Hands-on tutorials

Build a Full Stack React Application

Build a Full Stack React Application: Hands-on tutorials

Table of Contents

Build a Full Stack React Application

AWS experience	Beginner
Time to complete	30 minutes
Cost to complete	Free Tier eligible
Services used	AWS Amplify
Last updated	July 25, 2024

Overview

In this tutorial, you will learn how to create a simple full-stack web application using AWS Amplify. Amplify offers a Git-based CI/CD workflow for building, deploying, and hosting single-page web applications or static sites with serverless backends.

What you will accomplish

In this tutorial, you will:

- Build and host a React application on AWS
- Use Amplify to add authentication, data & storage solutions to the app
- Start a cloud sandbox environment that provides an isolated development space to rapidly build, test, and iterate on a fullstack app
- Implement the frontend code to enable users to create, update, and delete notes

Prerequisites

Before starting this tutorial, verify that you have the following prerequisites completed:

- **An AWS account:** If you don't already have one, follow the Setup Your Environment tutorial.
- **Configured** your AWS profile for local development.
- **Installed** on your environment: Nodejs and npm.

- **Familiarity** with git and a GitHub account.

Tasks

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

1. Task 1: Deploy and Host a React App (10 minutes): Create a React app, then deploy and host it using AWS Amplify.
2. Task 2: Initialize a Local Amplify App (10 minutes): Initialize a cloud backend that include authentication, a database, and storage.
3. Task 3: Build the Frontend (10 minutes): Implement the frontend code to connect to the authorization, data and storage backend enabling users to create, update, and delete notes.
4. Task 4: Clean up resources (2 minutes): Clean up the resources used in this tutorial.

Task 1: Deploy and Host a React App

Time to complete	10 minutes
Services used	AWS Amplify
Requires	• A GitHub account • GitHub SSH connection • Nodejs and npm
Get help	• Troubleshooting Amplify • Learn about Hosting

Overview

AWS Amplify offers a Git-based CI/CD workflow for building, deploying, and hosting single-page web applications or static sites with serverless backends. When connected to a Git repository, Amplify determines the build settings for both the frontend framework and any configured serverless 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. 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. Finally, you will demonstrate continuous deployment capabilities by making changes to the React application, pushing a new version to the main branch, and observing how it automatically invokes a new deployment.

What you will accomplish

- Create a React application
- Initialize a GitHub repository
- Deploy your app with AWS Amplify
- Implement code changes and redeploy your app

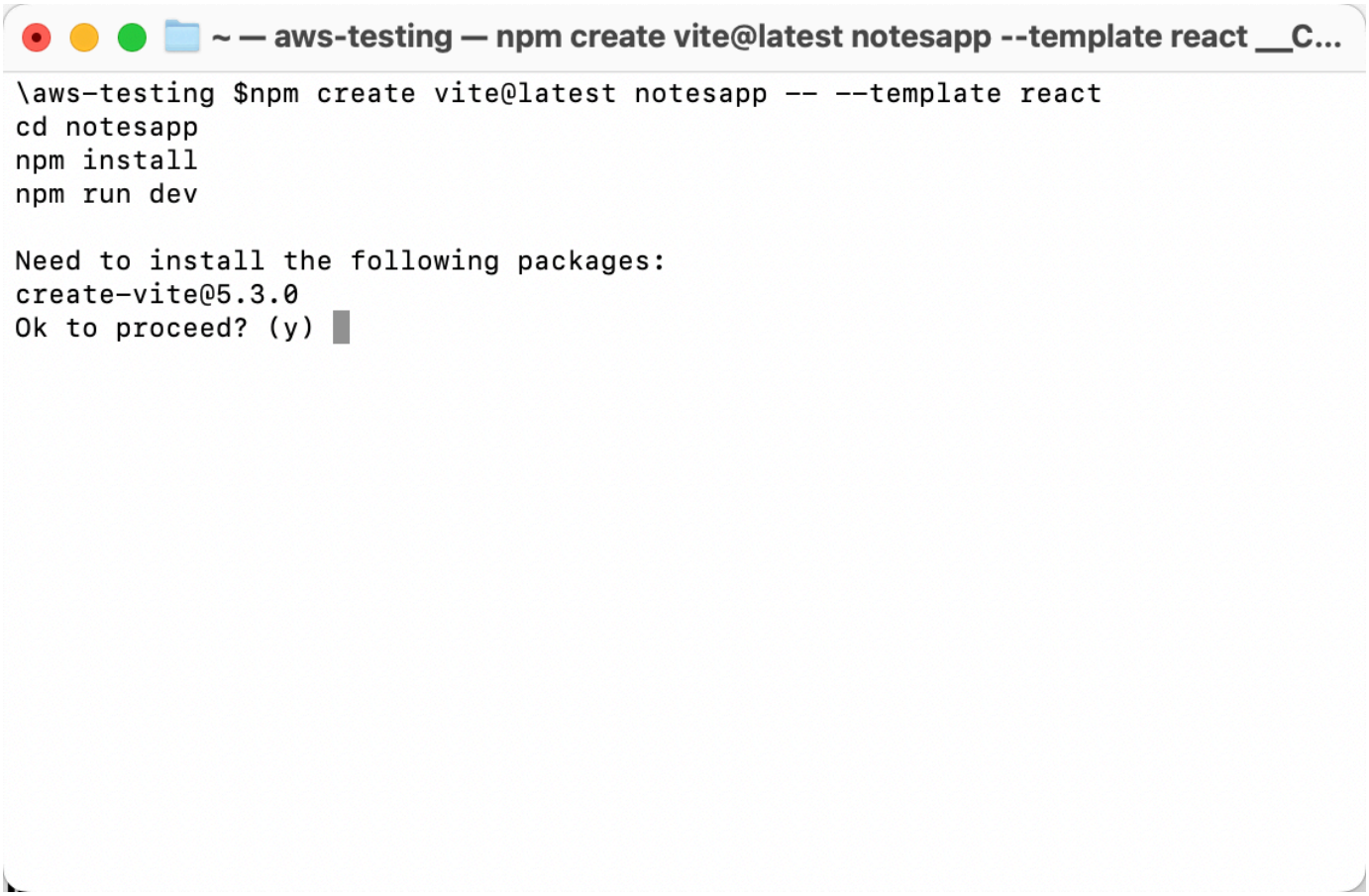
Implementation

Step 1: Create a new React Application

1. Set up the React environment

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

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



2. View your application

   In the terminal window, choose the **Local link**.

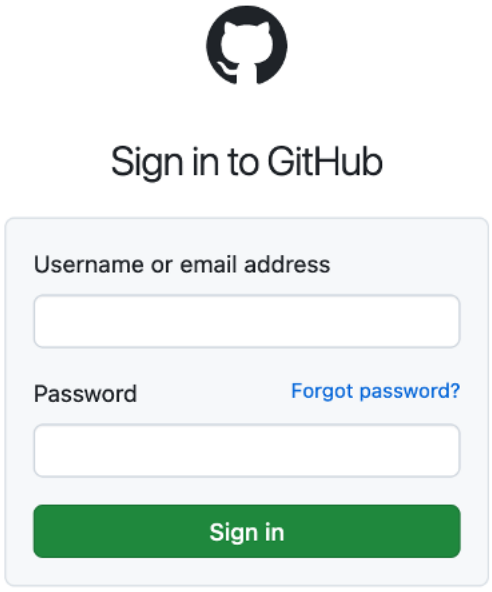
Step 2: Create the GitHub repository and commit code

Before you begin:

- You need a GitHub account. If you don't have one, sign up here.
- If you've never used GitHub on your computer, generate and add an SSH key to your account. For instructions, see Connecting to GitHub with SSH.

1. Initialize GitHub repository

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

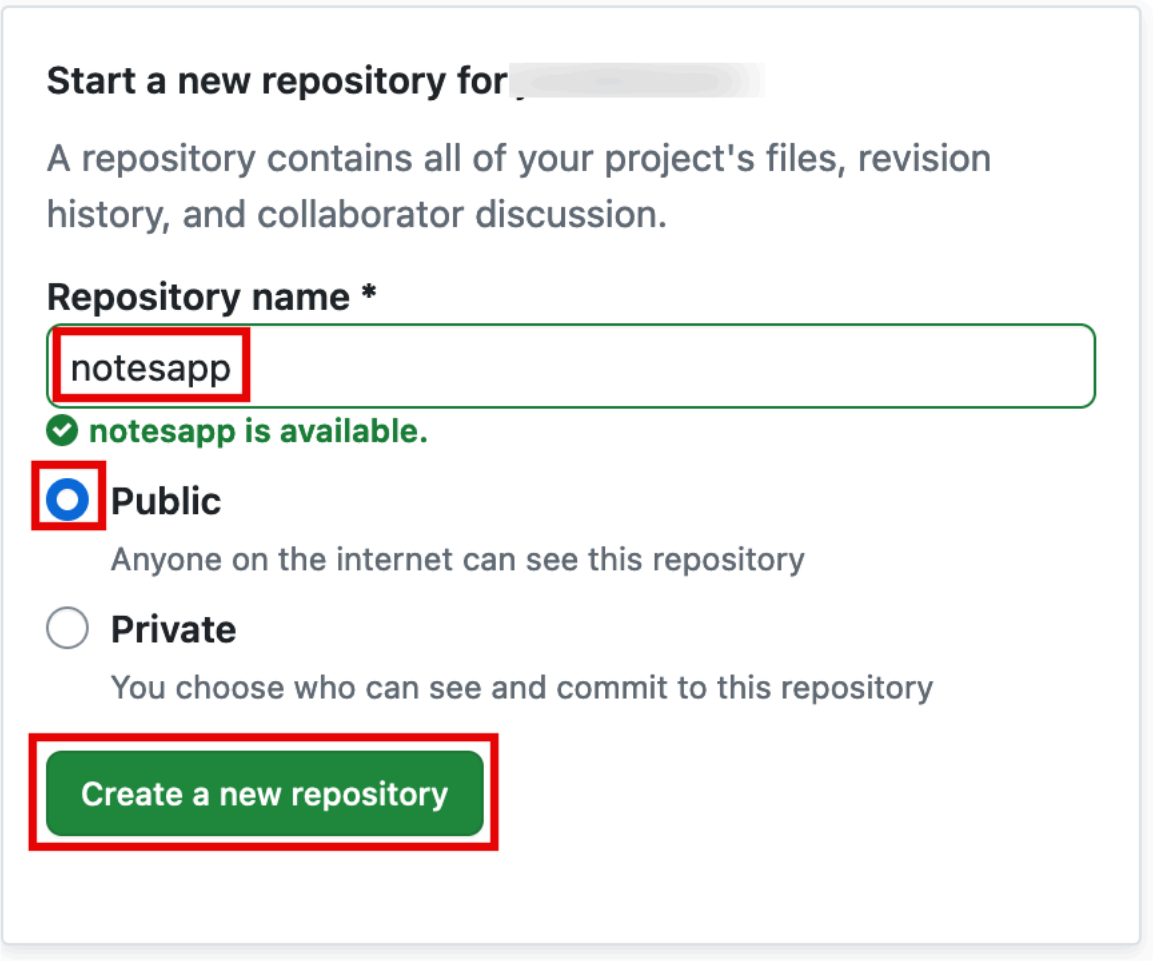


2. Create a repository

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

   For **Repository name**, enter **notesapp**, and choose the **Public** radio button.

Then select, **Create a new repository**.



3. Push the new repo

Open a new terminal window, **navigate** to your app's root folder (**notesapp**), and run the following commands to initialize a git and push the application to the new GitHub repo:

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

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

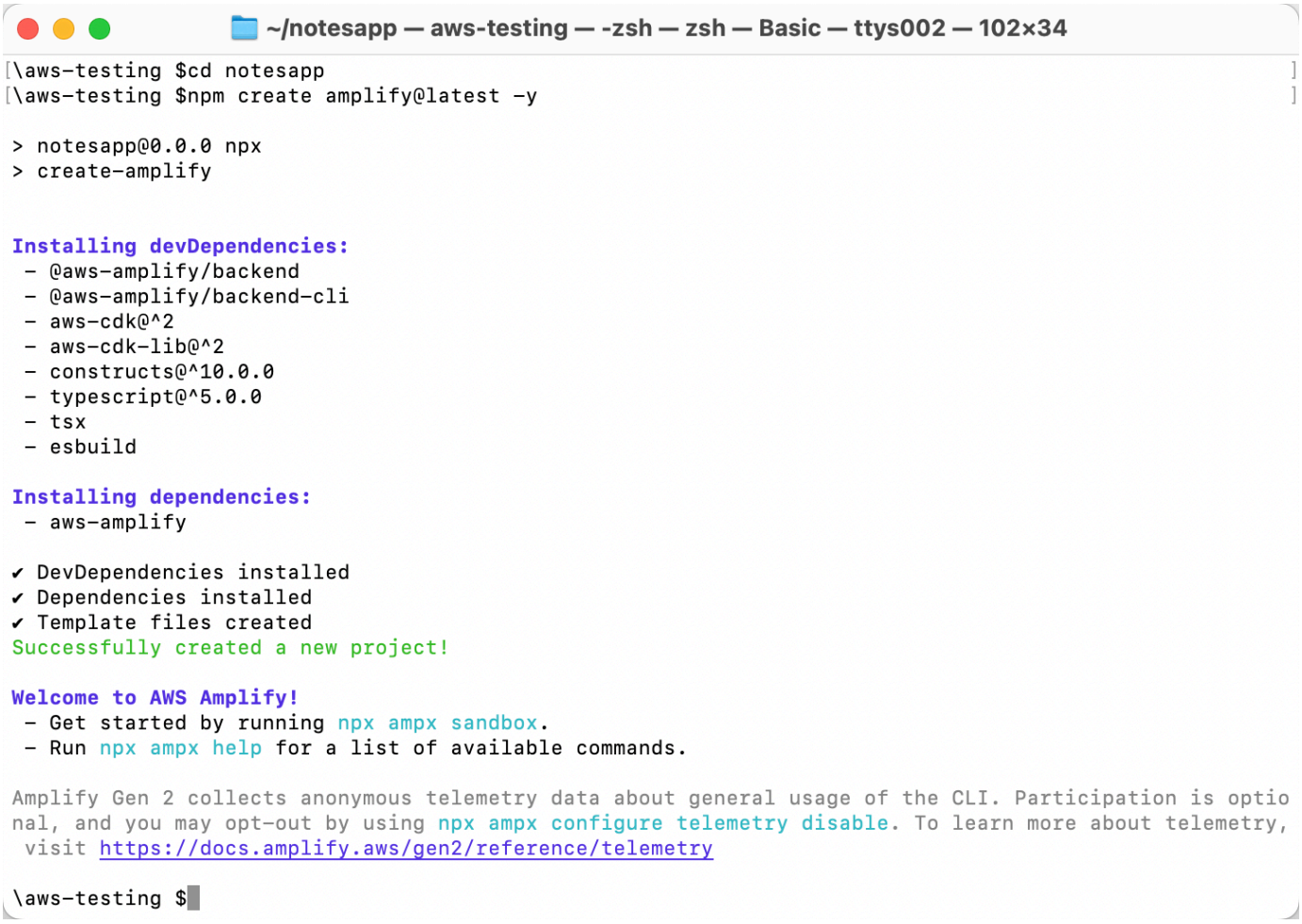
```
git push -u origin main
```

Step 3: Install the Amplify Packages

1. Configure your local repository

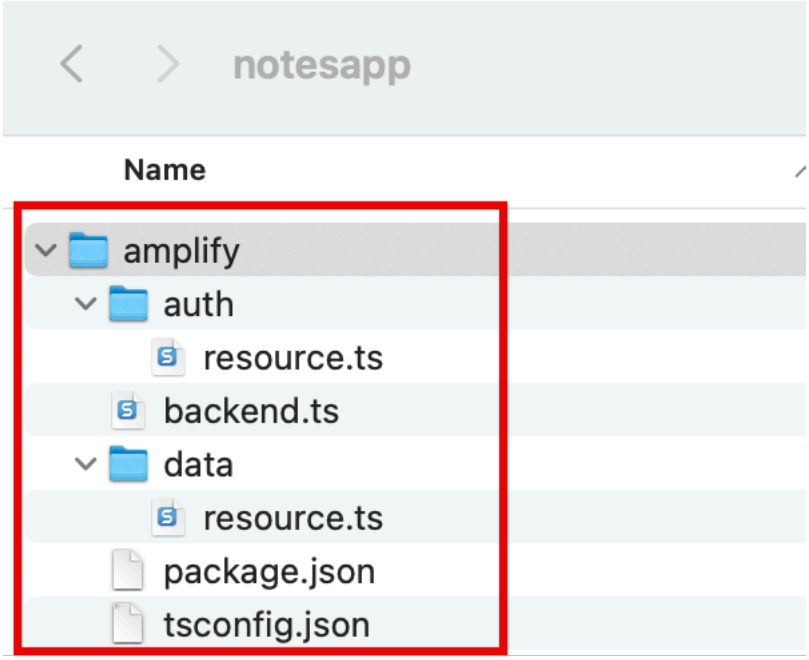
   Open a new terminal window, **navigate** to your app's root folder (**notesapp**), and run the following command:

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



2. Review the Amplify project structure

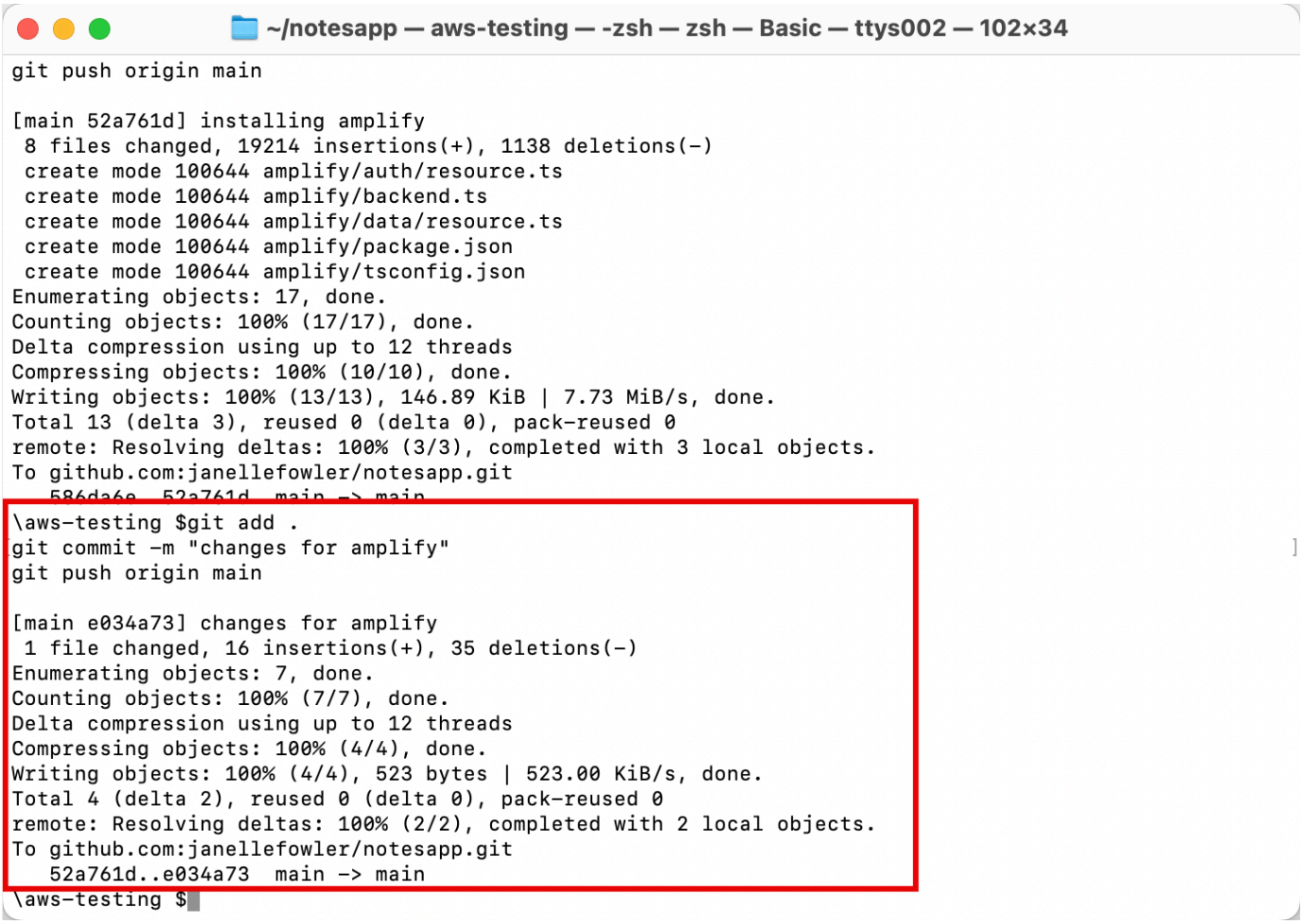
   Running the previous command will scaffold a lightweight Amplify project in the app's directory.



3. Push your changes to GitHub

   In your open terminal window, run the following commands to push the changes to GitHub:

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



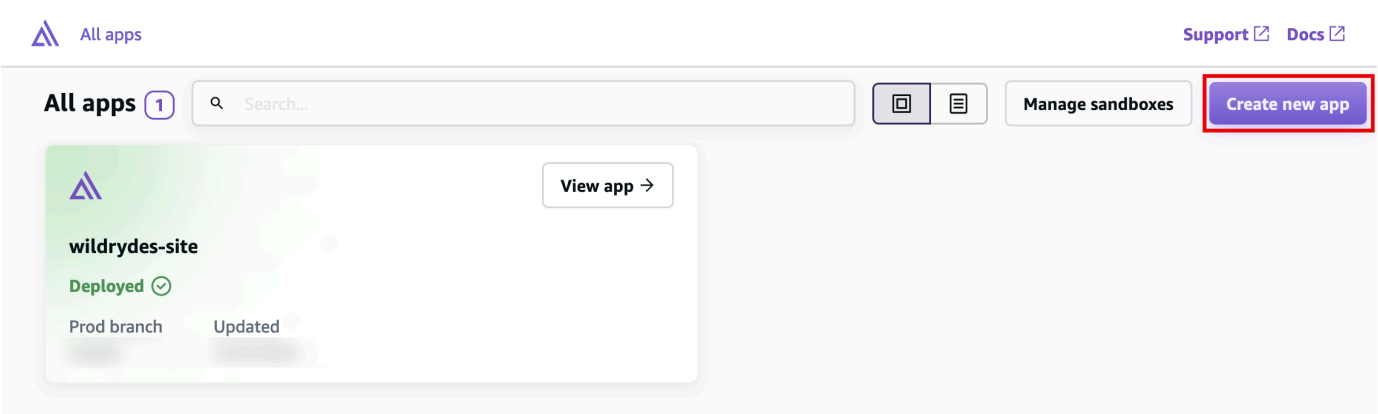
Step 4: Deploy your app with AWS Amplify

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

1. Create the Amplify App

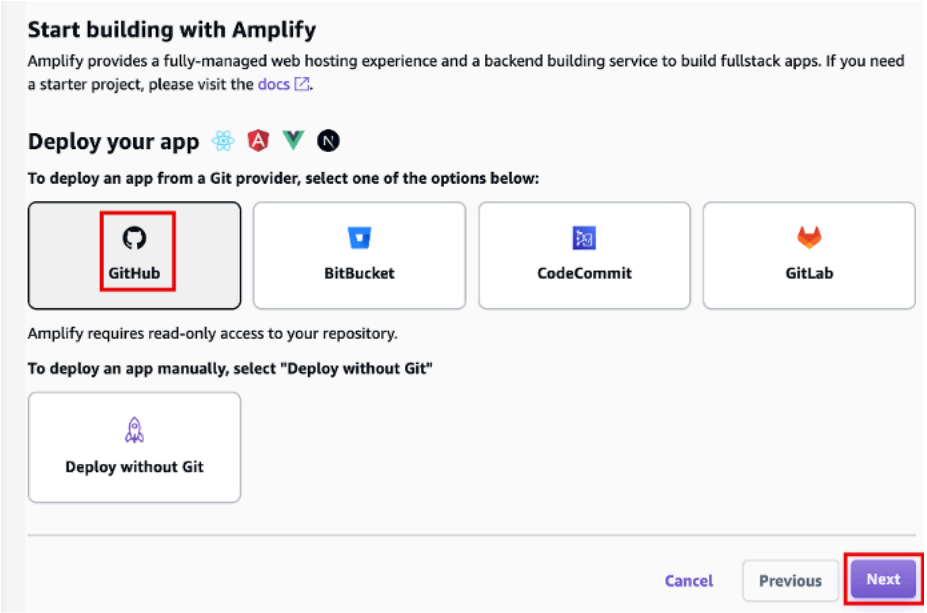
   Sign in to the AWS Management console in a new browser window, and **open** the AWS Amplify console at https://console.aws.amazon.com/amplify/apps.

   Choose **Create new app**.



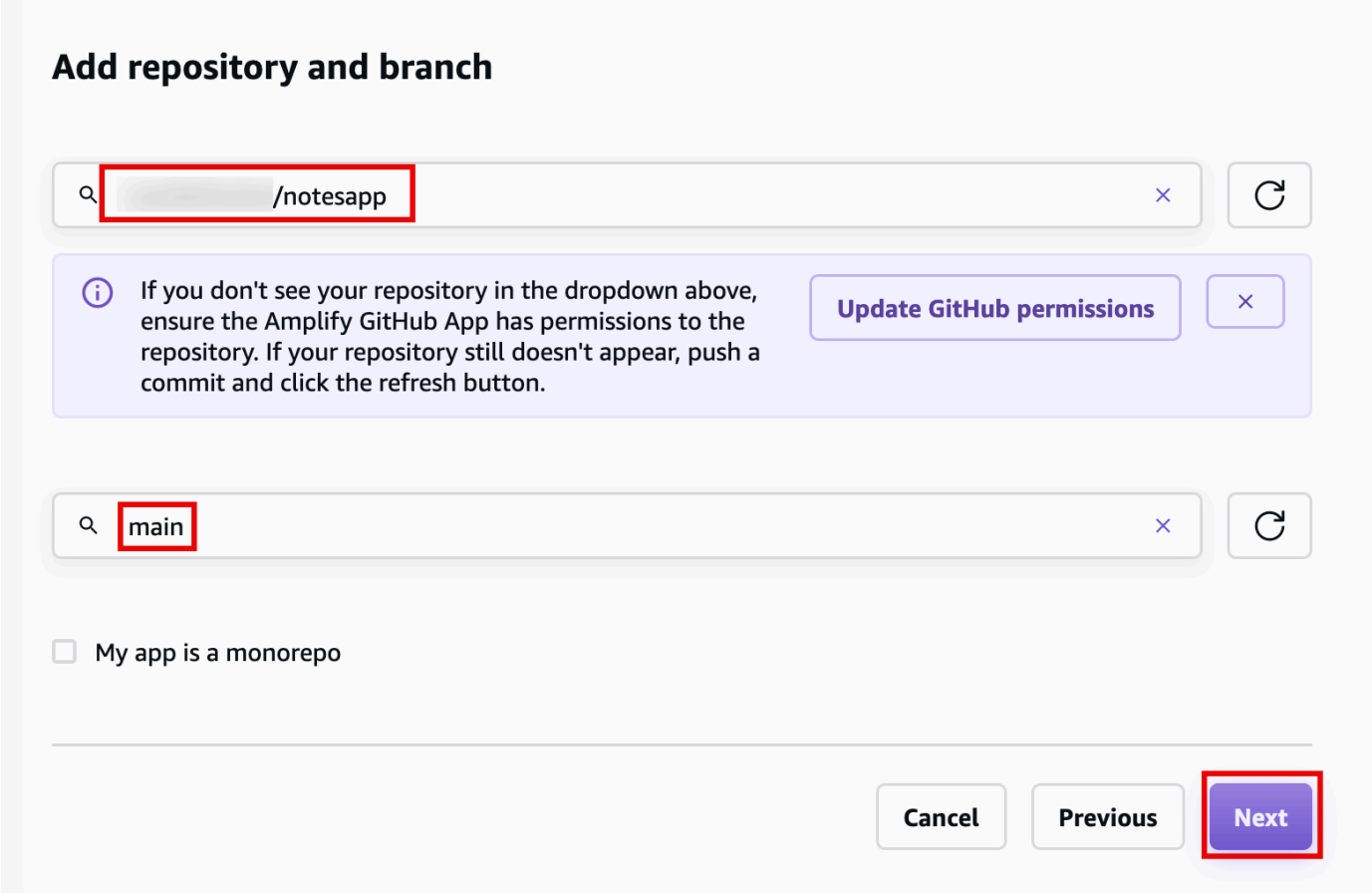
2. Connect to your GitHub repository

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



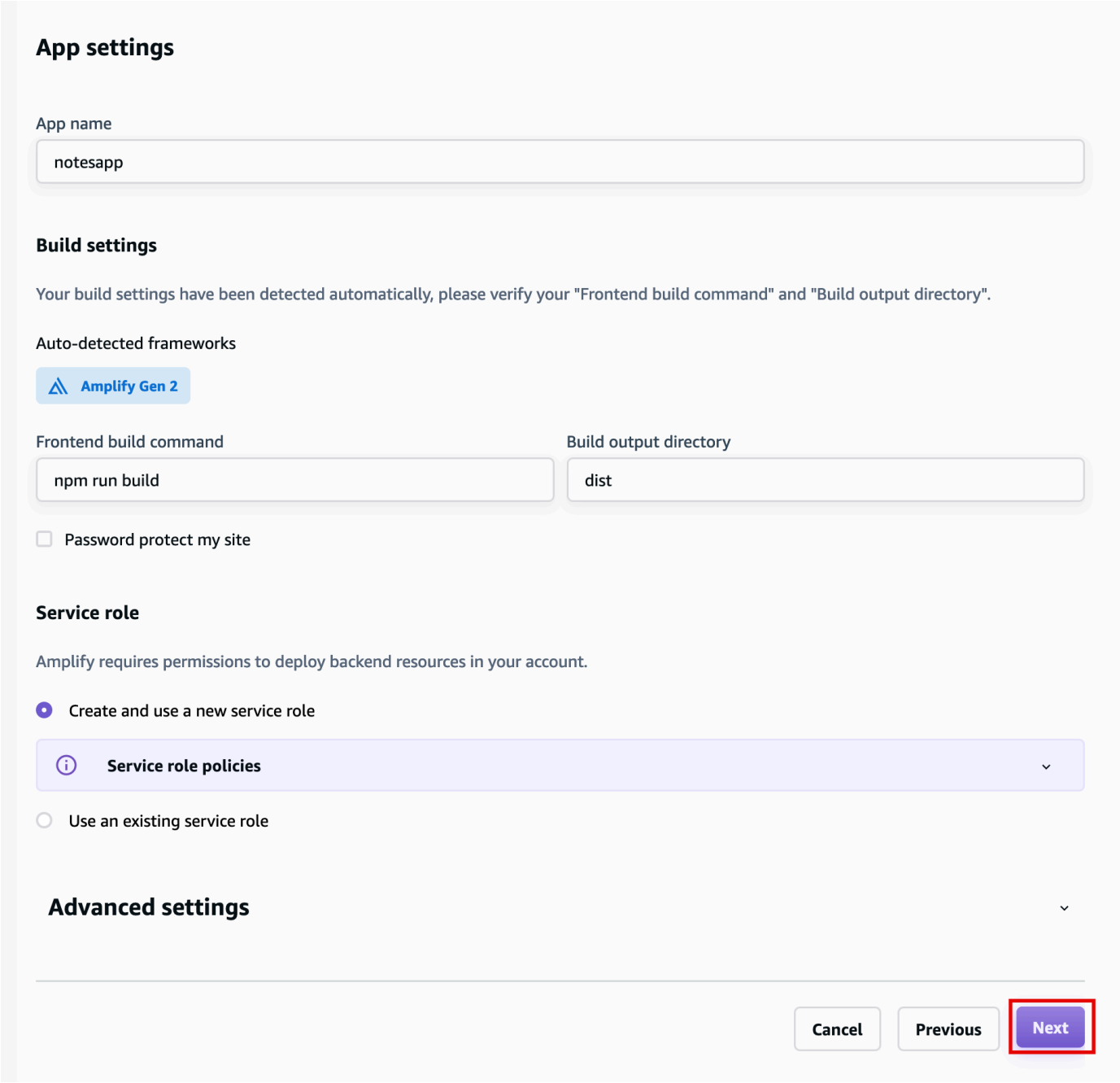
3. Authorize and select your respository

   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**.



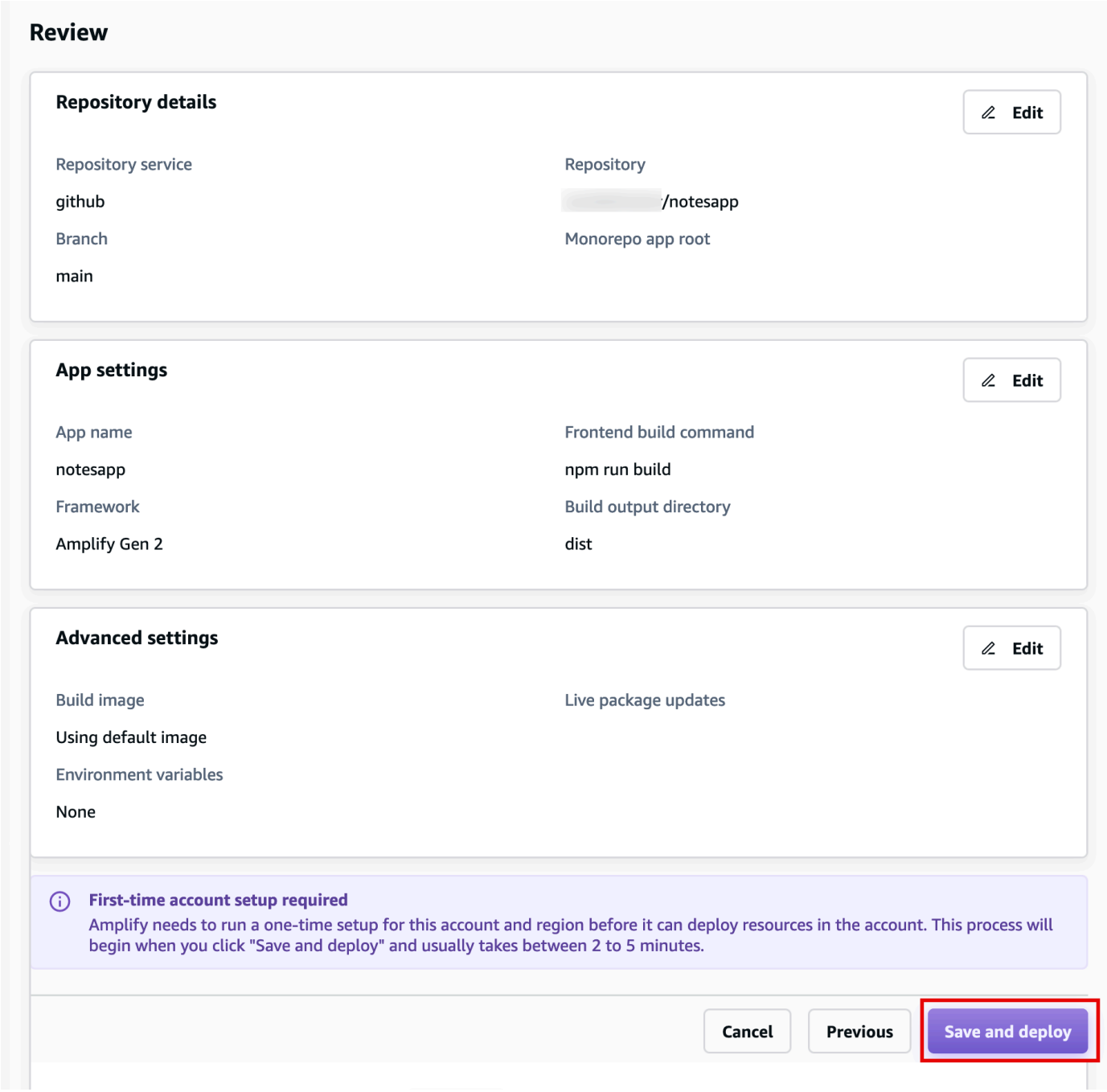
4. Configure build settings

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



5. Deploy your application

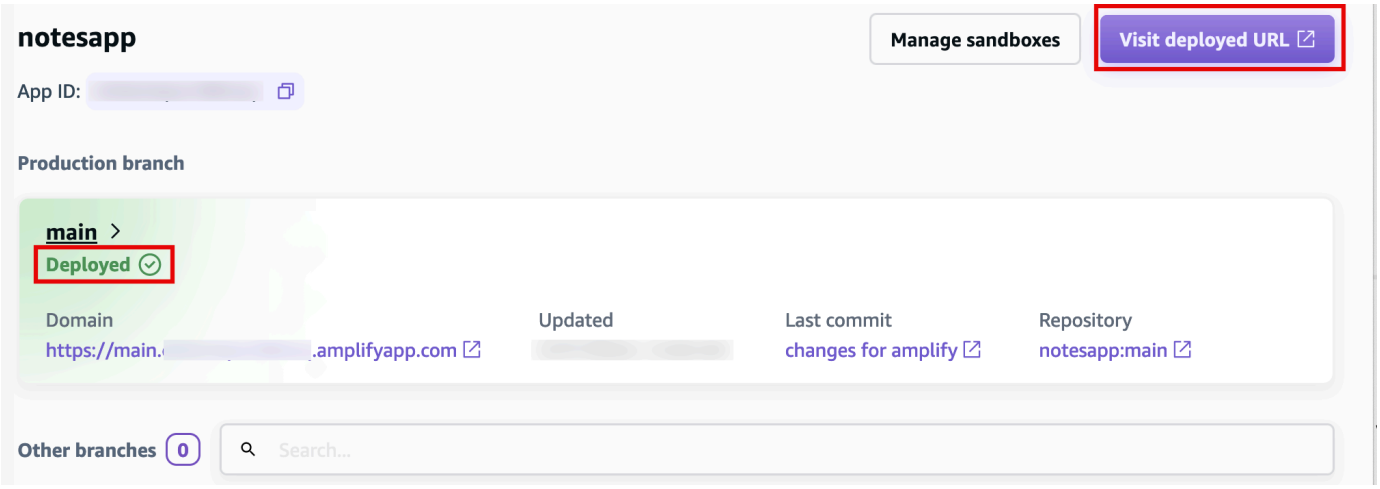
   Review the inputs selected, and choose **Save and deploy**.



6. Verify your deployment

   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.

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



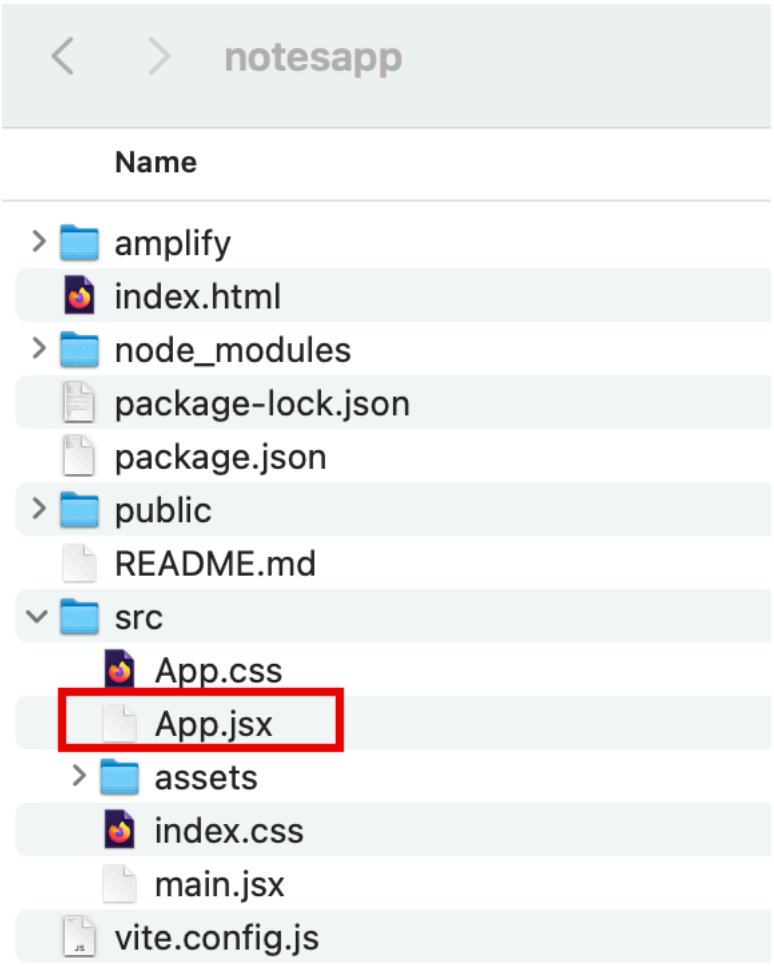
Step 5: Automatically deploy code changes

In this step, you will make some changes to the code using your text editor and push the changes to the main branch of your app.

1. Update your application code

   On your local machine, navigate to the notesapp/src/App.jsx file, and update it with the following code. Then, save the file.

```
import reactLogo from "./assets/react.svg";
import "./App.css";
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={reactLogo} className="logo react" alt="React logo" />
        <h1>Hello from Amplify</h1>
      </header>
    </div>
  );
}
export default App;
```



2. Push your code changes

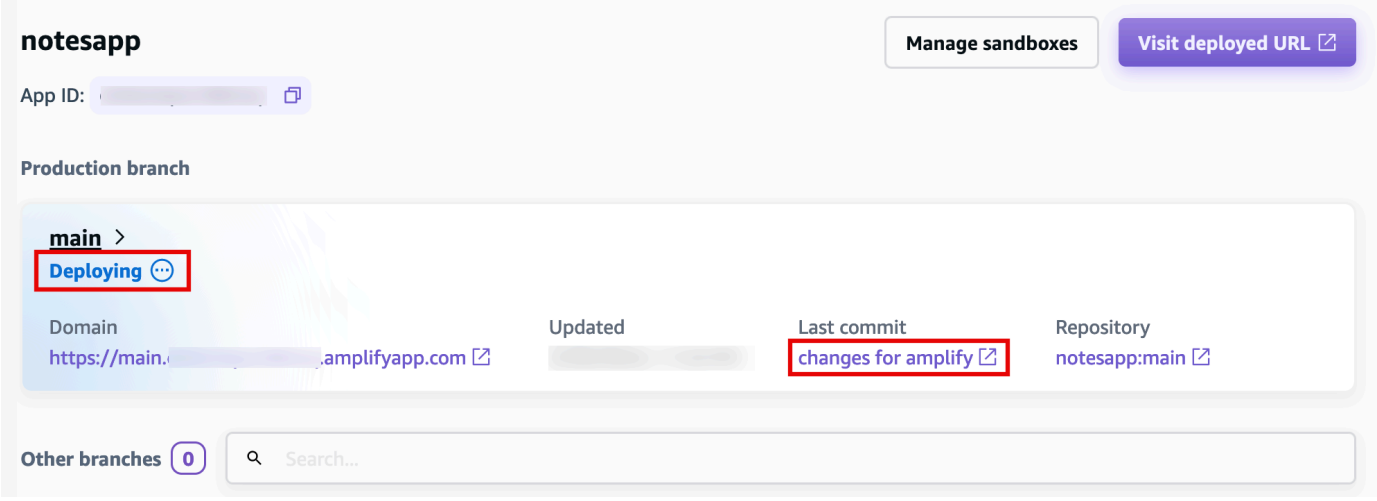
   In your terminal window, run the following command to push the changes to GitHub:

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

```
~/notesapp — aws-testing — -zsh — zsh — Basic — ttys002 — 102×34
✔ DevDependencies installed
✔ Dependencies installed
✔ Template files created
Successfully created a new project!

Welcome to AWS Amplify!
 - Get started by running npx ampx sandbox.
 - Run npx ampx help for a list of available commands.

Amplify Gen 2 collects anonymous telemetry data about general usage of the CLI. Participation is optio
nal, and you may opt-out by using npx ampx configure telemetry disable. To learn more about telemetry,
 visit https://docs.amplify.aws/gen2/reference/telemetry

\aws-testing $git add .
git commit -m 'installing amplify'
git push origin main

[main 52a761d] installing amplify
 8 files changed, 19214 insertions(+), 1138 deletions(-)
 create mode 100644 amplify/auth/resource.ts
 create mode 100644 amplify/backend.ts
 create mode 100644 amplify/data/resource.ts
 create mode 100644 amplify/package.json
 create mode 100644 amplify/tsconfig.json
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (13/13), 146.89 KiB | 7.73 MiB/s, done.
Total 13 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:janellefowler/notesapp.git
   586da6e..52a761d  main -> main
\aws-testing $
```

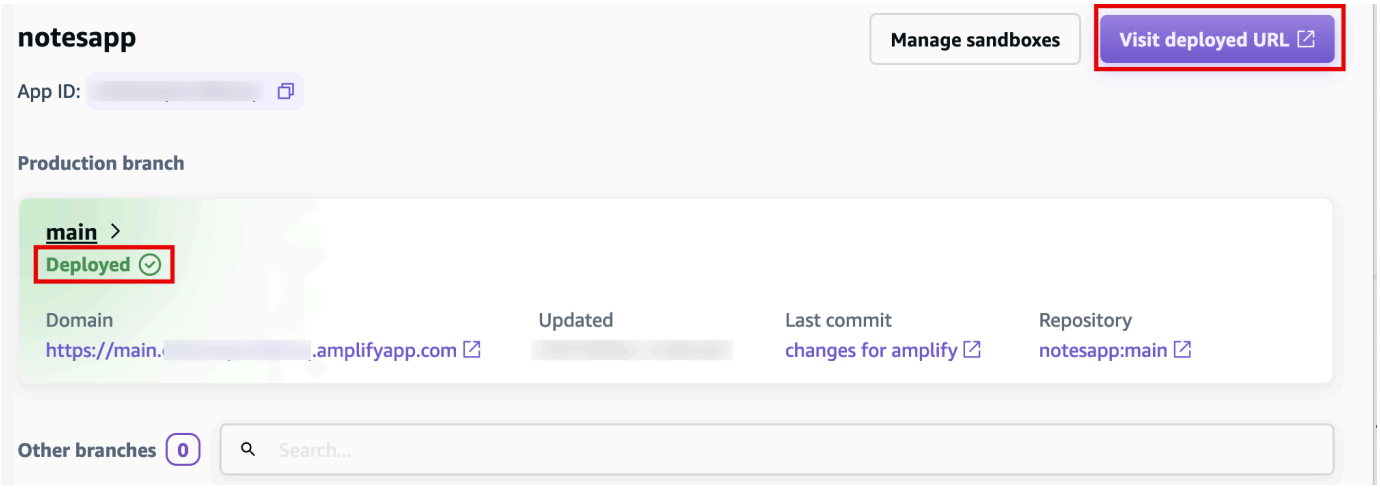
3. Update your deployed application

   AWS Amplify will now **build** your source code and **deploy** your app.



4. View your updated application

Navigate back to the Amplify console, and select the **Visit deployed URL** button to view your updated app.



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: Initialize a Local Amplify App

Time to complete	10 minutes
Requires	A text editor. Here are a few free ones: • Atom • Notepad++ • Sublime • Vim • Visual Studio Code
Get help	• Troubleshooting Amplify • How Amplify works

Overview

Now that you have a React web app, you will use AWS Amplify to configure a cloud backend for the app. AWS Amplify Gen 2 uses a fullstack TypeScript developer experience (DX) for defining backends. Amplify offers a unified developer experience with hosting, backend, and UI-building capabilities and a code-first approach.

What you will accomplish

- Set up Amplify Auth
- Set up Amplify Data
- Set up Amplify Storage

Implementation

Step 1: Set up Amplify Auth

The app uses email as the default login mechanism. When the users sign up, they receive a verification email.

- Set auth resource

  By default, your auth resource is configured as shown inside the **notesapp/amplify/auth/resource.ts** file. For this tutorial, **keep** the default auth set up as is.

resource.ts

```
import { defineAuth } from '@aws-amplify/backend';

/**
 * Define and configure your auth resource
 * @see https://docs.amplify.aws/gen2/build-a-backend/auth
 */
export const auth = defineAuth({
  loginWith: {
    email: true,
  },
});
```

Step 2: Set up Amplify Data

The app you will be building is a Notes app that will allow users to create, delete, and list notes. This example app will help you learn how to build many popular types of CRUD+L (create, read, update, delete, and list) applications.

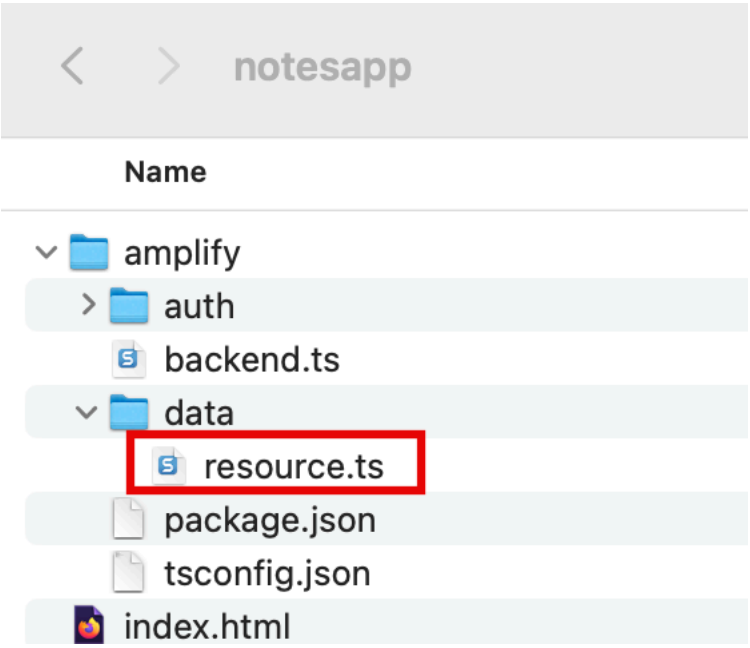
- Update authorization rule

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

  - The following updated code uses a per-owner authorization rule **allow.owner()** to restrict the note record's access to the owner of the record.
  - Amplify will automatically add an **owner: a.string()** field to each note which contains the note owner's identity information upon record creation.

```
import { type ClientSchema, a, defineData } from '@aws-amplify/backend';
const schema = a.schema({
Note: a
```

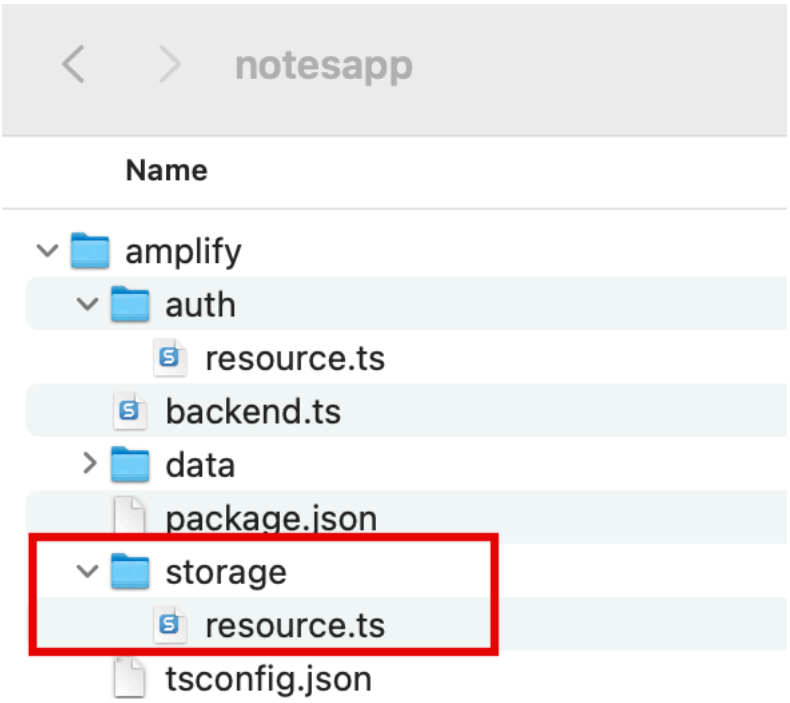
```
    .model({
      name:a.string(),
      description: a.string(),
      image: a.string(),
    })
    .authorization((allow) => [allow.owner()]),
});
export type Schema = ClientSchema<typeof schema>;
export const data = defineData({
  schema,
  authorizationModes: {
    defaultAuthorizationMode: 'userPool',
  },
});
```



Step 3: Set up Amplify Storage

1. Create a storage folder

   On your local machine, navigate to the **notesapp/amplify** folder, and **create** a new folder named **storage**, and then **create** a file named **resource.ts** inside of the new storage folder.



2. Configure a storage resource for your app

Update the **amplify/storage/resource.ts** file with the following code to configure a storage resource for your app. Then, **save** the file.

- The updated code will set up the access so that only the person who uploads the image can access. The code will use the **entity_id** as a reserved token that will be replaced with the users' identifier when the file is being uploaded.

```
import { defineStorage } from "@aws-amplify/backend";

export const storage = defineStorage({
  name: "amplifyNotesDrive",
  access: (allow) => ({
    "media/{entity_id}/*": [
      allow.entity("identity").to(["read", "write", "delete"]),
    ],
  }),
});
```

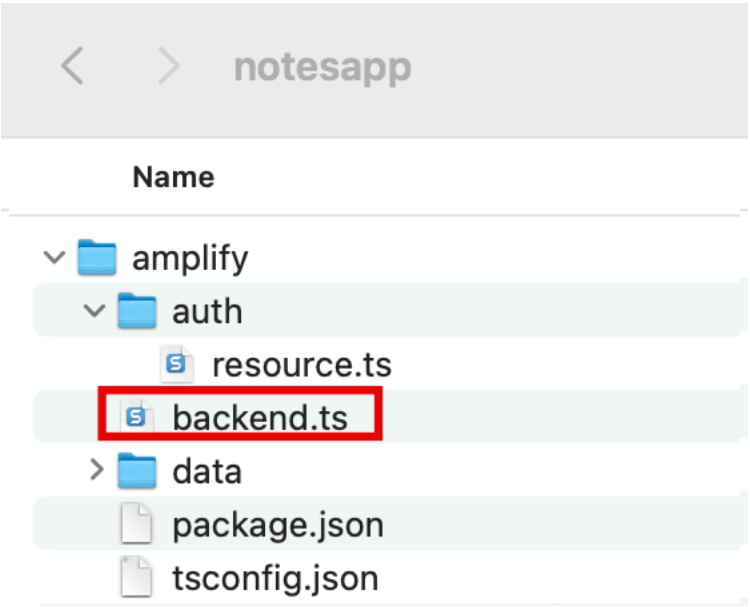
Step 4: Deploy Amplify Cloud sandbox

1. Import backend definitions

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

   - The following code will import the auth, data, and storage backend definitions:

```
import { defineBackend } from '@aws-amplify/backend';
import { auth } from './auth/resource';
import { data } from './data/resource';
import { storage } from './storage/resource';
/**
 * @see https://docs.amplify.aws/react/build-a-backend/ to add storage, functions,
 and more
 */
defineBackend({
  auth,
  data,
  storage
});
```



2. Start sandbox environment

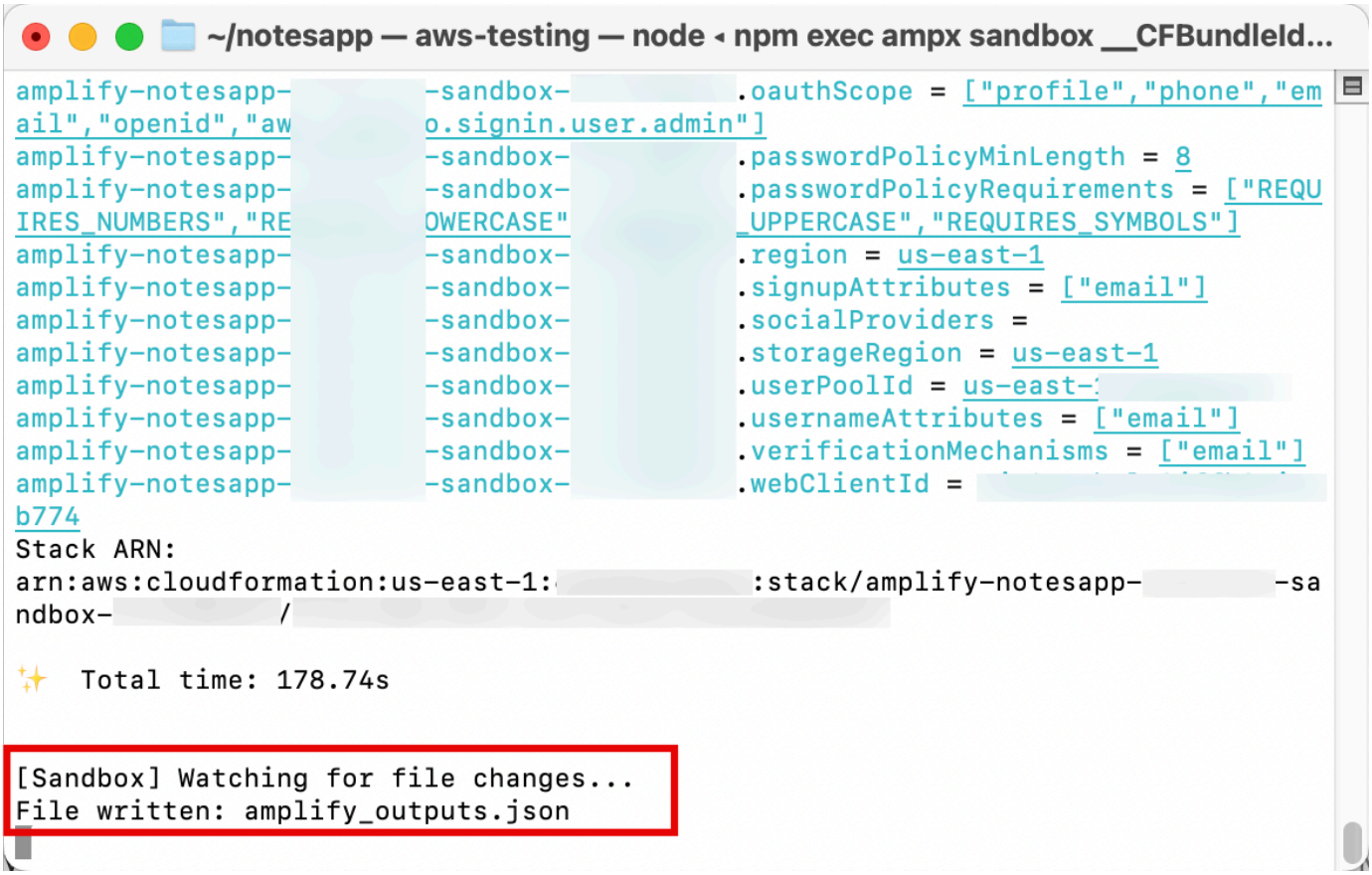
To start your own personal cloud sandbox environment that provides an isolated development space, in a new terminal window, **run** the following command in your apps root folder:

```
npx ampx sandbox
```

- The sandbox allows you to rapidly build, test, and iterate on a fullstack app. Each developer on your team can use their own disposable sandbox environment connected to cloud resources. You can learn more about it here.

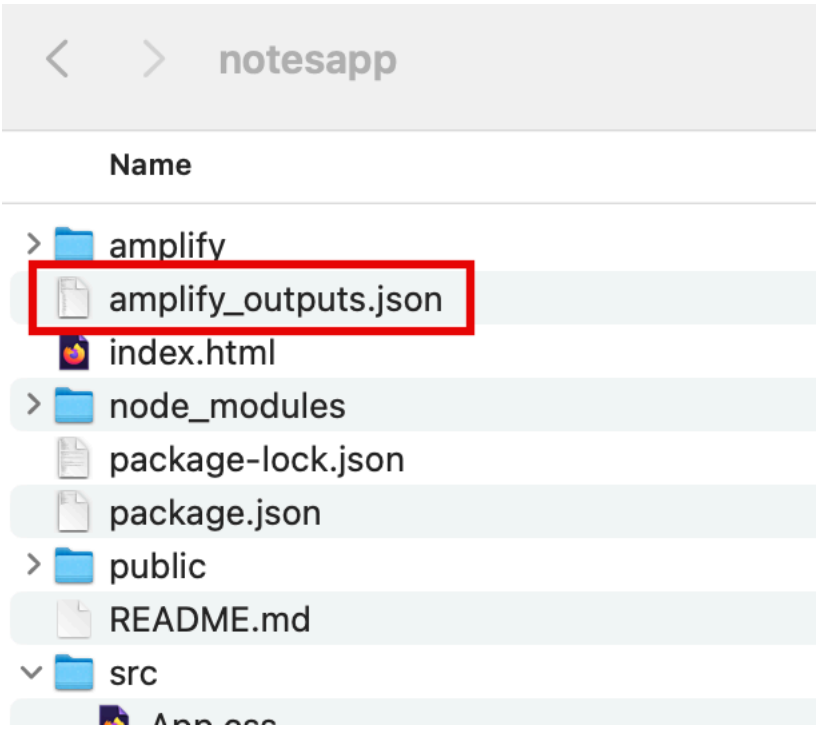
3. Confirm deployment

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



4. Verify JSON file was added

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



Conclusion

You used Amplify to configure auth, data, and storage resources. You also started your own cloud sandbox environment.

Task 3: Build the Frontend

Time to complete	10 minutes
Get help	• Troubleshooting Amplify • Troubleshooting common issues using Amplify UI

Overview

You will learn how 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 with just few lines of code. Additionally, you will build an app frontend that allows users to create, update, and delete their notes. They will also be able to upload an image and associate it with a note.

What you will accomplish

- Install Amplify libraries
- Configure your React app to include authentication, data, and storage for the Notes feature

Implementation

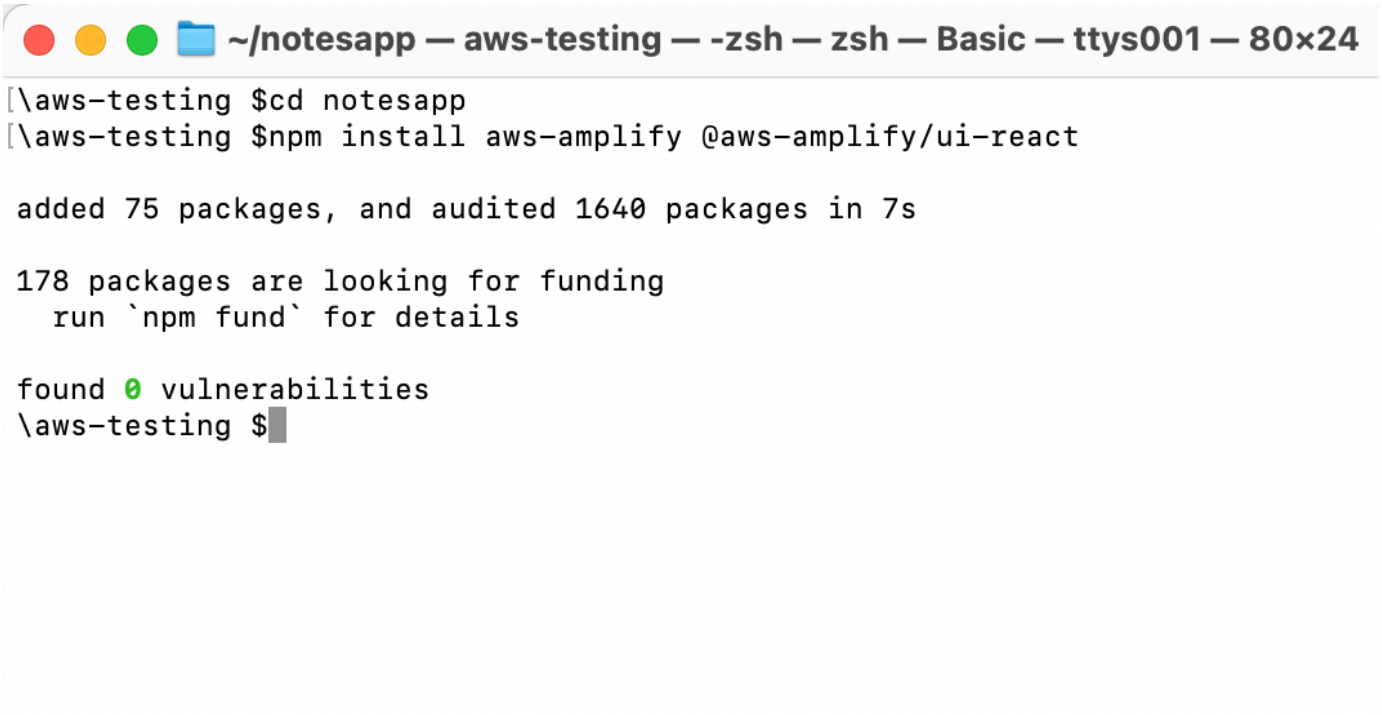
Step 1: Install the Amplify libraries

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.

- Use npm to install libraries

  Open a new terminal window, **navigate** to you projects folder (**notesapp**), and **run** the following command to install these libraries in the root of the project.

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



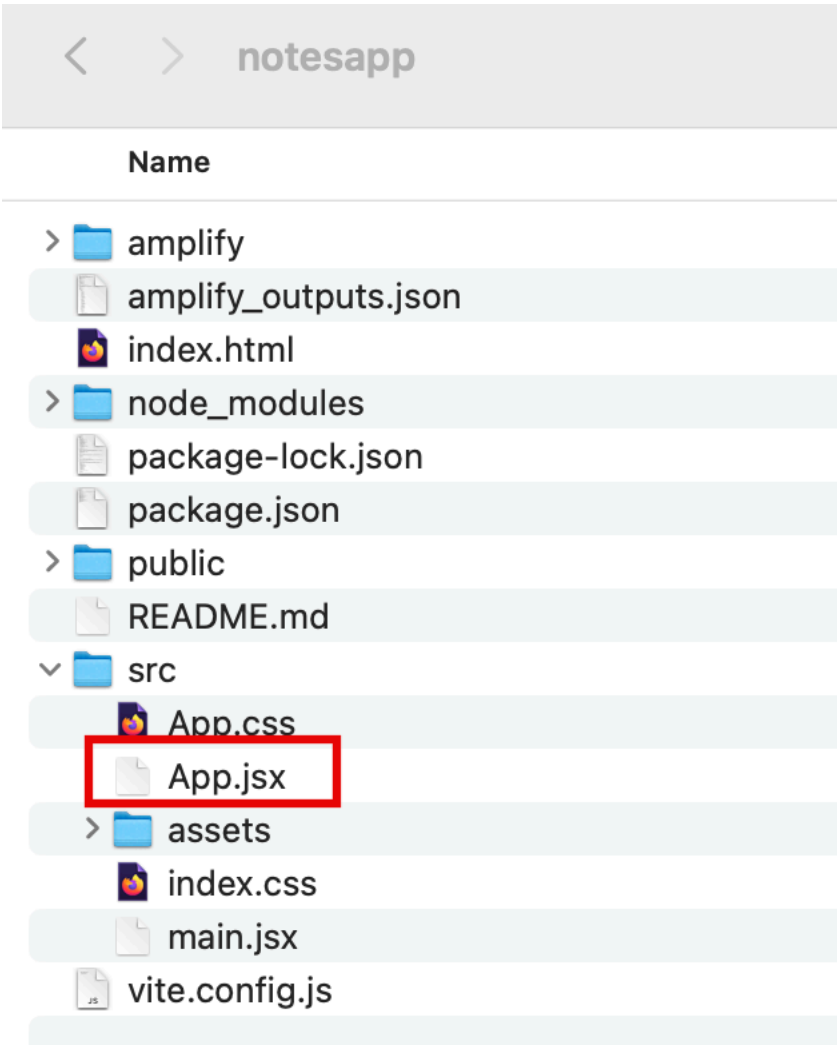
Step 2: Style the App UI

- Update the Notes UI

  On your local machine, navigate to the **notesapp/src/index.css** file, and **update** it with the following code to set the style of the Notes UI. 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 flow for Notes feature

In this step, you will update the **src/App.jsx** to configure the Amplify library with the client configuration file (**amplify_outputs.json**). Then, it will generate a data client using the **generateClient()** function. The code uses 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).

Additionally, the code contains the following:

- **fetchNotes** - Use the data client to list the items in the Notes model.

- **createNote** - Get the data from the form and use the data client to create a new note if the user selects an image. Then, the function will upload this image to Amplify storage and associate it with the new note.
- **deleteNote** - Use the data client to delete the selected note.

1. Update the Amplify Authenticator component

   On your local machine, navigate to the **notesapp/src/App.jsx** file, and **update** it with the following code.

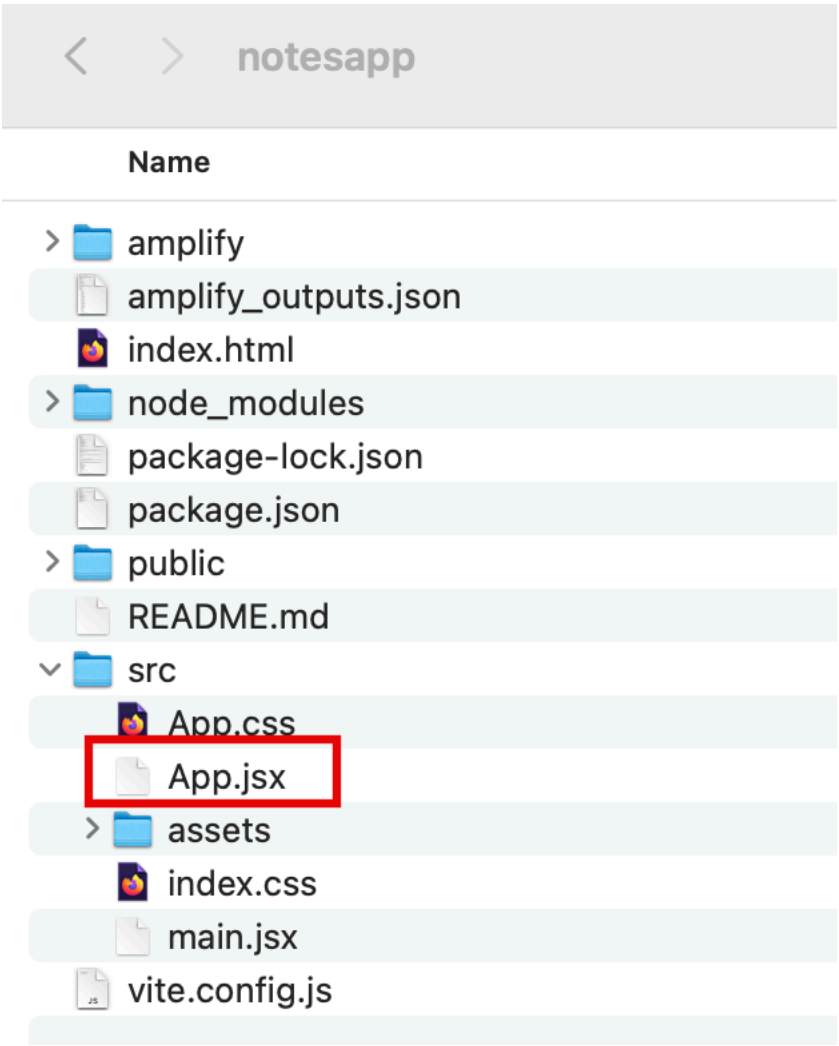
```
import { useState, useEffect } from "react";
import {
  Authenticator,
  Button,
  Text,
  TextField,
  Heading,
  Flex,
  View,
  Image,
  Grid,
  Divider,
} from "@aws-amplify/ui-react";
import { Amplify } from "aws-amplify";
import "@aws-amplify/ui-react/styles.css";
import { getUrl } from "aws-amplify/storage";
import { uploadData } from "aws-amplify/storage";
import { generateClient } from "aws-amplify/data";
import outputs from "../amplify_outputs.json";

/**
 * @type {import('aws-amplify/data').Client<import('../amplify/data/
resource').Schema>}
 */

Amplify.configure(outputs);
const client = generateClient({
  authMode: "userPool",
});

export default function App() {
  const [notes, setNotes] = useState([]);
```

```
useEffect(() => {
  fetchNotes();
}, []);

async function fetchNotes() {
  const { data: notes } = await client.models.Note.list();
  await Promise.all(
    notes.map(async (note) => {
      if (note.image) {
        const linkToStorageFile = await getUrl({
          path: ({ identityId }) => `media/${identityId}/${note.image}`,
        });
        console.log(linkToStorageFile.url);
        note.image = linkToStorageFile.url;
      }
      return note;
    })
  );
  console.log(notes);
  setNotes(notes);
}

async function createNote(event) {
  event.preventDefault();
  const form = new FormData(event.target);
  console.log(form.get("image").name);

  const { data: newNote } = await client.models.Note.create({
    name: form.get("name"),
    description: form.get("description"),
    image: form.get("image").name,
  });

  console.log(newNote);
  if (newNote.image)
    if (newNote.image)
      await uploadData({
        path: ({ identityId }) => `media/${identityId}/${newNote.image}`,
        data: form.get("image"),
      }).result;

  fetchNotes();
  event.target.reset();
```

```
  }

  async function deleteNote({ id }) {
    const toBeDeletedNote = {
      id: id,
    };

    const { data: deletedNote } = await client.models.Note.delete(
      toBeDeletedNote
    );
    console.log(deletedNote);

    fetchNotes();
  }

  return (
    <Authenticator>
      {({ signOut }) => (
        <Flex
          className="App"
          justifyContent="center"
          alignItems="center"
          direction="column"
          width="70%"
          margin="0 auto"
        >
          <Heading level={1}>My Notes App</Heading>
          <View as="form" margin="3rem 0" onSubmit={createNote}>
            <Flex
              direction="column"
              justifyContent="center"
              gap="2rem"
              padding="2rem"
            >
              <TextField
                name="name"
                placeholder="Note Name"
                label="Note Name"
                labelHidden
                variation="quiet"
                required
              />
              <TextField
                name="description"
```

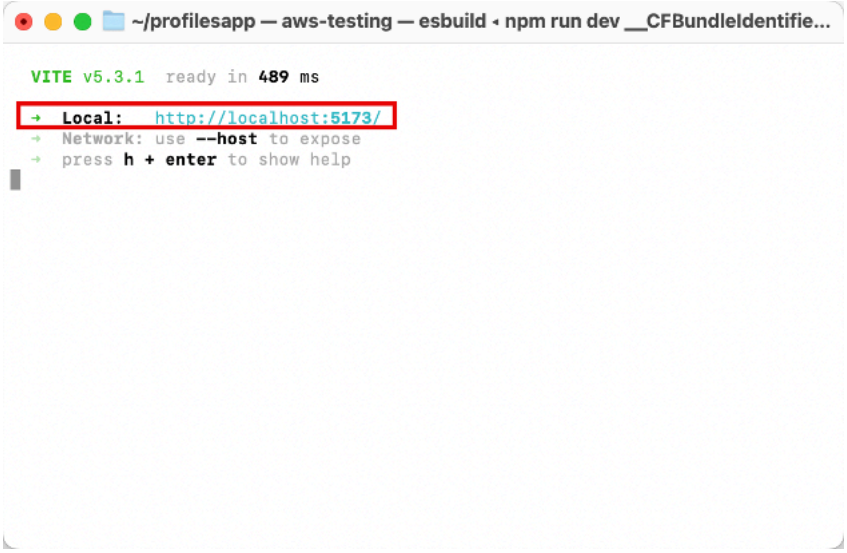
```
                placeholder="Note Description"
                label="Note Description"
                labelHidden
                variation="quiet"
                required
              />
              <View
                name="image"
                as="input"
                type="file"
                alignSelf={"end"}
                accept="image/png, image/jpeg"
              />

              <Button type="submit" variation="primary">
                Create Note
              </Button>
            </Flex>
          </View>
          <Divider />
          <Heading level={2}>Current Notes</Heading>
          <Grid
            margin="3rem 0"
            autoFlow="column"
            justifyContent="center"
            gap="2rem"
            alignContent="center"
          >
            {notes.map((note) => (
              <Flex
                key={note.id || note.name}
                direction="column"
                justifyContent="center"
                alignItems="center"
                gap="2rem"
                border="1px solid #ccc"
                padding="2rem"
                borderRadius="5%"
                className="box"
              >
                <View>
                  <Heading level="3">{note.name}</Heading>
                </View>
                <Text fontStyle="italic">{note.description}</Text>
```

```
                {note.image && (
                  <Image
                    src={note.image}
                    alt={`visual aid for ${notes.name}`}
                    style={{ width: 400 }}
                  />
                )}
                <Button
                  variation="destructive"
                  onClick={() => deleteNote(note)}
                >
                  Delete note
                </Button>
              </Flex>
            ))}
          </Grid>
          <Button onClick={signOut}>Sign Out</Button>
        </Flex>
      )}
    </Authenticator>
  );
}
```



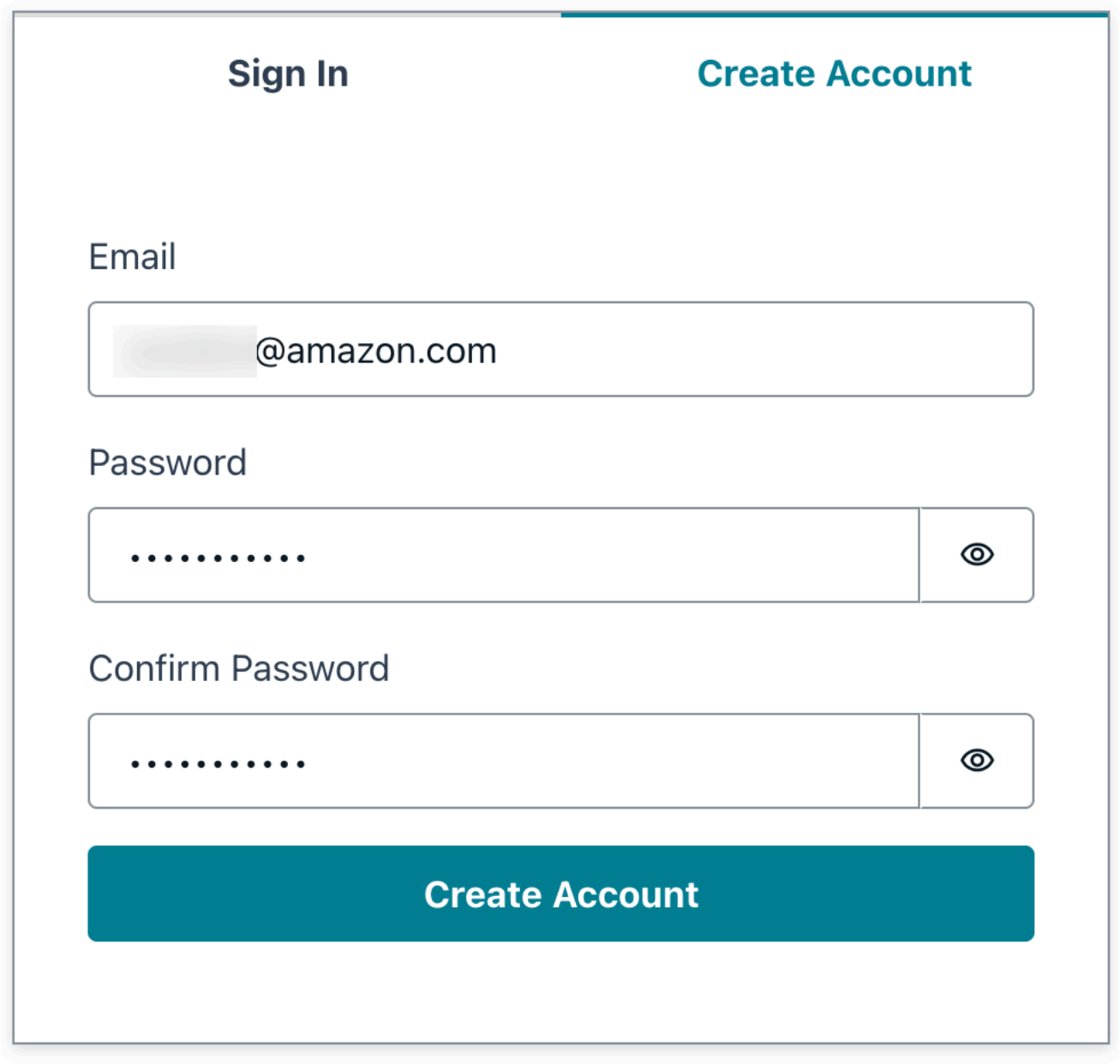
2. Open the app

   Select the **Local host link** to open the application.



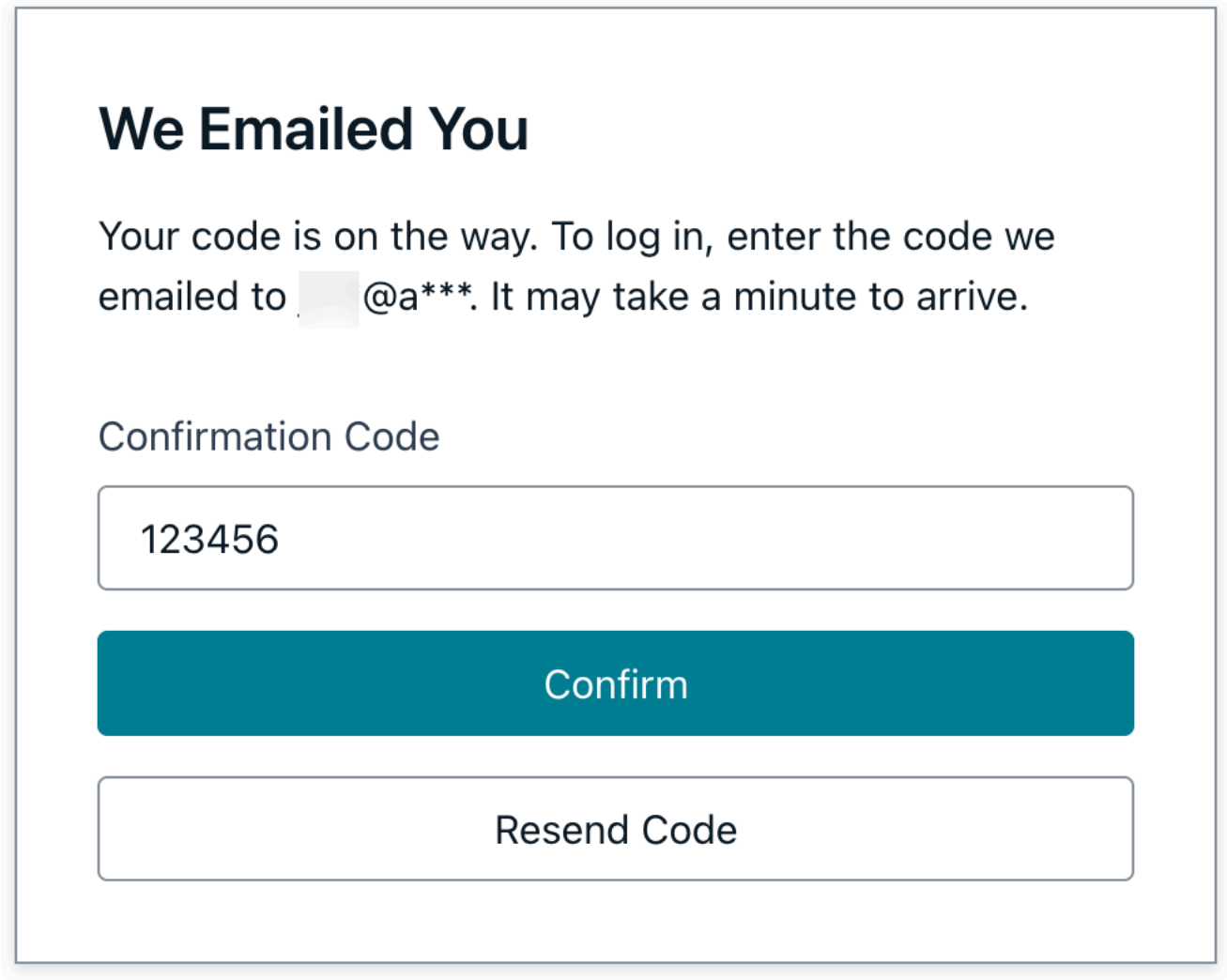
3. Create an account

   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**.



4. Get verification code

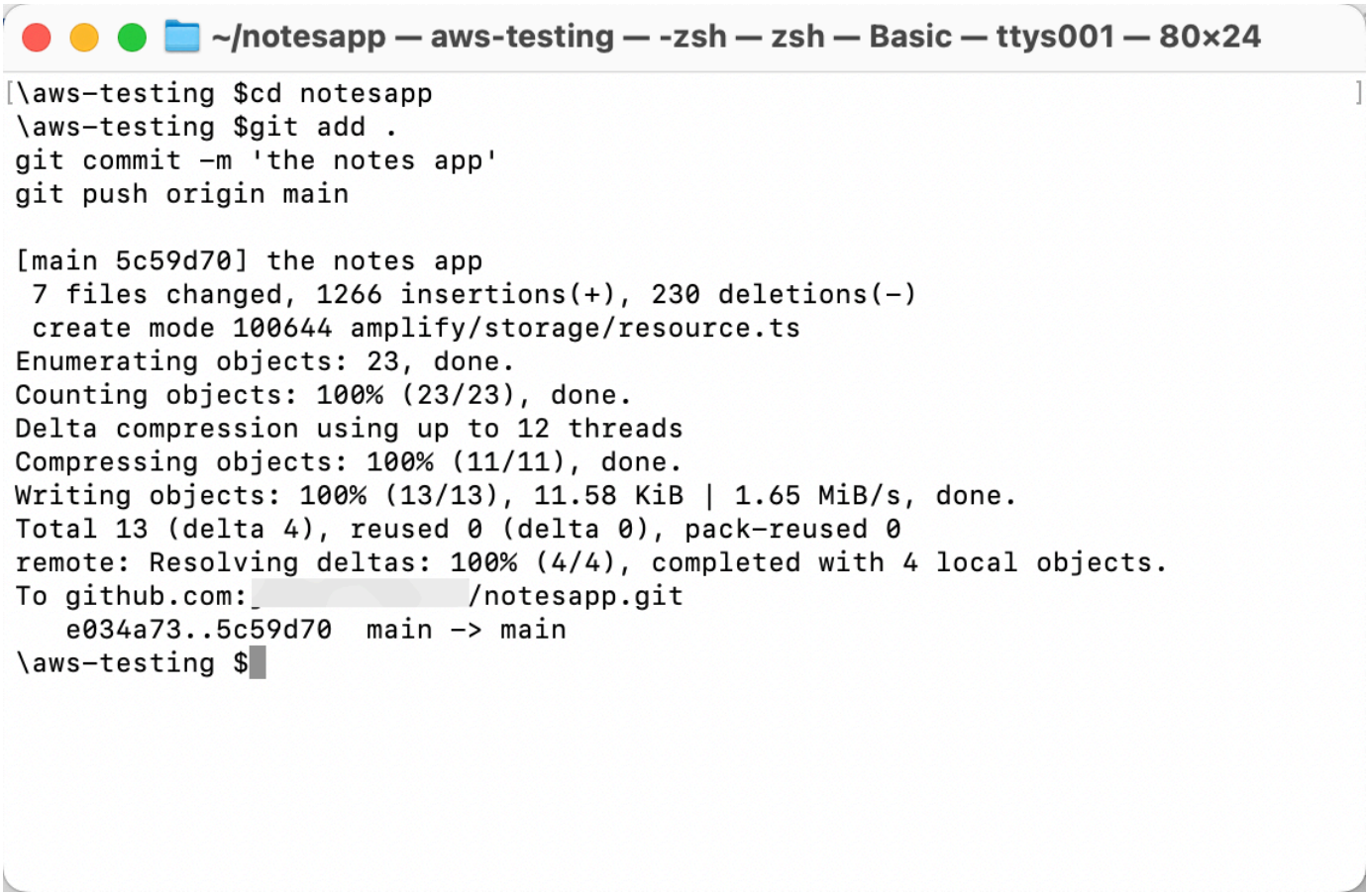
   You will get a verification code sent to your email. Enter the **verification code** to log in to the app. When signed in, you can start creating notes and delete them.



5. Push changes to GitHub

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

```
git add .
git commit -m 'the notes app'
git push origin main
```

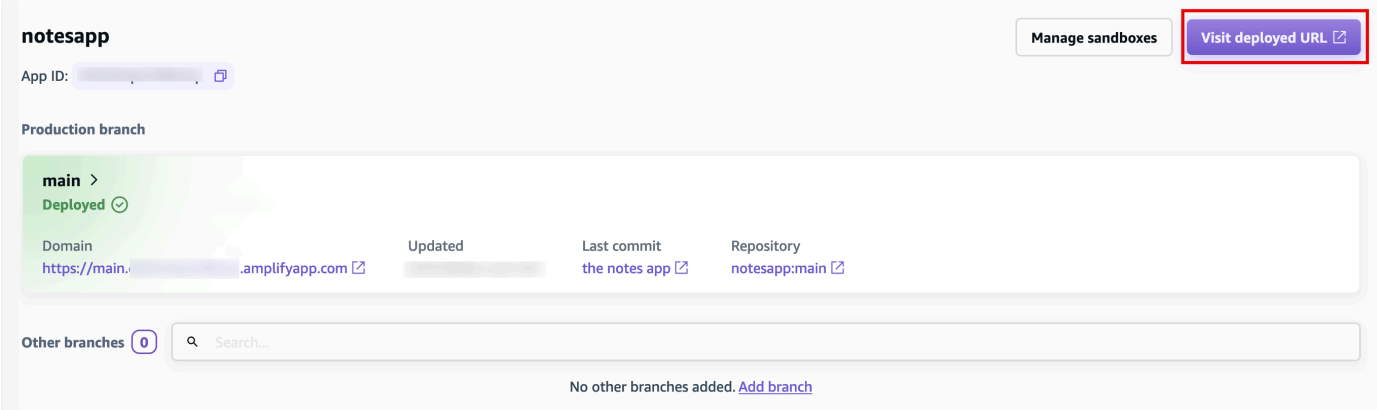


6. Sign into the console

   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.

7. View your web app

   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.



Congratulations

You have now connected your App to the Amplify backend and built a frontend allowing the users to create, edit, and delete notes. Users will also be able to upload an image and associate it with a note.

Task 4: Clean up resources

Time to complete	< 2 minutes

Overview

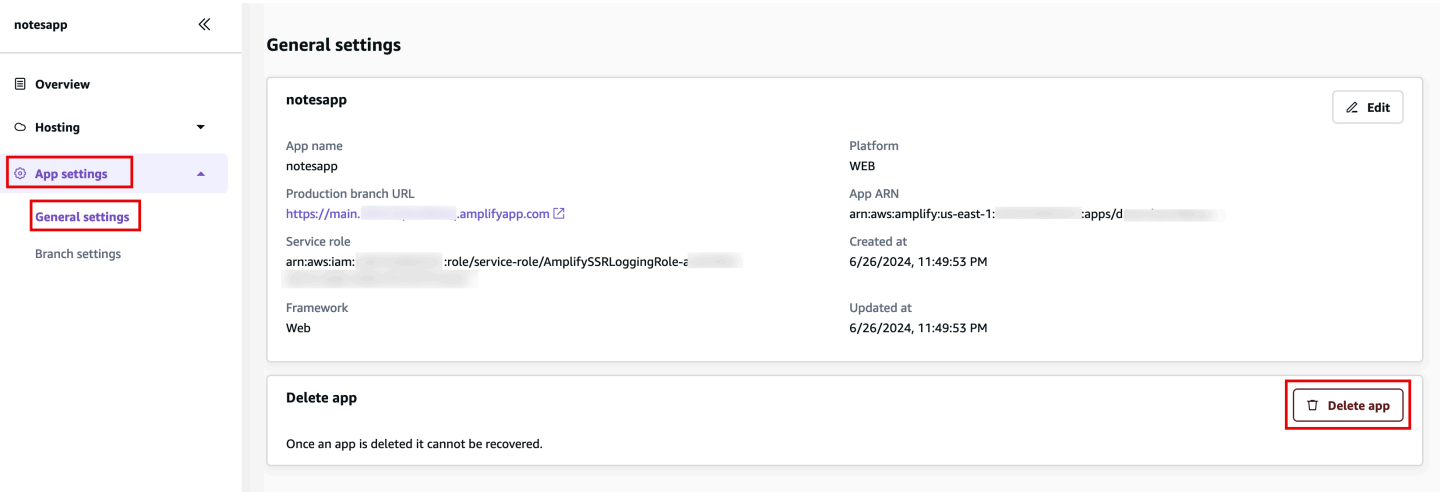
In this task, you will go through the steps to delete all the resources you created throughout this tutorial. It is a best practice to delete resources you are no longer using to avoid unwanted charges.

Implementation

Step 1: Delete the app

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

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



Conclusion

You have created a React web app and used AWS Amplify to host it on AWS! You provisioned a cloud backend and used Amplify to configure auth, data, and storage resources. Additionally, you created a frontend allowing users to create, read, update, and delete their notes. You have also enabled them to select and upload an image to associate it with their notes.