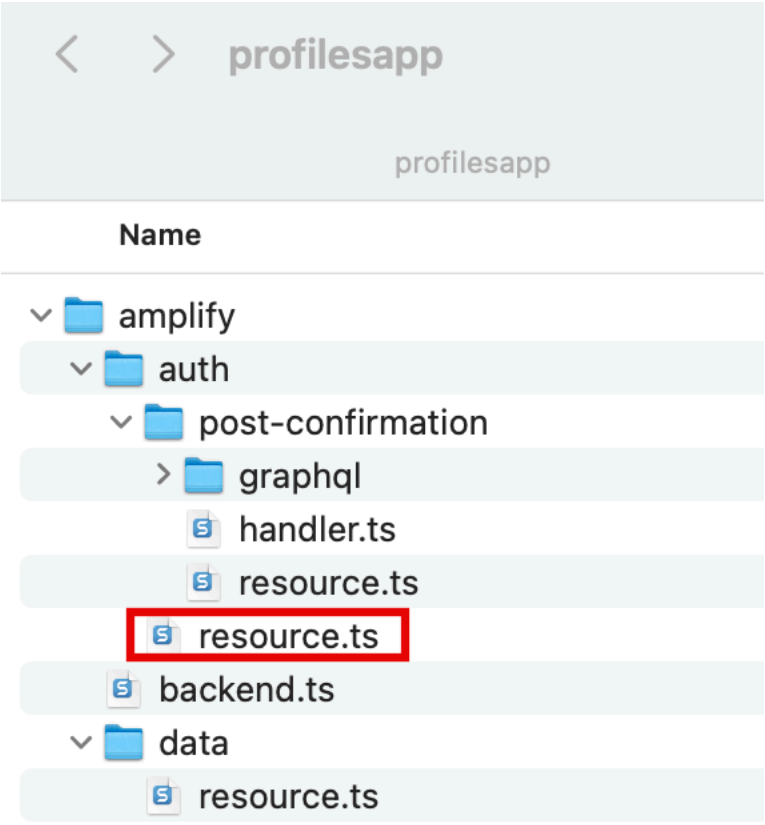
By default, your auth resource is configured allowing the user to sign up using email, but you need to update the resource to invoke the previously created postConfirmation function.

1. Update the resource file

   On your local machine, navigate to the **amplify/auth/resource.ts** file and **update** it with the following code. Then, **save** the file.

```
import { defineAuth } from '@aws-amplify/backend';
import { postConfirmation } from './post-confirmation/resource';
```

```
export const auth = defineAuth({
  loginWith: {
    email: true,
  },
  triggers: {
    postConfirmation
  }
});
```



2. Start the sandbox, if necessary

   The sandbox will automatically get updated and redeployed once the file is updated. If the sandbox is not running, you can run the following command in a new terminal window to start it.

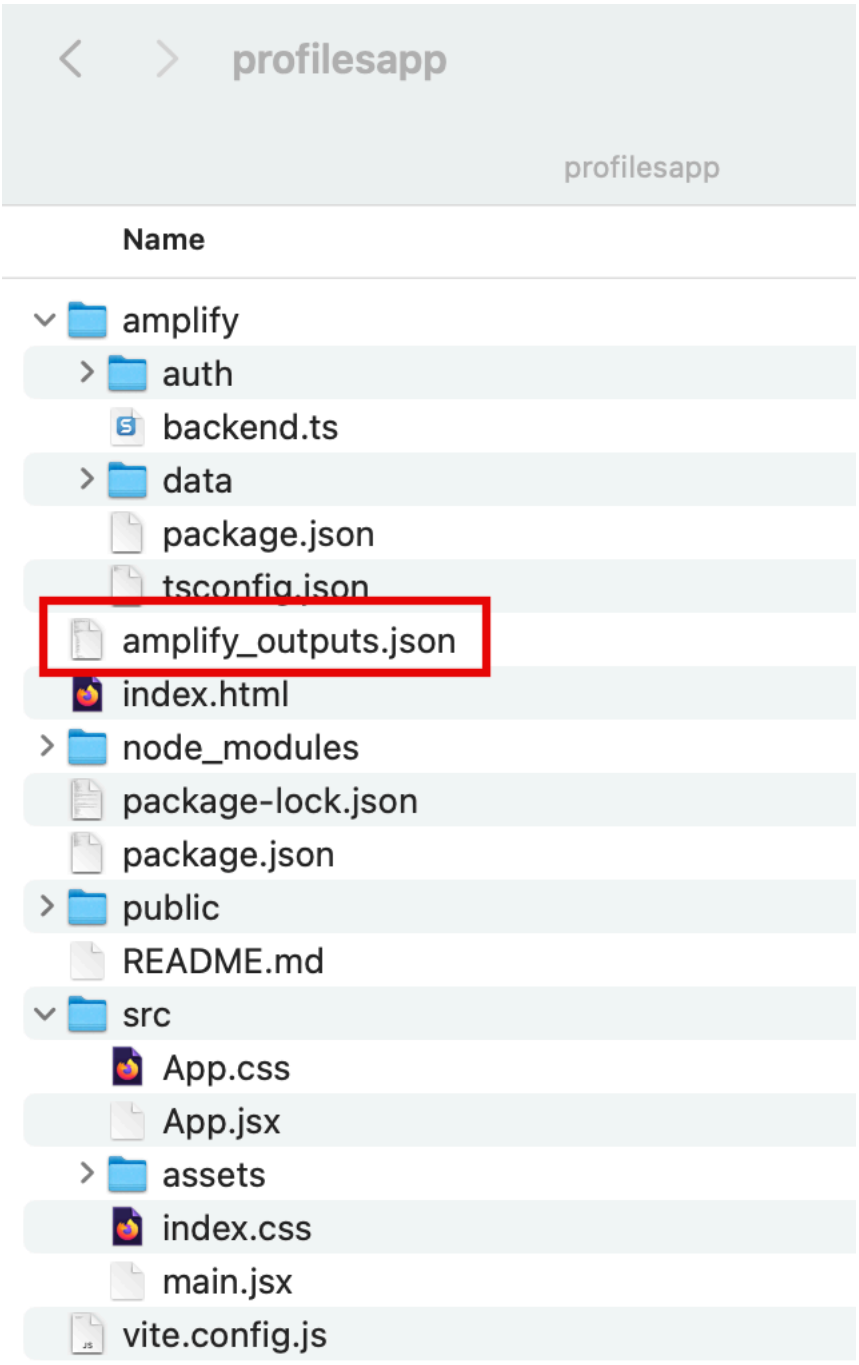
   ```
   npx ampx sandbox
   ```

3. View the confirmation message

   Once the cloud sandbox has been fully deployed, your terminal will display a **confirmation message**.

4. View the file

   The **amplify_outputs.json** file will be **generated/updated** and **added** to your **profilesapp** project.



Conclusion

You used Amplify to configure auth and configured the Lambda function to be invoked when the user signs in to the app.

Task 5: Add Interactivity to Your Web App

Time to complete	5 minutes
Services used	AWS Amplify
Get help	Troubleshooting Amplify Troubleshooting common issues using Amplify UI

Overview

In this task, you will update the website you created in task one to use the Amplify UI component library to scaffold out an entire user authentication flow, allowing users to sign up, sign in, and reset their password and invoke the GraphQL API we created in module three. This will add the ability to display the user's email that was captured using a serverless function.

Key concepts

Amplify libraries: The Amplify libraries allow you to interact with AWS services from a web or mobile application.

Implementation

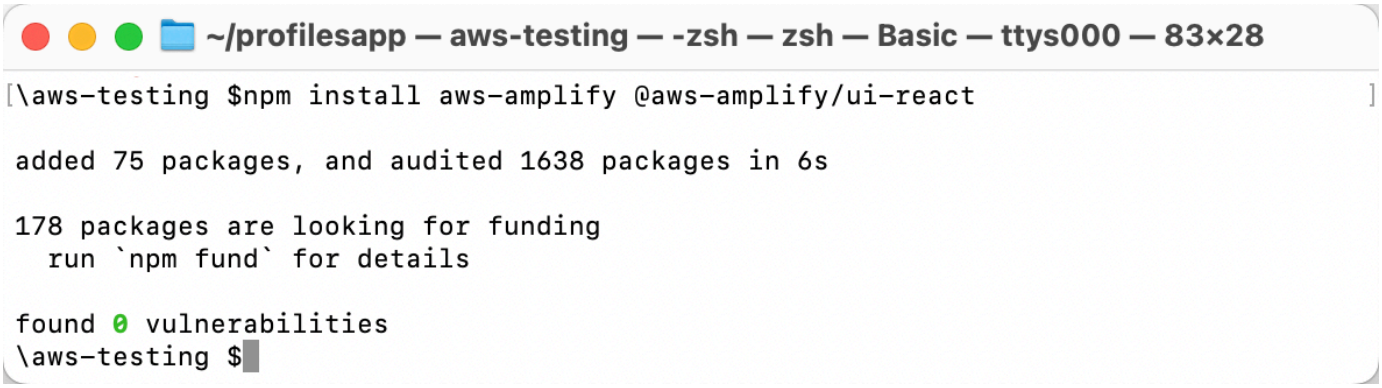
Step 1: Install the Amplify libraries

You will need two Amplify libraries for your project. The main **aws-amplify library** contains all of the client-side APIs for connecting your app's frontend to your backend, and the **@aws-amplify/ui-react** library contains framework-specific UI components.

- Install the libraries

  In a new terminal window, in your projects root folder (**profilesapp**), **run** the following command to install the libraries.

```
npm install aws-amplify @aws-amplify/ui-react
```



Step 2: Style the App UI

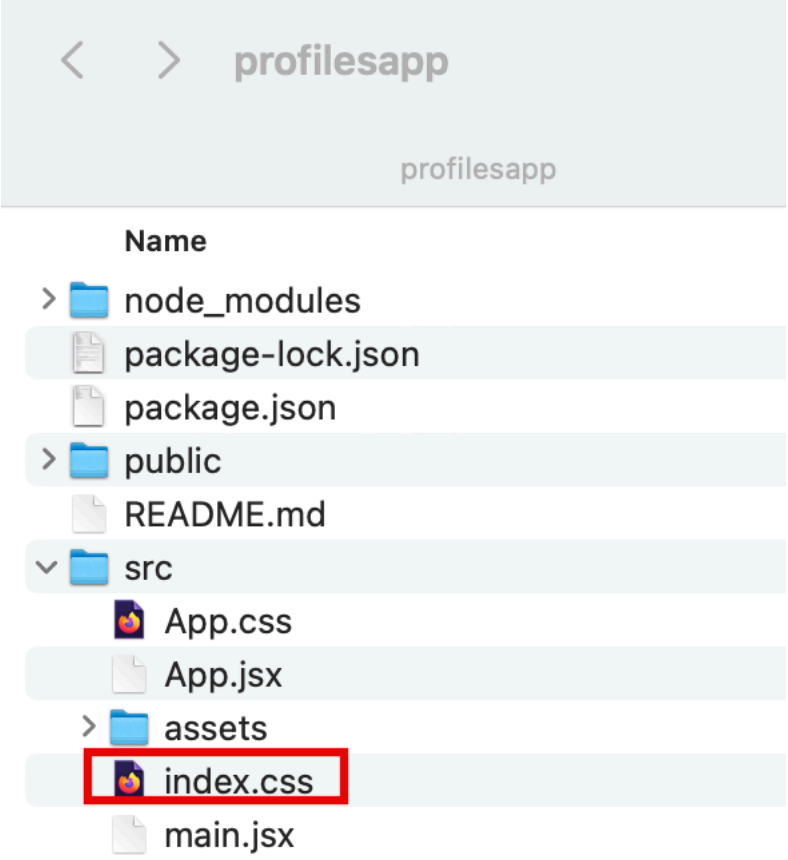
You will need two Amplify libraries for your project. The main **aws-amplify library** contains all of the client-side APIs for connecting your app's frontend to your backend, and the **@aws-amplify/ui-react** library contains framework-specific UI components.

- Modify the CSS

  On your local machine, navigate to the **profilesapp/src/index.css** file and **update** it with the following code. Then, **save** the file.

```
:root {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
  line-height: 1.5;
  font-weight: 400;

  color: rgba(255, 255, 255, 0.87);

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;

  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;

}
```

```
.card {
  padding: 2em;
}

.read-the-docs {
  color: #888;
}

.box:nth-child(3n + 1) {
  grid-column: 1;
}
.box:nth-child(3n + 2) {
  grid-column: 2;
}
.box:nth-child(3n + 3) {
  grid-column: 3;
}
```

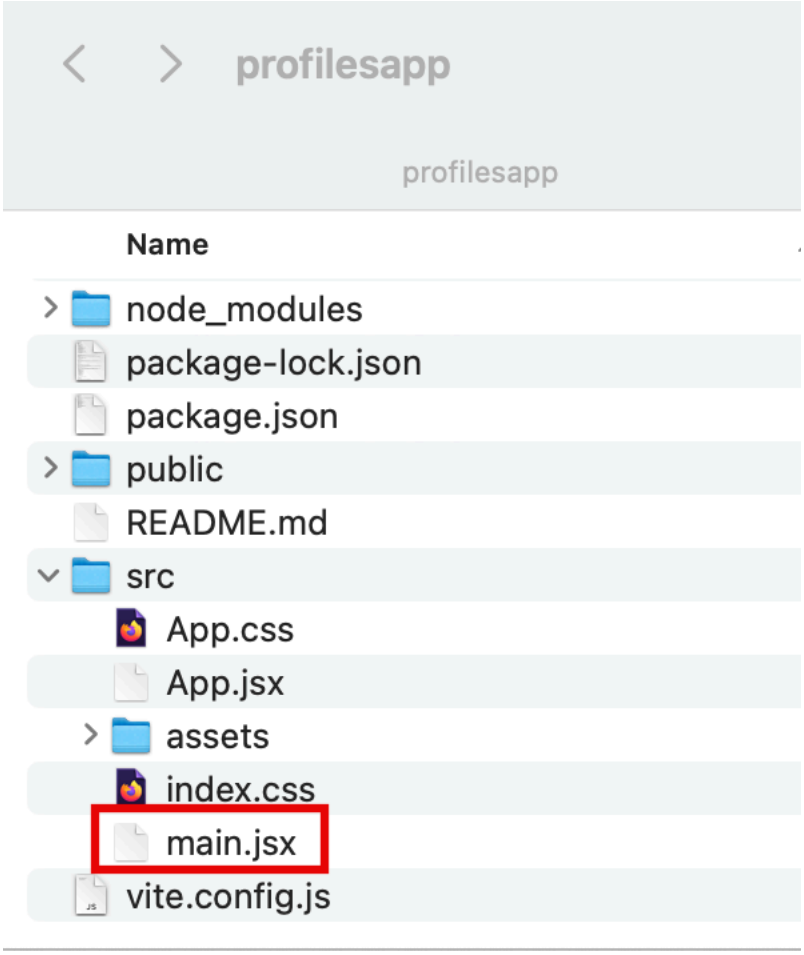


Step 3: Implement the UI

1. Modify the main.jsx file

   On your local machine, navigate to the **profilesapp/src/main.jsx** file and **update** it with this code. Then, **save** the file.

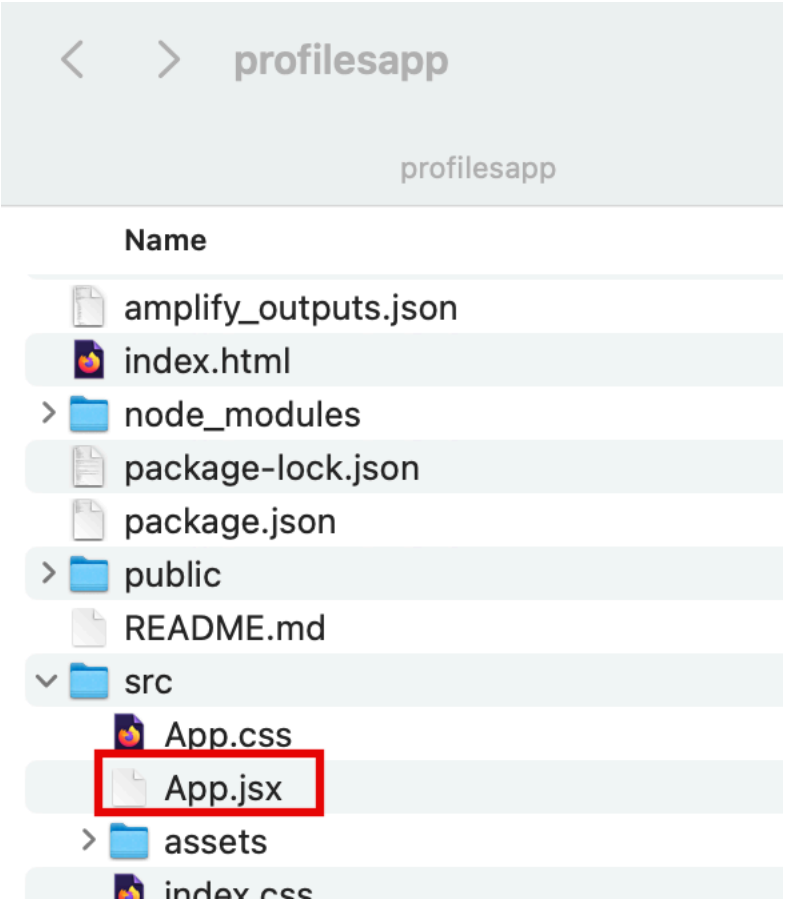
   - The code will use the Amplify Authenticator component to scaffold out an entire user authentication flow allowing users to sign up, sign in, reset their password, and confirm sign-in for multifactor authentication (MFA).



2. Modify the app.jsx file

   Open the **profilesapp/src/App.jsx** file, and **update** it with this code. Then, **save** the file.

- The code starts by configuring the Amplify library with the client configuration file (**amplify_outputs.json)** . It then generates a data client using the **generateClient()** function. The app will use the data client to get the user's profile data.



3. **Open** a new terminal window, **navigate** to your project's root directory (*profilesapp*), and **run** the following command to launch the app:

```
npm run dev
```

4. Select the **Local host link** to open the Vite + React application.

5. Create a new user

   Choose the **Create Account** tab, and use the authentication flow to create a new user by entering your **email address** and a **password**.

   Then, choose **Create Account**.

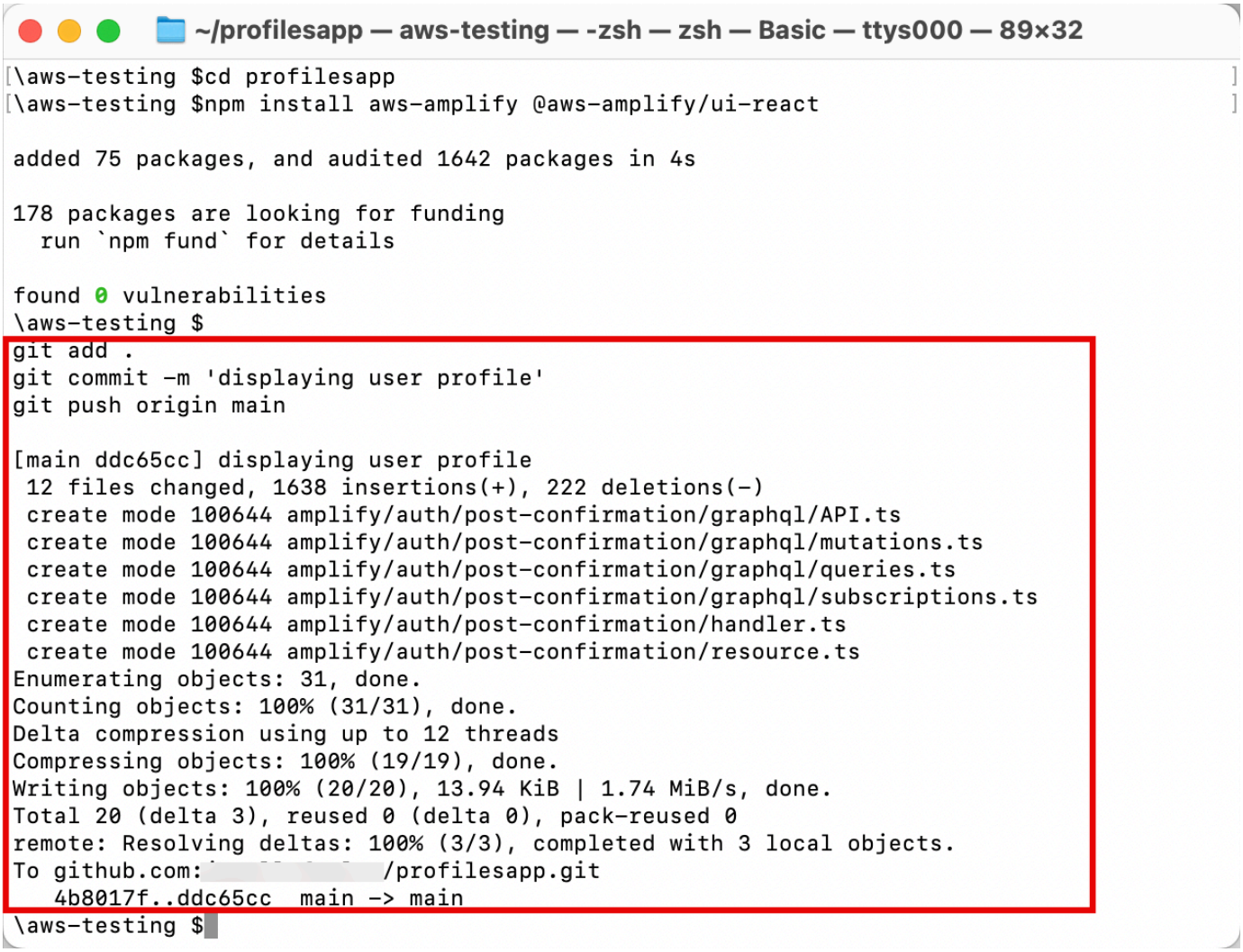
   You will get a verification code sent to your email. Enter the **verification code** to log in to the app.

   When signed in, the app will display your email address.

6. Push the changes

   In the open terminal window, **run** the following command to push the changes to GitHub:

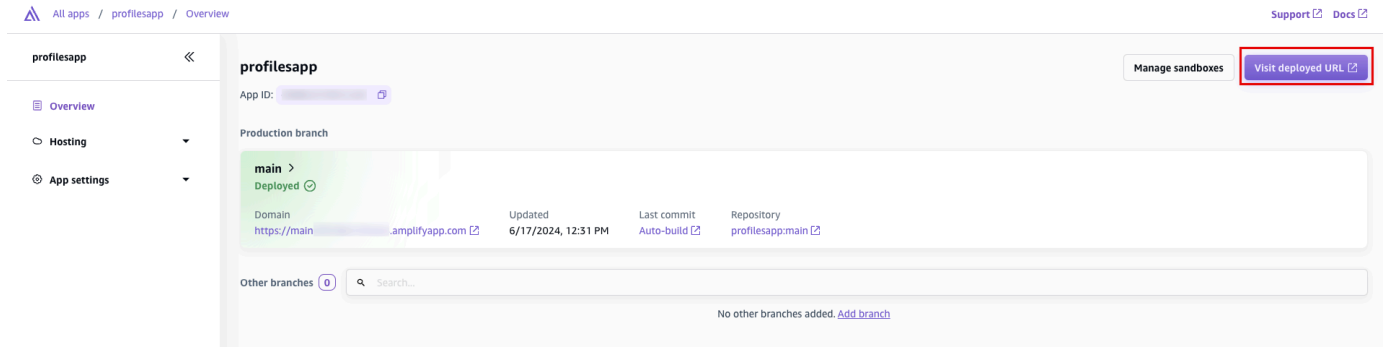
```
git add .
git commit -m 'displaying user profile'
git push origin main
```



7. Verify the deployment instance updates

   Sign in to the AWS Management console in a new browser window, and **open** the AWS Amplify console at https://console.aws.amazon.com/amplify/apps .

   AWS Amplify automatically builds your source code and deployed your app at **https://...amplifyapp.com** , and on every git push your deployment instance will update. Select the **Visit deployed URL** button to see your web app up and running live.


All apps / profilesapp / Overview
Support
Docs
profilesapp
Overview
Hosting
App settings
profilesapp
Manage sandboxes
Visit deployed URL
App ID:
Production branch
main
Deployed
Domain
https://main .amplifyapp.com
Updated
6/17/2024, 12:31 PM
Last commit
Auto-build
Repository
profilesapp:main
Other branches 0
Search...
No other branches added. Add branch

Task 6: Clean Up Resources

Time to complete	5 minutes

Overview

In this task, you will go through the steps to delete all the resources you created throughout this tutorial. It is a best practice to delete resources you are no longer using to avoid unwanted charges.

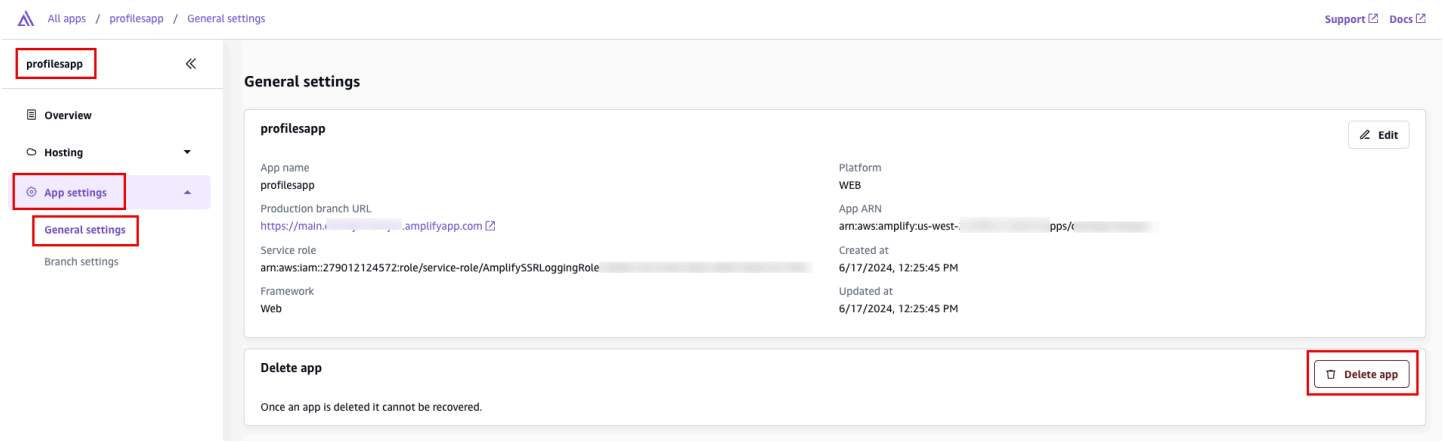
Implementation

Step 1: Open general settings

In the Amplify console, in the left-hand navigation for the **profilesapp**, choose **App settings**, and select **General settings**.

Step 2: Delete the app

In the **General settings** section, choose **Delete app**.



Congratulations

You have created a React web app and used Amplify to configure auth and data resources. Additionally, you created a function to update the data model with the user's details when they sign up. Then, you created frontend to display the captured data. Finally, you used AWS Amplify to host the app!