

Hands-on tutorials

Deploy a Web App on AWS Amplify



Deploy a Web App on AWS Amplify: Hands-on tutorials

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Deploy a Web App on AWS Amplify	i
Overview	1
What you will accomplish	1
Prerequisites	1
Tasks	1
Task 1: Create a new Amplify Project	3
Overview	1
What you will accomplish	1
Implementation	3
Conclusion	7
Task 2: Initialize the Amplify Backend	8
Overview	1
What you will accomplish	1
Implementation	3
Conclusion	7
Task 3: Build the Frontend	15
Overview	1
What you will accomplish	1
Implementation	3
Conclusion	7
Task 4: Deploy the App	28
Overview	1
What you will accomplish	1
Implementation	3
Task 5: Clean up Resources	35
Overview	1
Implementation	3
Congratulations	35

Deploy a Web App on AWS Amplify

Overview

In this tutorial, you will learn how to deploy a web application with 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. With fullstack TypeScript capabilities, Amplify brings the power and breadth of AWS services to a familiar frontend developer experience. 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.

What you will accomplish

In this tutorial, you will learn how to:

- Create your first Amplify project
- Use Amplify to add authentication, a real-time API, and a database to your application
- Use Amplify libraries to connect the frontend with AWS services
- Deploy a web app with AWS Amplify

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.

Tasks

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

1. [Task 1: Create a new Amplify Project](#) (5 mins): Create a new application with Amplify.
2. [Task 2: Initialize the Amplify Backend](#) (10 mins): Initialize the AWS Amplify backend for your app.
3. [Task 3: Build the Frontend](#) (5 mins): Connect the app to the backend.
4. [Task 4: Deploy the App](#): (10 mins): Deploy your new application.
5. [Task 5: Clean up Resources](#): (2 mins): Clean up resources.

Task 1: Create a new Amplify Project

Time to complete	5 minutes
Requires	<ul style="list-style-type: none">• A GitHub account• GitHub SSH connection• Nodejs and npm
Get help	Troubleshooting Amplify

Overview

In this task, you will create a new web application using [React](#), a JavaScript library for building user interfaces, and learn how to configure AWS Amplify for your first project.

What you will accomplish

- Create a new web application
- Set up Amplify on your project

Implementation

Step 1: Create a new Amplify Project

1. Check environment

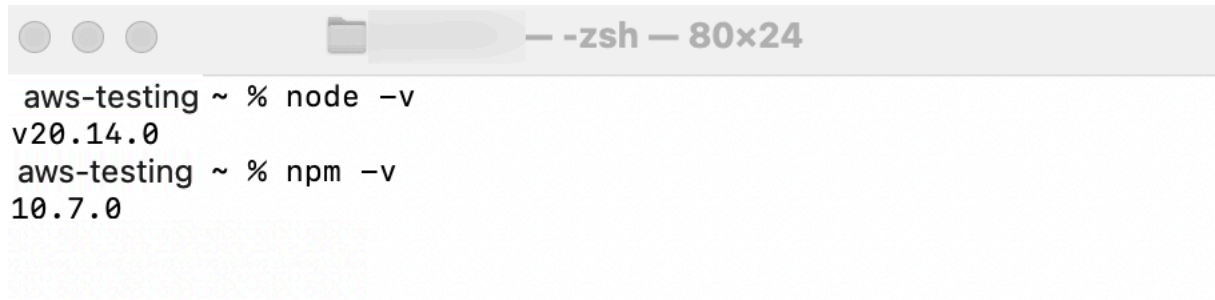
In a new terminal window or command line, **run** the following commands to verify that you are running at least Node.js version 18.16.0 and npm version 6.14.4 or greater.

- If you are not running these versions, visit the [nodejs](#) and [npm website](#) for more information.

Note

Your output might differ based on the version installed.

```
node -v
npm -v
```

A terminal window titled "-- zsh -- 80x24" showing the output of version checks. The prompt is "aws-testing ~ %". The first command is "node -v" which outputs "v20.14.0". The second command is "npm -v" which outputs "10.7.0".

```
aws-testing ~ % node -v
v20.14.0
aws-testing ~ % npm -v
10.7.0
```

2. Create a new React application

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

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

```
aws-testing584962325871 ~ % npm create vite@latest expensetracker --template react __CFBu...
aws-testing584962325871 ~ % npm create vite@latest expensetracker -- --template re
act
cd expensetracker
npm install
npm run dev

Need to install the following packages:
create-vite@5.2.3
Ok to proceed? (y) y
```

3. Open the local development server

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

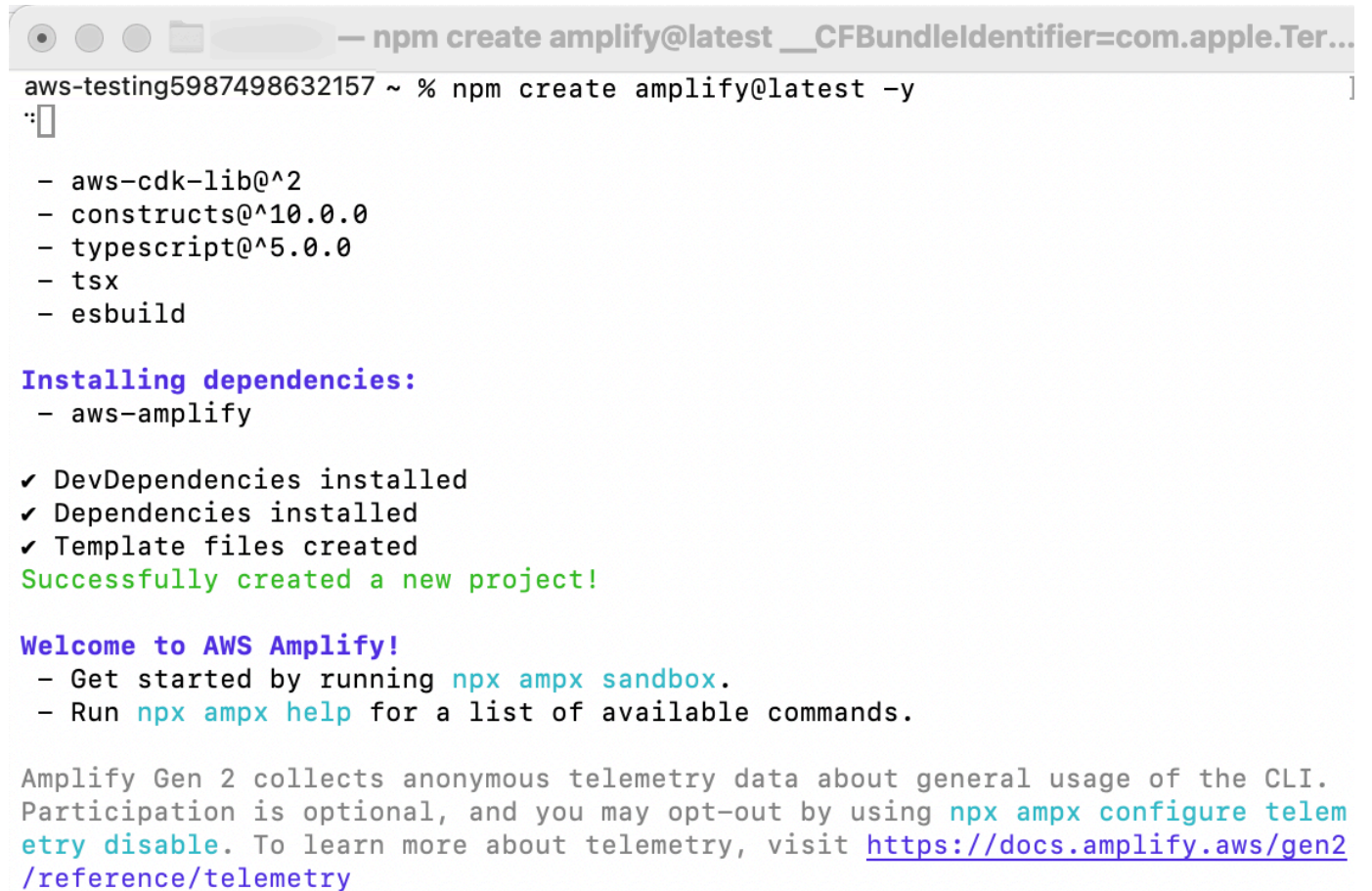
```
esbuild < npm run dev __CFBundleIdentifier=com.apple.Terminal.

VITE v5.2.13 ready in 192 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

4. Install Amplify CLI

Open a new terminal window, **navigate** to your projects root folder (**expensetracker**), and **run** the following command:

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



```
aws-testing5987498632157 ~ % npm create amplify@latest -y
:::

- aws-cdk-lib@^2
- constructs@^10.0.0
- typescript@^5.0.0
- tsx
- esbuild

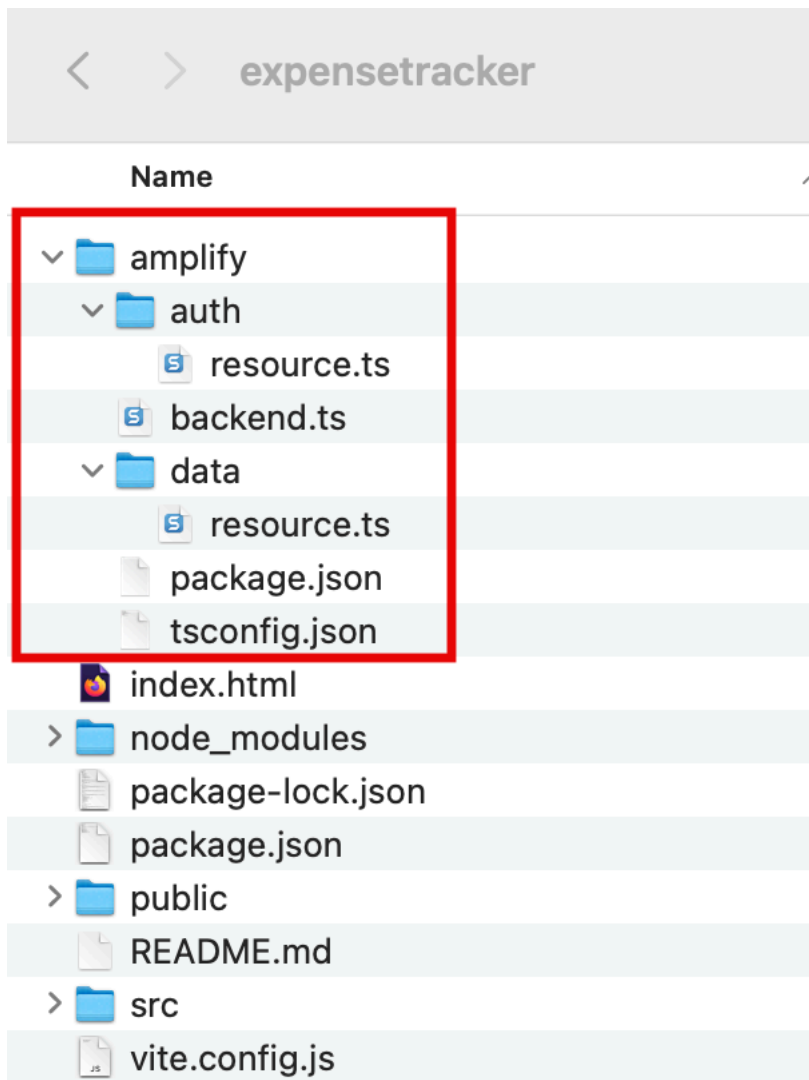
Installing dependencies:
- aws-amplify

✓ 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 optional, and you may opt-out by using npx ampx configure telem
etry disable. To learn more about telemetry, visit https://docs.amplify.aws/gen2/reference/telemetry
```

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



Conclusion

In this task, you learned how to create a React frontend application, and installed the amplify packages in preparation to configure a backend for the app.

Task 2: Initialize the Amplify Backend

Time to complete

10 minutes

Requires

- AWS profile [configured](#) for local development
- A text editor. Here are a few free ones:
 - [Atom](#)
 - [Notepad++](#)
 - [Sublime](#)
 - [Vim](#)
 - [Visual Studio Code](#)

Get help

[Troubleshooting Amplify](#)

Overview

In this task 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.

The app that you build in this tutorial is an expense tracker app that will allow users to create, delete, and list expenses. This example app is a starting point to learn how to build many popular types of CRUD+L (create, read, update, delete, and list) applications.

What you will accomplish

In this task, you will:

- Set up Amplify Authentication
- Set up Amplify Data

Implementation

Step 1: Set up Amplify Auth

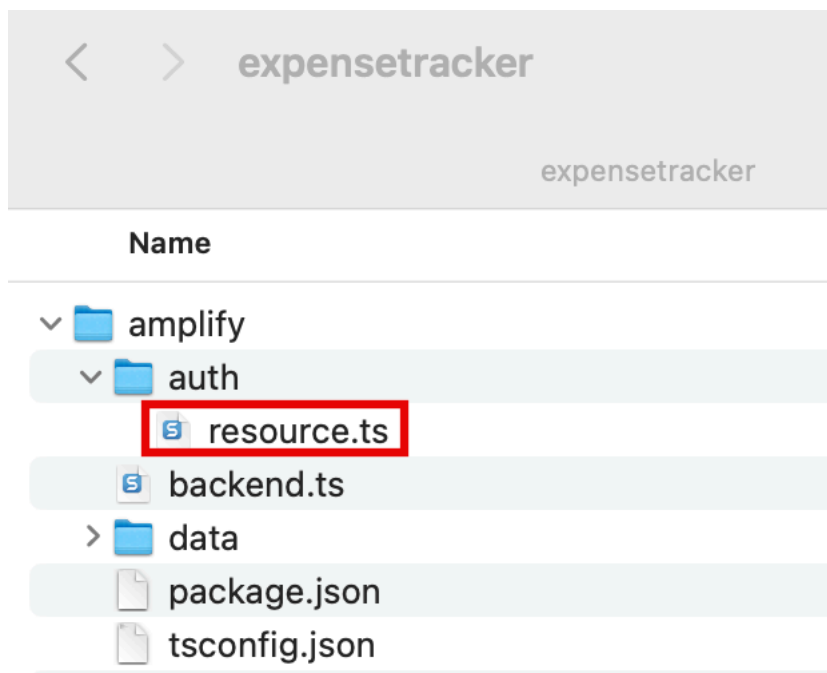
The app uses email as the default login mechanism. When the users sign up, they receive a verification email. In this step, you will customize the verification email.

1. Update the resource file

On your local machine, navigate to the **amplify/auth/resource.ts** file, and use the following code to customize the verification email. Then, **save** the file.

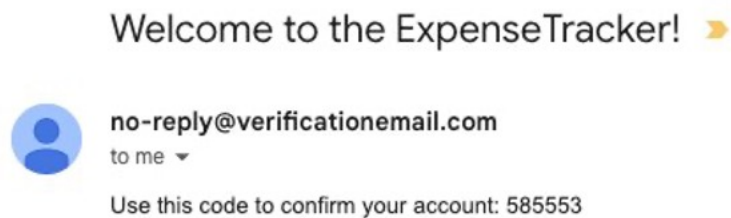
```
import { defineAuth } from "@aws-amplify/backend";

export const auth = defineAuth({
  loginWith: {
    email: {
      verificationEmailStyle: "CODE",
      verificationEmailSubject: "Welcome to the ExpenseTracker!",
      verificationEmailBody: (createCode) =>
        `Use this code to confirm your account: ${createCode()}`,
    },
  },
});
```



2. View the customized email

This image shows an example of the customized verification email.



Step 2: Set up Amplify Data

In this step, you will define the schema for the Expense data model, and use a per-owner authorization rule **allow.owner()** to restrict the expense record's access to the owner of the record. Amplify will automatically add a **owner: a.string()** field to each expense which contains the expense owner's identity information upon record creation.

- Update the resource file

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

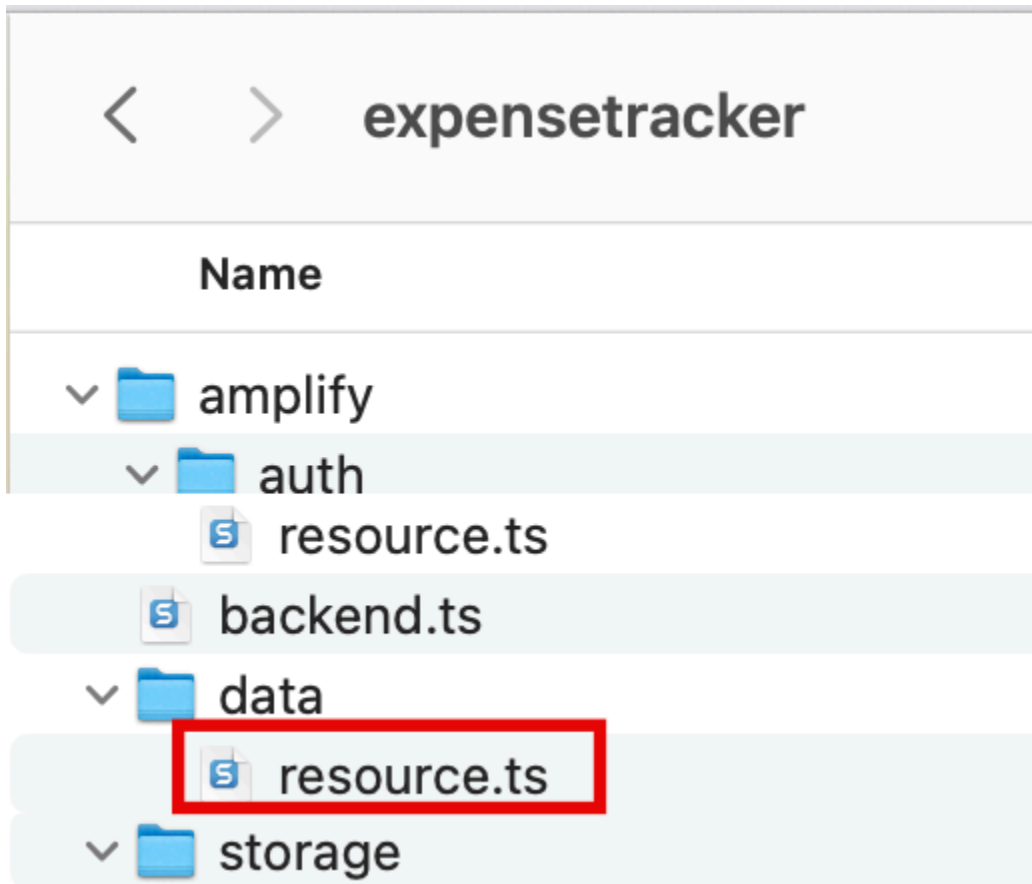
```
import { type ClientSchema, a, defineData } from '@aws-amplify/backend';

const schema = a.schema({
  Expense: a
    .model({
      name: a.string(),
      amount: a.float(),
    })
    .authorization((allow) => [allow.owner()]),
});

export type Schema = ClientSchema
<typeof schema>
;

export const data = defineData({
  schema,
  authorizationModes: {
```

```
defaultAuthorizationMode: 'userPool',  
  },  
});
```



Step 3: Deploy Amplify Cloud sandbox

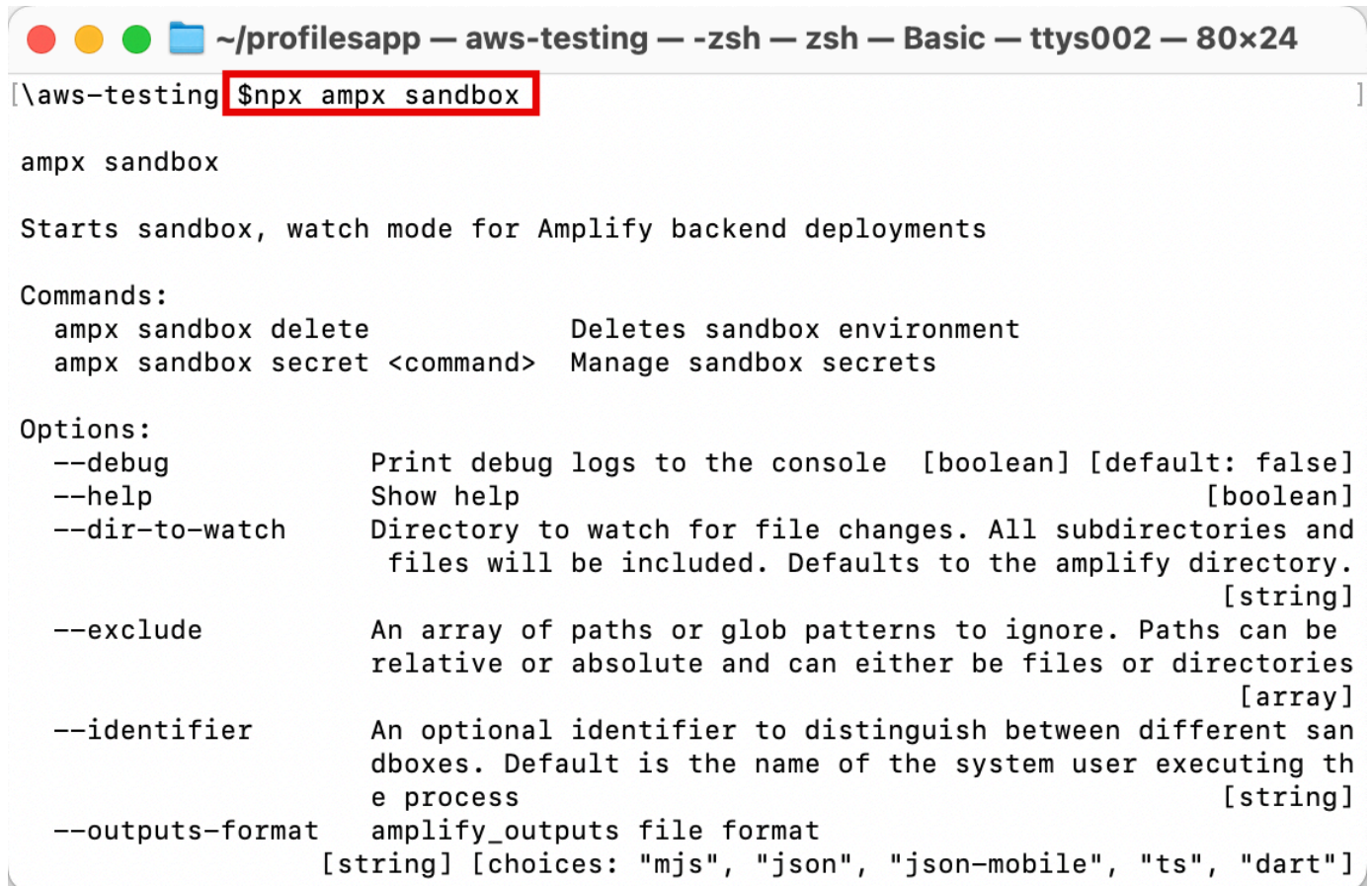
Note

The **amplify/backend.ts** file is already configured to import the auth and data backend definitions. You don't need to change it.

1. Deploy sandbox

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

```
npx ampx sandbox
```



```
~/profilesapp — aws-testing — -zsh — zsh — Basic — ttys002 — 80x24
[\aws-testing] $npx ampx sandbox
ampx sandbox

Starts sandbox, watch mode for Amplify backend deployments

Commands:
  ampx sandbox delete           Deletes sandbox environment
  ampx sandbox secret <command> Manage sandbox secrets

Options:
  --debug          Print debug logs to the console [boolean] [default: false]
  --help          Show help [boolean]
  --dir-to-watch  Directory to watch for file changes. All subdirectories and
                  files will be included. Defaults to the amplify directory.
                  [string]
  --exclude       An array of paths or glob patterns to ignore. Paths can be
                  relative or absolute and can either be files or directories
                  [array]
  --identifier    An optional identifier to distinguish between different san
                  dboxes. Default is the name of the system user executing th
                  e process [string]
  --outputs-format amplify_outputs file format
                  [string] [choices: "mjs", "json", "json-mobile", "ts", "dart"]
```

2. View confirmation message

After the cloud sandbox has been fully deployed, your terminal will display a **confirmation message**. This deployment will take several minutes to complete.

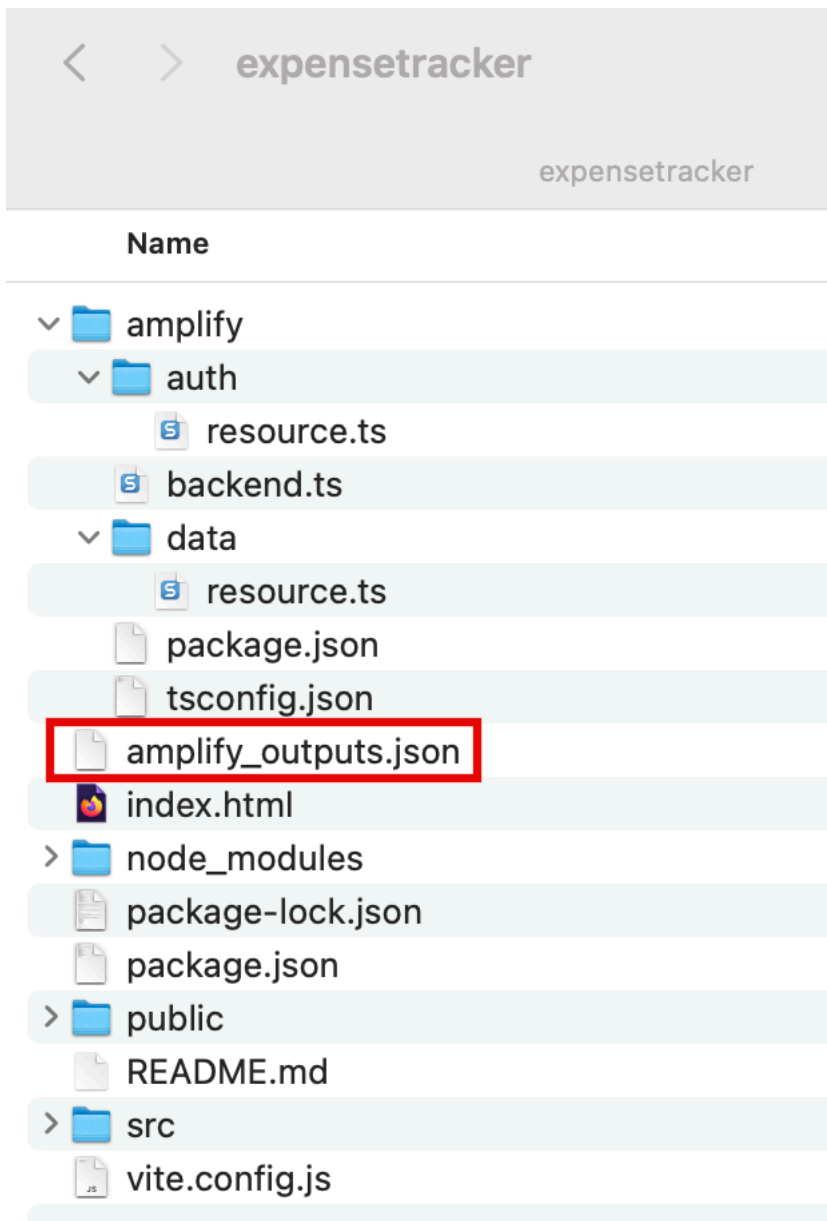
```
~/profilesapp — aws-testing — node < npm exec ampx sandbox __CFBundleIdentifier...
amplify-profilesapp- -sandbox-16ba761c9c.oauthRedirectSignOut =
amplify-profilesapp- -sandbox-16ba761c9c.oauthResponseType = code
amplify-profilesapp- -sandbox-16ba761c9c.oauthScope = ["profile","phone","email","o
penid","aws.cognito. ser.admin"]
amplify-profilesapp- -sandbox-16ba761c9c.passwordPolicyMinLength = 8
amplify-profilesapp- -sandbox-16ba761c9c.passwordPolicyRequirements = ["REQUIRES_NU
MBERS","REQUIRES_LOW "REQUIRES_UPPERCASE","REQUIRES_SYMBOLS"]
amplify-profilesapp- -sandbox-16ba761c9c.region = us-east-1
amplify-profilesapp- -sandbox-16ba761c9c.signupAttributes = ["email"]
amplify-profilesapp- -sandbox-16ba761c9c.socialProviders =
amplify-profilesapp- -sandbox-16ba761c9c.userPoolId = us-east-1_k6GaOT9pW
amplify-profilesapp- -sandbox-16ba761c9c.usernameAttributes = ["email"]
amplify-profilesapp- -sandbox-16ba761c9c.verificationMechanisms = ["email"]
amplify-profilesapp- -sandbox-16ba761c9c.webClientId = 2sdo9 cg3dq1g
Stack ARN:
arn:aws:cloudformation:us-east-1: :stack/amplify-profilesapp- -sandbox-1
6ba761c9c/381d7e30- 153

✨ Total time: 192.38s

[Sandbox] Watching for file changes...
File written: amplify_outputs.json
```

3. Verify output file

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



Conclusion

In this task, you used Amplify to configure auth and data resources. You also started your own cloud sandbox environment. In the next module, you will connect your app's frontend to your backend and build app features.

Task 3: Build the Frontend

Time to complete

5 minutes

Get help

[Troubleshooting Amplify](#)

Overview

In this task, you will learn how to use the Amplify UI component library to scaffold out an entire user authentication flow, allow 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 expenses.

What you will accomplish

In this task, you will:

- Install the Amplify client libraries
- Configure your React app to include authentication, data, and storage for the expenses 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 our app's frontend to our backend, and the **@aws-amplify/ui-react** library contains framework-specific UI components.

- Install the libraries

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

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

```
~/expensetracker — aws-testing — -zsh — zsh — Basic — ttys001 — 80x24
[\aws-testing $cd expensetracker ]
[\aws-testing $npm install aws-amplify @aws-amplify/ui-react ]

added 75 packages, and audited 1642 packages in 10s

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

found 0 vulnerabilities
\aws-testing $
```

Step 2: Style the App UI

- Navigate to the CSS

On your local machine, navigate to the **expensetracker/src/index.css** file. Replace your CSS with the following CSS:

```
: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);

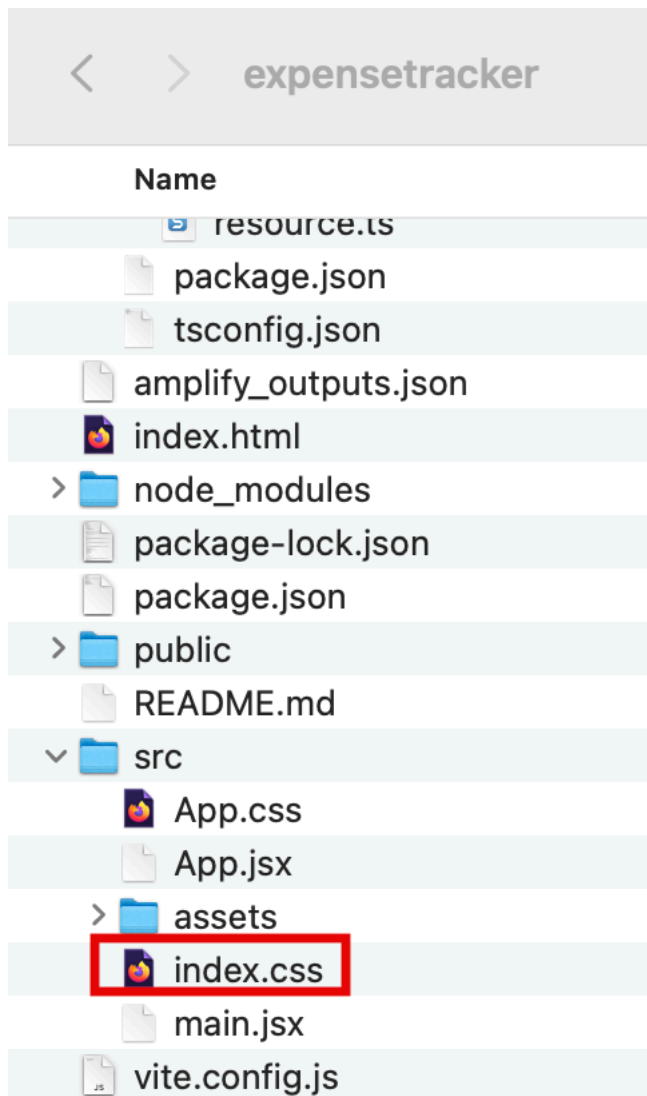
  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
```

```
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 Expenses feature

In this step, you will update the `src/App.jsx` file 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:

- A code to use a real-time `observeQuery` to subscribe to a live feed of the user's expenses data.

- **createExpense** - Get the data from the form and use the data client to create a new expense.
- **deleteExpense** - Use the data client to delete the selected expense.

1. Update the App.jsx file

On your local machine, navigate to the **expensetracker/src/app.jsx** file.

Replace the code in the App.jsx with the following code from [this file](#). Then, save the updated file.

```
import { useState, useEffect } from "react";
import {
  Authenticator,
  Button,
  Text,
  TextField,
  Heading,
  Flex,
  View,
  Grid,
  Divider,
} from "@aws-amplify/ui-react";
import { Amplify } from "aws-amplify";
import "@aws-amplify/ui-react/styles.css";
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 [expenses, setExpenses] = useState([]);

  useEffect(() => {
    client.models.Expense.observeQuery().subscribe({
```

```
    next: (data) => setExpenses([...data.items]),
  });
}, []);

async function createExpense(event) {
  event.preventDefault();
  const form = new FormData(event.target);

  await client.models.Expense.create({
    name: form.get("name"),
    amount: form.get("amount"),
  });

  event.target.reset();
}

async function deleteExpense({ id }) {
  const toBeDeletedExpense = {
    id,
  };

  await client.models.Expense.delete(toBeDeletedExpense);
}

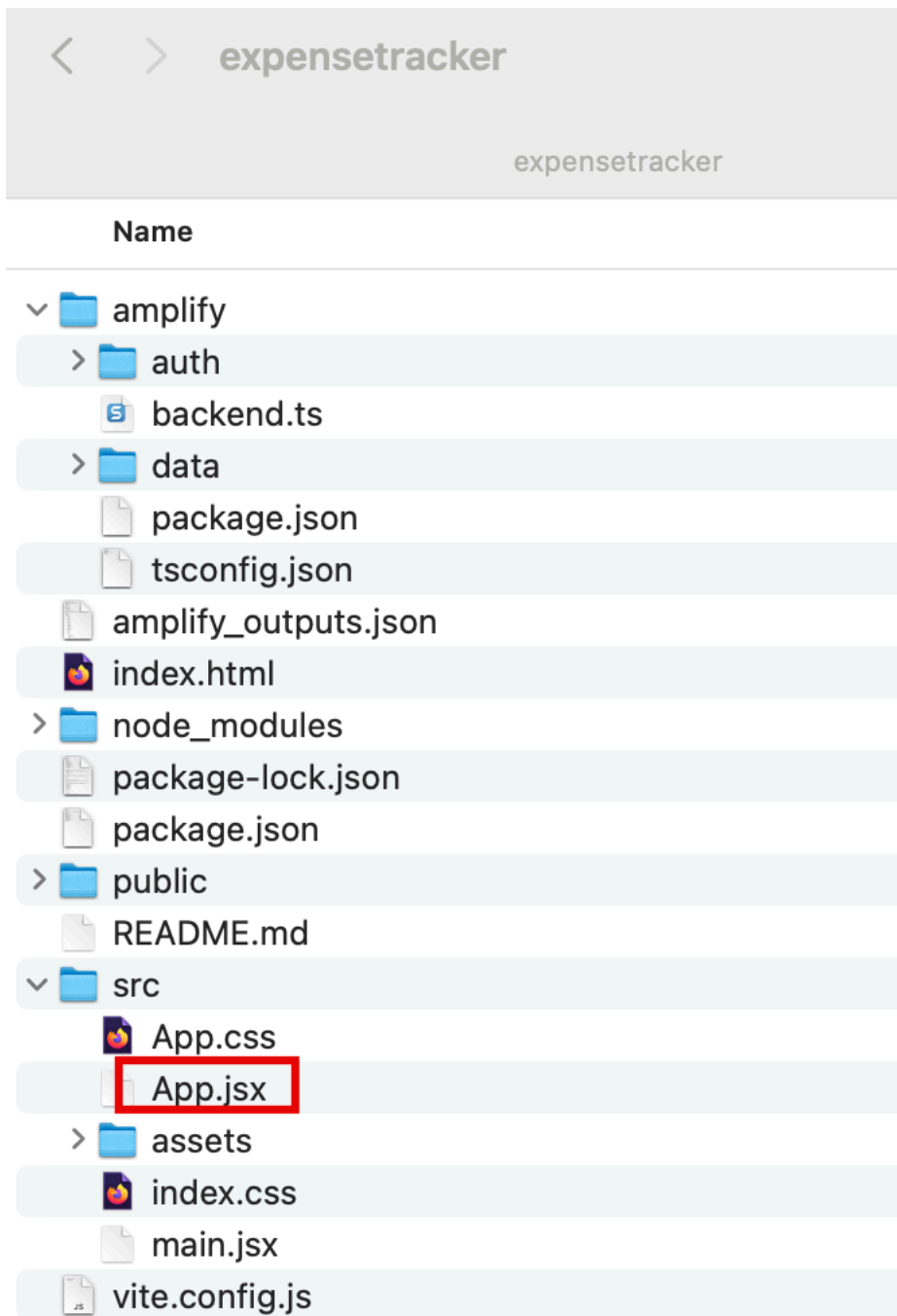
return (
  <Authenticator>
    {{{ signOut }}} => (
      <Flex
        className="App"
        justifyContent="center"
        alignItems="center"
        direction="column"
        width="70%"
        margin="0 auto"
      >
        <Heading level={1}>Expense Tracker</Heading>
        <View as="form" margin="3rem 0" onSubmit={createExpense}>
          <Flex
            direction="column"
            justifyContent="center"
            gap="2rem"
            padding="2rem"
          >
            <TextField
```

```
        name="name"
        placeholder="Expense Name"
        label="Expense Name"
        labelHidden
        variation="quiet"
        required
    />
    <TextField
        name="amount"
        placeholder="Expense Amount"
        label="Expense Amount"
        type="float"
        labelHidden
        variation="quiet"
        required
    />

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

```
        <Heading level="3">{expense.name}</Heading>
    </View>
    <Text fontStyle="italic">${expense.amount}</Text>

    <Button
      variation="destructive"
      onClick={() => deleteExpense(expense)}
    >
      Delete note
    </Button>
  </Flex>
  )})
</Grid>
  <Button onClick={signOut}>Sign Out</Button>
</Flex>
  )}
</Authenticator>
);
}
```



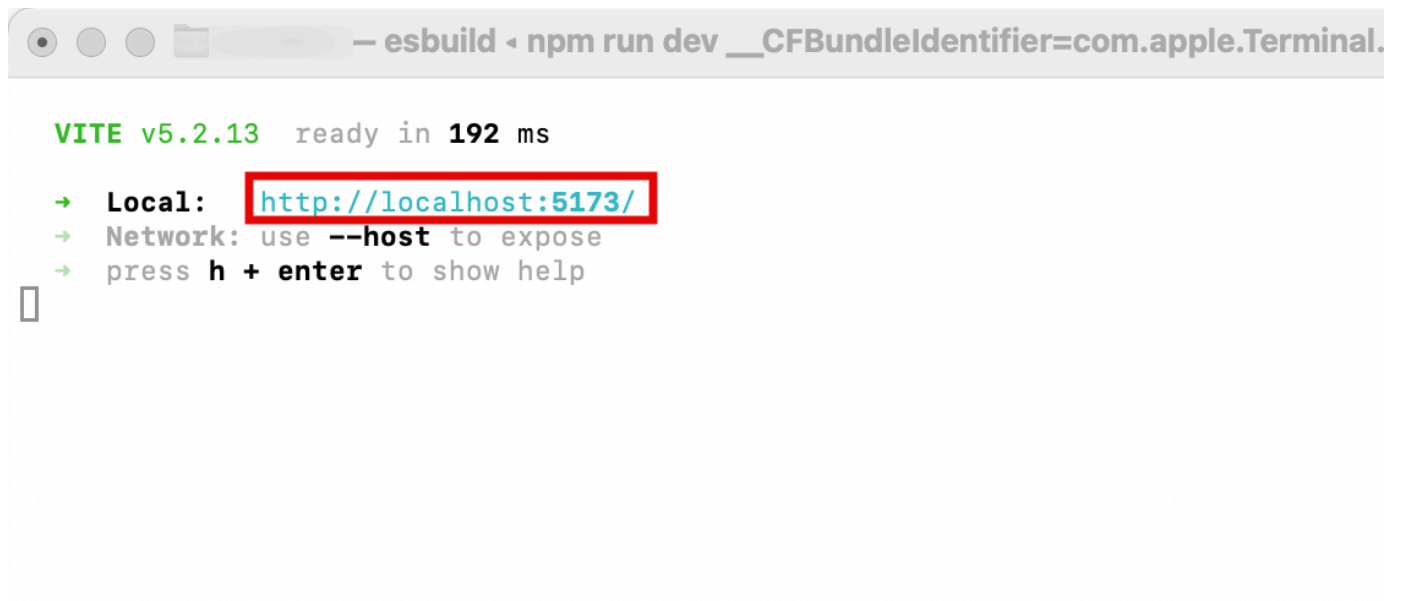
2. Launch the app

Open a new terminal window, **navigate** to your projects root folder (**expensetracker**), and **run** the following command to launch the app:

```
npm run dev
```

3. Open the app

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



```
esbuild ◀ npm run dev __CFBundleIdentifier=com.apple.Terminal.  
  
VITE v5.2.13 ready in 192 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help  
█
```

4. 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**.

Sign In

Create Account

Email

Password

Confirm Password

Create Account

5. Enter verification code

You will get a verification code sent to your email.

Enter the **verification code** to log into the app.

We Emailed You

Your code is on the way. To log in, enter the code we emailed to [redacted]@a***. It may take a minute to arrive.

Confirmation Code

Confirm

Resend Code

6. Create and delete expenses

When signed in, you can start **creating expenses** and **delete** them.

Expense Tracker

Hotel

162.00

Create Expense

Conclusion

You have now connected your App to the Amplify backend and built a frontend allowing the users to create, edit, and delete expenses.

Task 4: Deploy the App

Time to complete

10 minutes

Get help

[Troubleshooting Amplify](#)

Overview

In this task, you will store your application on a GitHub repository, and then set up continuous deployment using the Amplify Console.

What you will accomplish

In this task, you will:

- Connect a Github repository to Amplify
- Set up continuous deployment using Amplify

Implementation

Step 1: Initialize GitHub repository

In this step, you will create a GitHub repository and commit your code to the repository. You will need a GitHub account to complete this step. If you do not have an account, [sign up here](#) .

 **Note**

If you have never used GitHub on your computer, follow [these steps](#) before continuing to allow connection to your account.

1. Sign in to GitHub

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



Sign in to GitHub

Username or email address

Password [Forgot password?](#)

Sign in

2. Start a new repository

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

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

Start a new repository for [REDACTED]

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

✔ expensetracker is available.

Public
Anyone on the internet can see this repository

Private
You choose who can see and commit to this repository

3. Initialize git and push the application

Open a new terminal window, **navigate** to your projects root folder (**expensetracker**), 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
```

```
expensetracker — git-remote-https < git push -u origin main — 84x29
[6c7e67ba14d4 ~ % cd expensetracker ]
[6c7e67ba14d4 expensetracker % git init ]
Initialized empty Git repository in /Users/ /expensetracker/.git/
[6c7e67ba14d4 expensetracker % git add . ]
[6c7e67ba14d4 expensetracker % git commit -m "first commit" ]
[main (root-commit) e31a872] first commit
19 files changed, 4746 insertions(+)
create mode 100644 .eslintrc.cjs
create mode 100644 .gitignore
create mode 100644 README.md
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
create mode 100644 amplify_outputs.json
create mode 100644 index.html
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 public/vite.svg
create mode 100644 src/App.css
create mode 100644 src/App.jsx
create mode 100644 src/assets/react.svg
create mode 100644 src/index.css
create mode 100644 src/main.jsx
create mode 100644 vite.config.js
[6c7e67ba14d4 expensetracker % git branch -M main ]
[6c7e67ba14d4 expensetracker % git remote add origin https://github.com/ /expensetracker ]
[6c7e67ba14d4 expensetracker % git push -u origin main ]
```

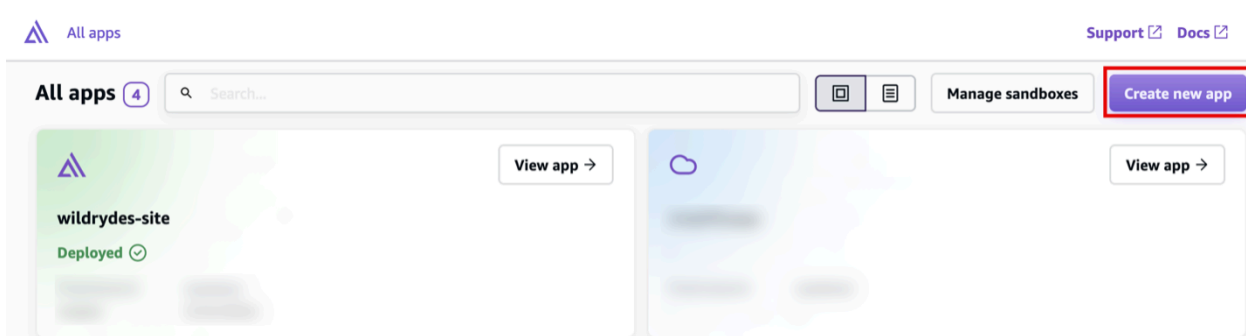
Step 2: Create your app with AWS Amplify

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

1. Create a new app

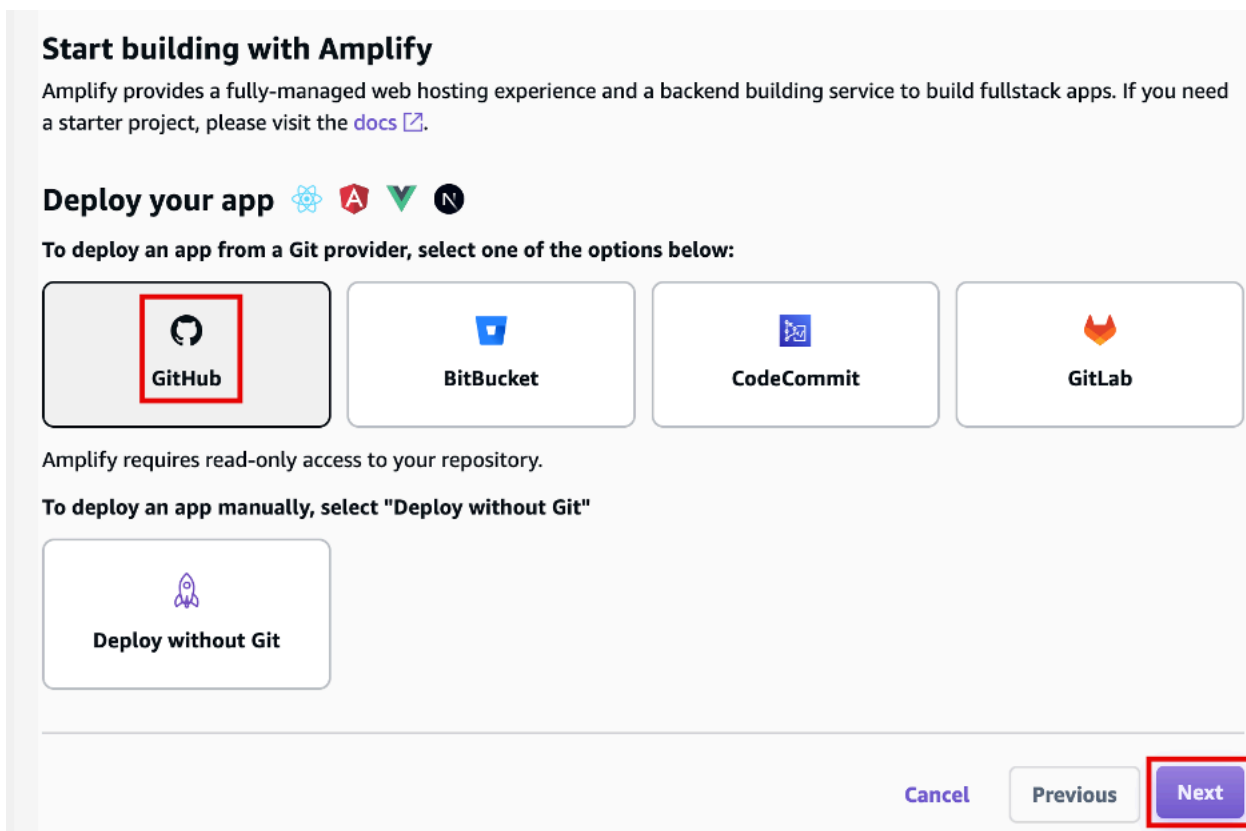
Sign in to the AWS Management Console in a new browser window, and **open** the AWS Amplify console at <https://console.aws.amazon.com/amplify/apps>.

Choose **Create new app**.



2. Choose GitHub for deployment

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



3. Add your repository and main branch

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

The screenshot shows the 'Add repository and branch' step in the AWS Amplify console. On the left, a progress indicator shows four steps: 'Choose source code provider' (completed), 'Add repository and branch' (current), 'App settings' (3), and 'Review' (4). The main area has two search boxes: the first contains '/expensetracker' and the second contains 'main'. A checkbox for 'My app is a monorepo' is unchecked. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted in blue.

4. Confirm default settings

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

The screenshot shows the 'App settings' step in the AWS Amplify console. The progress indicator on the left shows 'App settings' as the current step. The main area is titled 'App settings' and includes: 'App name' (expensetracker), 'Build settings' (Auto-detected frameworks: Amplify Gen 2; Frontend build command: npm run build; Build output directory: dist), 'Service role' (Create and use a new service role; Service role policies selected), and 'Advanced settings'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted in blue.

5. Review settings

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

The screenshot shows the 'Review' step in the AWS Amplify console. On the left, a progress indicator shows four steps: 'Choose source code provider', 'Add repository and branch', 'App settings', and 'Review' (which is currently selected). The main content area is titled 'Review' and contains three sections: 'Repository details', 'App settings', and 'Advanced settings'. Each section has an 'Edit' button. Below these sections is a blue banner with a warning icon and the text: 'First-time account setup required. Amplify needs to run a one-time setup for this account and region before it can deploy resources in the account. This process will begin when you click "Save and deploy" and usually takes between 2 to 5 minutes.' At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Save and deploy' (which is highlighted with a red box).

6. Verify app 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.

The screenshot shows the 'Overview' page for an existing app named 'expensetracker' in the AWS Amplify console. The left sidebar shows navigation options: 'Overview' (selected), 'Hosting', and 'App settings'. The main content area shows the app's details, including the App ID, Production branch, and a table of branches. The 'main' branch is highlighted as 'Deployed'. Below the table, there is a search bar for other branches and a note: 'No other branches added. Add branch'. At the top right, there are two buttons: 'Manage sandboxes' and 'Visit deployed URL' (which is highlighted with a red box).

Task 5: Clean up Resources

Time to complete

< 2 minutes

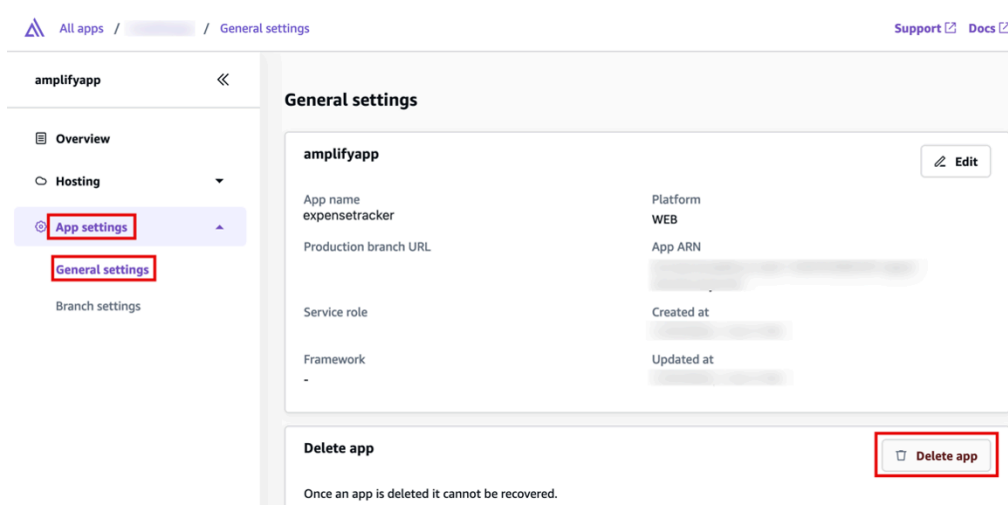
Overview

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

1. In the Amplify console, in the left-hand navigation for the **expensetracker** app, choose **App settings**, and select **General settings**.
2. In the **General settings** section, choose **Delete app**.



Congratulations

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.