

Hands-on tutorials

Build a Serverless Web Application using Generative AI



Build a Serverless Web Application using Generative AI: Hands-on tutorials

Copyright © 2025 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

Build a Serverless Web Application using Generative AI	i
Overview	1
What you will accomplish	1
Prerequisites	2
Application Architecture	2
Tasks	3
Task 1: Host a Static Website	4
Overview	1
What you will accomplish	1
Implementation	4
Conclusion	15
Task 2: Manage Users	16
Overview	1
What you will accomplish	1
Implementation	4
Conclusion	15
Task 3: Build a Serverless Backend	24
Overview	1
What you will accomplish	1
Implementation	4
Conclusion	15
Task 4: Deploy the Backend API	29
Overview	1
What you will accomplish	1
Implementation	4
Conclusion	15
Task 5: Build the Frontend	35
Overview	1
What you will accomplish	1
Implementation	4
Conclusion	15
Task 6: Clean up Resources	51
Overview	1
Congratulations	51

Build a Serverless Web Application using Generative AI

AWS experience	Beginner
Time to complete	35 minutes
Cost to complete	Free tier eligible
Requires	• AWS account with administrator-level access • AWS profile configured • Nodejs and npm • A Github account  Note Accounts created within the past 24 hours might not yet have access to the services required for this tutorial.
Last updated	July 19, 2024

Overview

In this tutorial, you will learn how to use AWS Amplify to build a serverless web application powered by Generative AI using Amazon Bedrock and the [Claude 3 Sonnet](#) foundation model. Users can enter a list of ingredients, and the application will generate delicious recipes based on the input ingredients. The application includes an HTML-based user interface for ingredient submission and a backend web app to request AI-generated recipes.

What you will accomplish

In this tutorial, you will:

- Configure AWS Amplify to host your frontend application with continuous deployment built in

- Configure Amplify Auth and enable Amazon Bedrock foundation model Access
- Build an app backend for handling requests for your web application
- Use Amplify Data to call the serverless backend
- Connect the app to the backend

Prerequisites

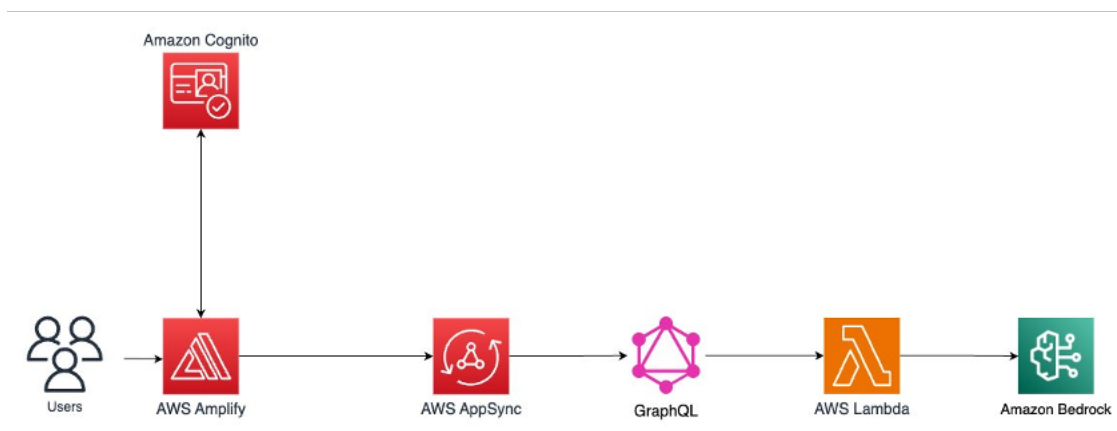
Before starting this tutorial, you will need:

- **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, a GraphQL API built with AWS AppSync, AWS Lambda, and Amazon Bedrock.

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 the following tasks. You must complete each task before moving to the next one.

1. [Task 1: Host a Static Website](#) (5 minutes): Configure AWS Amplify to host your frontend application with continuous deployment built in
2. [Task 2: Manage Users](#) (5 minutes): Configure Amplify Auth and enable Amazon Bedrock foundation model Access
3. [Task 3: Build a Serverless Backend](#) (10 minutes): Build an app backend for handling requests for your web application
4. [Task 4: Deploy the Backend API](#) (5 minutes): Use Amplify Data to call the serverless backend
5. [Task 5: Build the Frontend](#) (5 minutes): Connect the app to the backend
6. [Task 6: Clean up Resources](#) (2 minutes): Clean up the resources used in this tutorial

Task 1: Host a Static Website

Time to complete	5 minutes
Requires	• A GitHub account • GitHub SSH connection
Get help	Troubleshooting Amplify How Amplify works 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 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.

What you will accomplish

In this tutorial, you will:

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

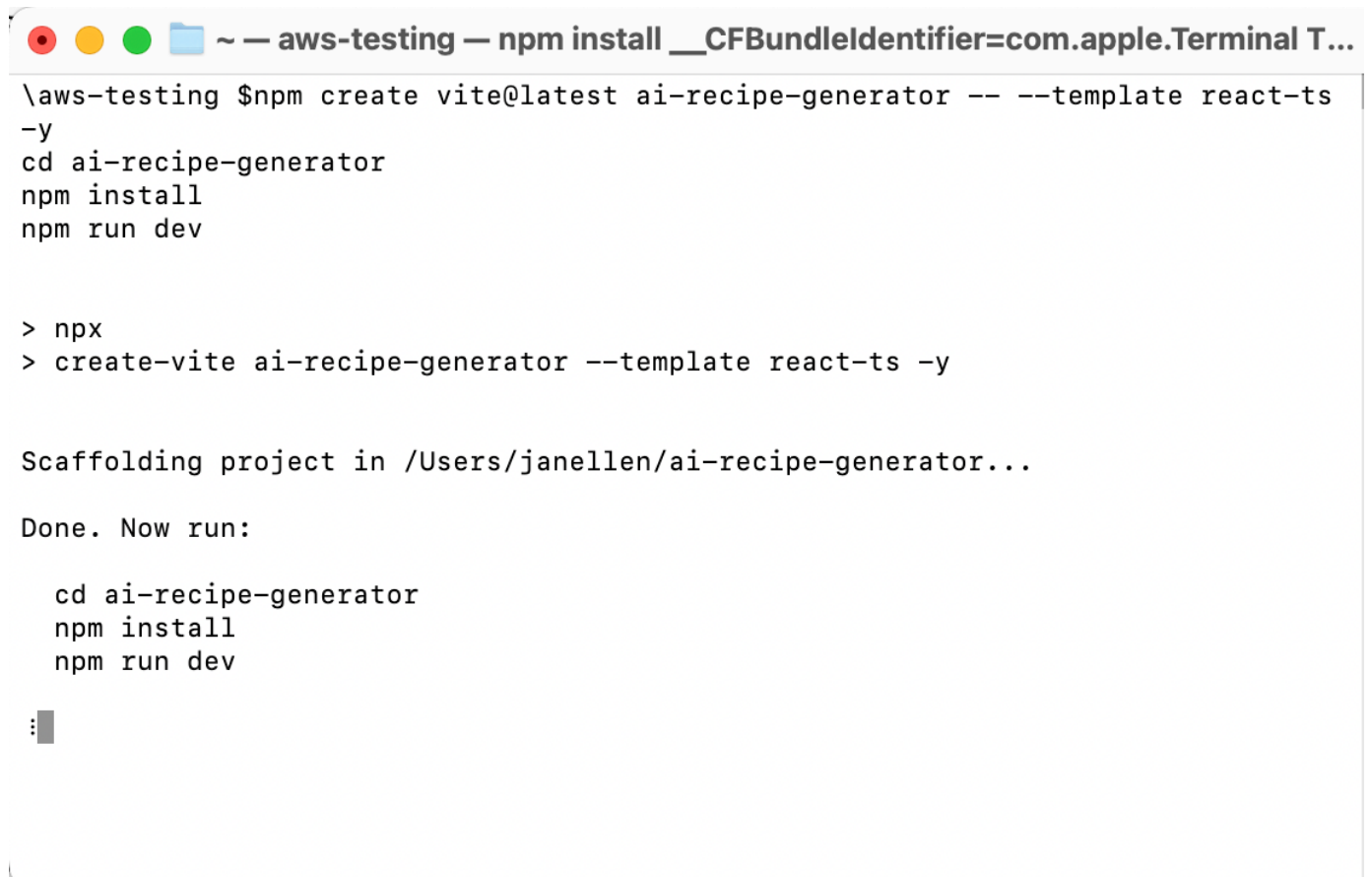
Implementation

Step 1: Create a new React application

1. Create the 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 ai-recipe-generator -- --template react-ts -y
cd ai-recipe-generator
npm install
npm run dev
```

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green) and a folder icon, followed by the text '~ — aws-testing — npm install __CFBundleIdentifier=com.apple.Terminal T...'. The terminal content shows the execution of the command 'npm create vite@latest ai-recipe-generator -- --template react-ts -y', followed by 'cd ai-recipe-generator', 'npm install', and 'npm run dev'. Below these commands, the output shows '> npx' and '> create-vite ai-recipe-generator --template react-ts -y'. This is followed by the message 'Scaffolding project in /Users/janellen/ai-recipe-generator...' and 'Done. Now run:'. Then, the commands 'cd ai-recipe-generator', 'npm install', and 'npm run dev' are repeated. The terminal ends with a prompt character ':' and a cursor bar.

```
~ — aws-testing — npm install __CFBundleIdentifier=com.apple.Terminal T...
\aws-testing $npm create vite@latest ai-recipe-generator -- --template react-ts
-y
cd ai-recipe-generator
npm install
npm run dev

> npx
> create-vite ai-recipe-generator --template react-ts -y

Scaffolding project in /Users/janellen/ai-recipe-generator...

Done. Now run:

  cd ai-recipe-generator
  npm install
  npm run dev

:█
```

2. Open the application

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

```
~ — aws-testing — esbuild ◀ npm run dev __CFBundleIdentifier=com.ap  
  
VITE v5.3.3 ready in 281 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Step 2: Initialize a GitHub repository

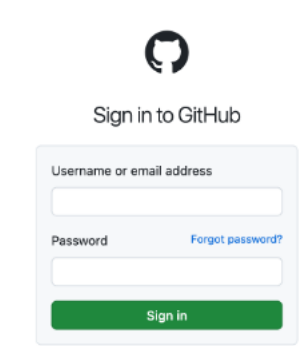
In this step, you will create a GitHub repository and commit your code to the repository. You will need a GitHub account to complete this step, if you do not have an account, [sign up here](#).

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/>.



2. Start a new repository

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

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

Start a new repository for janellefowler

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

Repository name *

ai-recipe-generator

✓ ai-recipe-generator is available.

☒ **Public**

Anyone on the internet can see this repository

☐ **Private**

You choose who can see and commit to this repository

Create a new repository

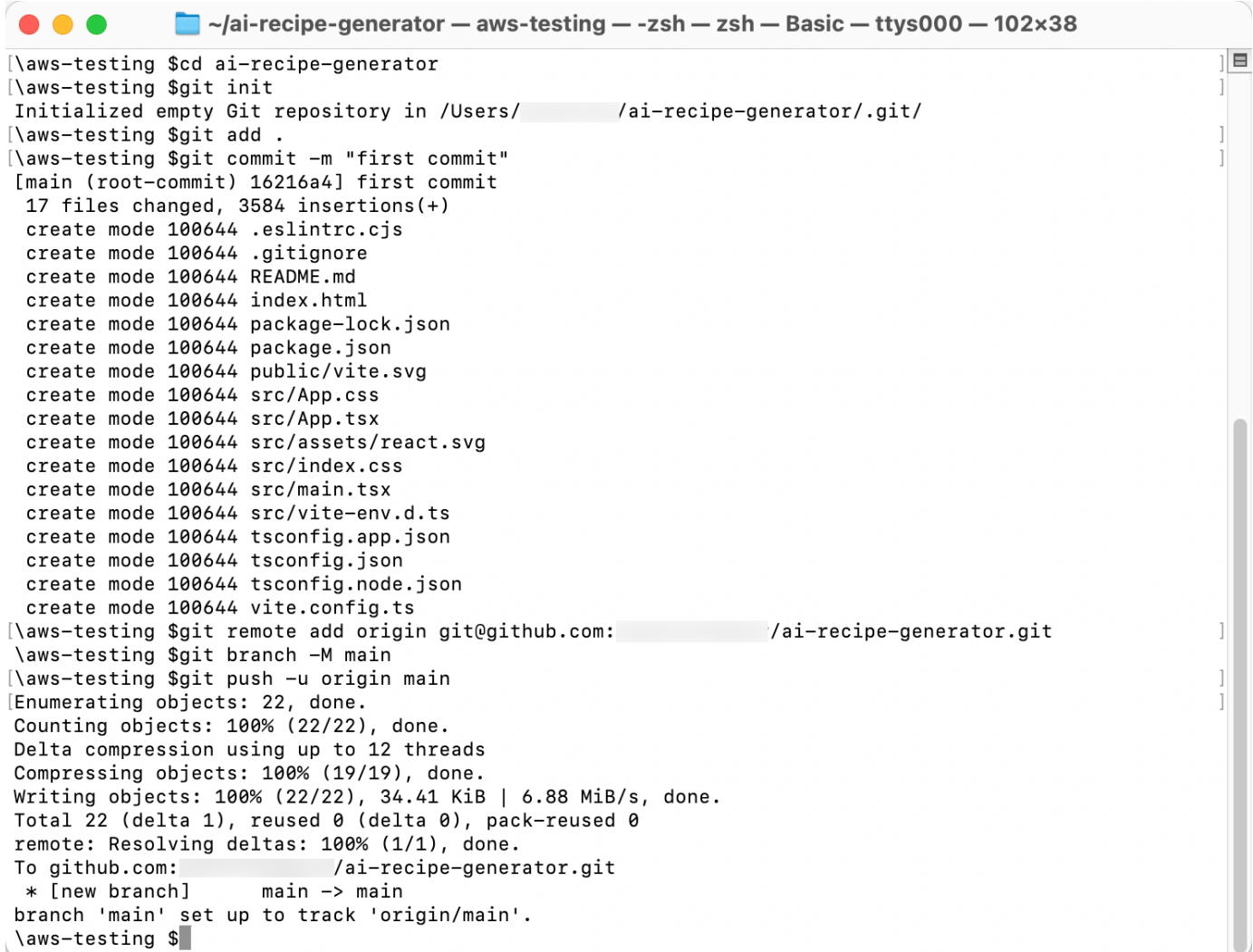
3. Initialize Git

Open a new terminal window, **navigate** to your projects root folder (**ai-recipe-generator**), 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>/ai-recipe-generator.git
git branch -M main
git push -u origin main
```

A terminal window titled '~ /ai-recipe-generator — aws-testing — zsh — zsh — Basic — ttys000 — 102x38'. The terminal shows the execution of git commands to initialize a repository, add files, commit, and push to GitHub. The output shows the creation of 17 files, including .eslintrc.cjs, .gitignore, README.md, index.html, package-lock.json, package.json, public/vite.svg, src/App.css, src/App.tsx, src/assets/react.svg, src/index.css, src/main.tsx, src/vite-env.d.ts, tsconfig.app.json, tsconfig.json, tsconfig.node.json, and vite.config.ts. The push command is successful, and the 'main' branch is set up to track 'origin/main'.

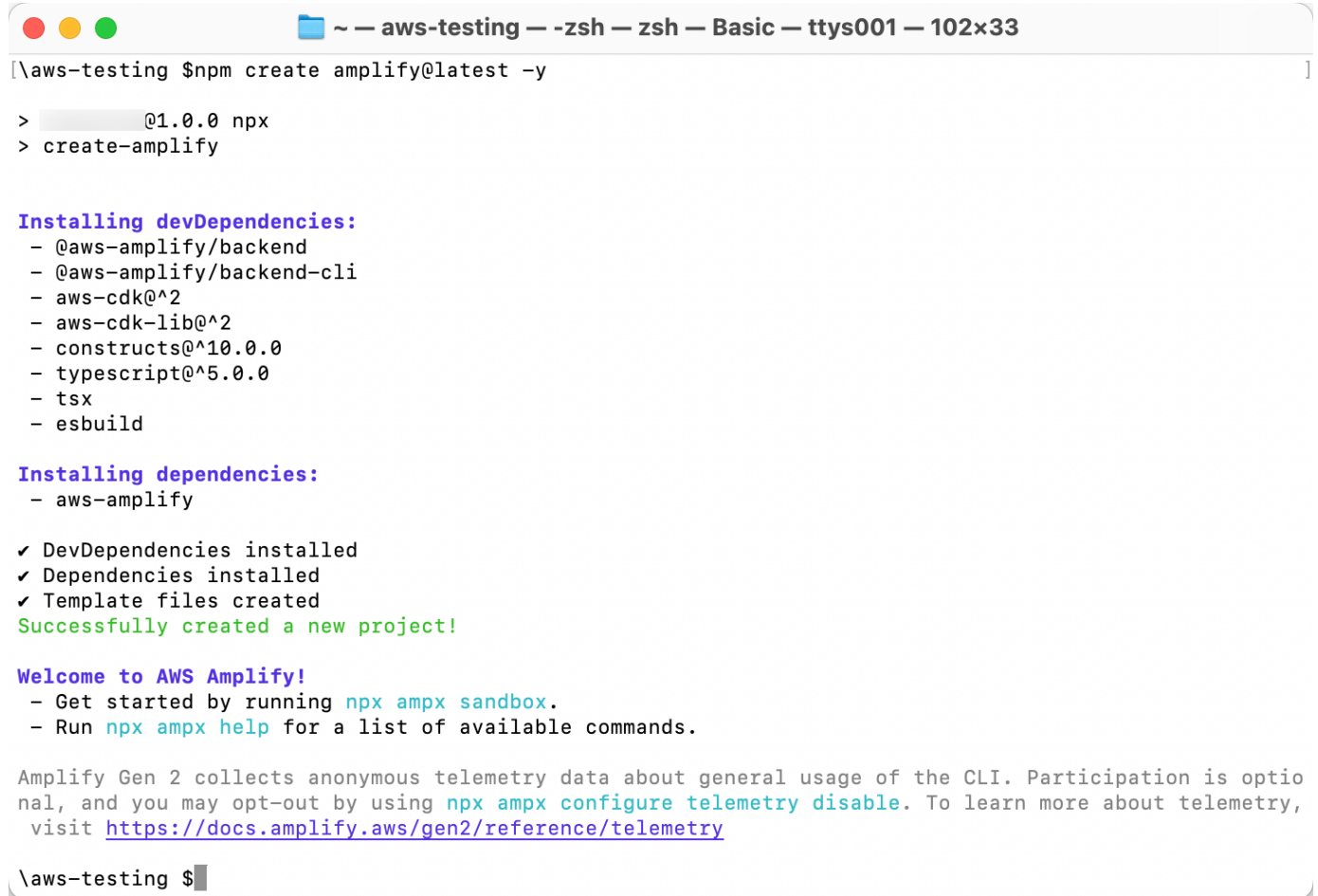
```
\aws-testing $cd ai-recipe-generator
\aws-testing $git init
Initialized empty Git repository in /Users/ /ai-recipe-generator/.git/
\aws-testing $git add .
\aws-testing $git commit -m "first commit"
[main (root-commit) 16216a4] first commit
17 files changed, 3584 insertions(+)
create mode 100644 .eslintrc.cjs
create mode 100644 .gitignore
create mode 100644 README.md
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.tsx
create mode 100644 src/assets/react.svg
create mode 100644 src/index.css
create mode 100644 src/main.tsx
create mode 100644 src/vite-env.d.ts
create mode 100644 tsconfig.app.json
create mode 100644 tsconfig.json
create mode 100644 tsconfig.node.json
create mode 100644 vite.config.ts
\aws-testing $git remote add origin git@github.com: /ai-recipe-generator.git
\aws-testing $git branch -M main
\aws-testing $git push -u origin main
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 12 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (22/22), 34.41 KiB | 6.88 MiB/s, done.
Total 22 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com: /ai-recipe-generator.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
\aws-testing $
```

Step 3: Install the Amplify packages

1. Install Amplify

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

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



```
~ — aws-testing — -zsh — zsh — Basic — ttys001 — 102x33
[aws-testing $ npm create amplify@latest -y
> @1.0.0 npx
> create-amplify

Installing devDependencies:
- @aws-amplify/backend
- @aws-amplify/backend-cli
- aws-cdk@^2
- 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!

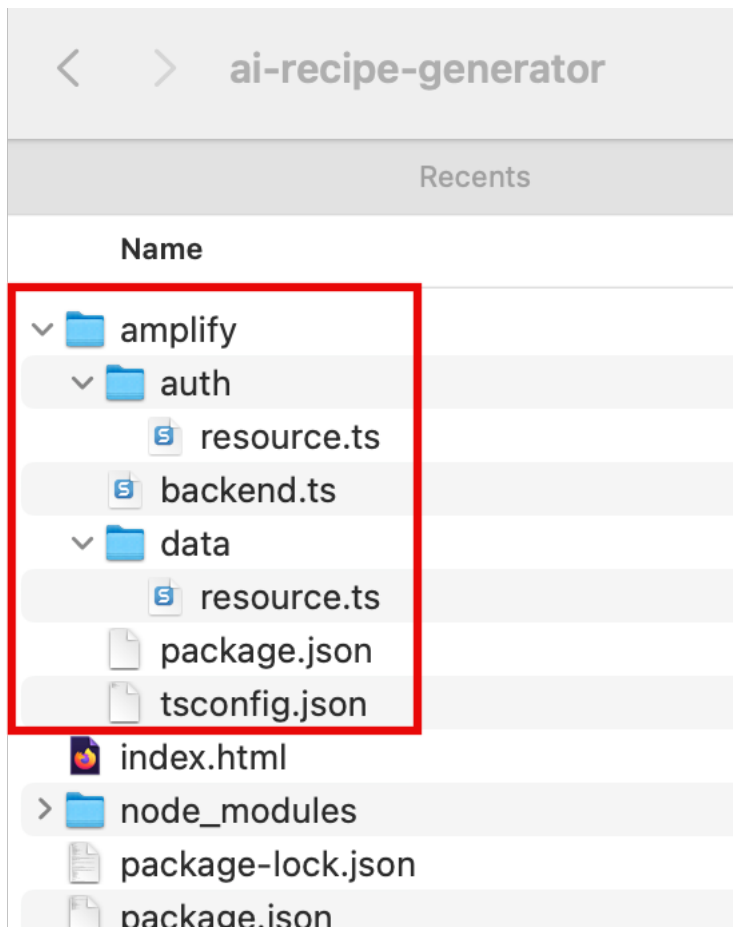
Welcome to AWS Amplify!
- Get started by running npx amp sandbox.
- Run npx amp help for a list of available commands.

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

[aws-testing $
```

2. View directory

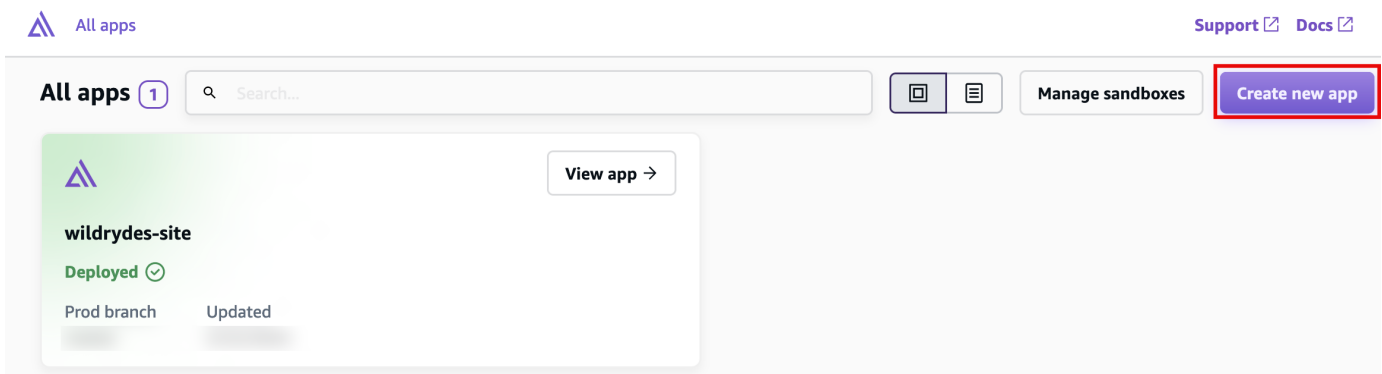
Running the previous command will scaffold a lightweight Amplify project in the app's directory.



Step 4: Deploy your app with AWS Amplify

1. **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. Select GitHub to deploy your app

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

Start building with Amplify

Amplify provides a fully-managed web hosting experience and a backend building service to build fullstack apps. If you need a starter project, please visit the [docs](#).

Deploy your app

To deploy an app from a Git provider, select one of the options below:


GitHub


BitBucket


CodeCommit


GitLab

Amplify requires read-only access to your repository.

To deploy an app manually, select "Deploy without Git"


Deploy without Git

[Cancel](#) [Previous](#) [Next](#)

3. Authenticate with GitHub

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**.

Add repository and branch



If you don't see your repository in the dropdown above, ensure the Amplify GitHub App has permissions to the repository. If your repository still doesn't appear, push a commit and click the refresh button.

Update GitHub permissions

×

☐ My app is a monorepo

Cancel

Previous

Next

4. Select Next

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

App settings

App name

ai-recipe-generator

Build settings

Your build settings have been detected automatically, please verify your "Frontend build command" and "Build output directory".

Auto-detected frameworks

 Amplify Gen 2

Frontend build command

npm run build

Build output directory

dist

☐ Password protect my site

Service role

Amplify requires permissions to deploy backend resources in your account.

☒ Create and use a new service role



Service role policies



☐ Use an existing service role

Advanced settings



Cancel

Previous

Next

5. Review configuration

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

Review

Repository details

Edit

Repository service	Repository
github	/ai-recipe-generator
Branch	Monorepo app root
main	

App settings

Edit

App name	Frontend build command
ai-recipe-generator	npm run build
Framework	Build output directory
Amplify Gen 2	dist

Advanced settings

Edit

Build image	Live package updates
Using default image	
Environment variables	
None	

Cancel

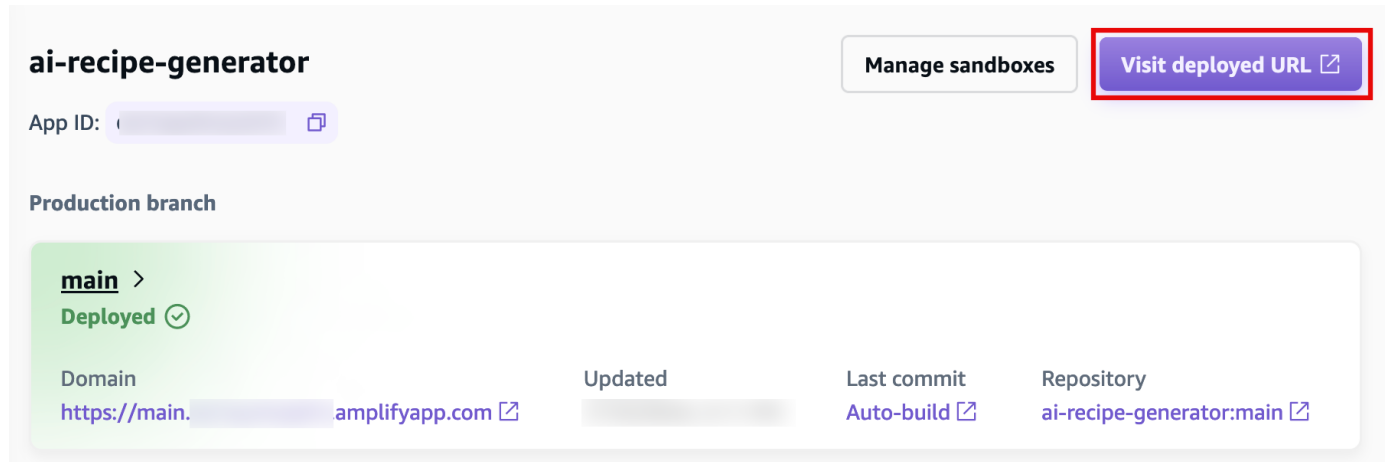
Previous

Save and deploy

6. View your app

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.



Conclusion

You have deployed a React application to AWS 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: Manage Users

Time to complete	15 minutes
Requires	A text editor. Here are a few free ones: • Atom • Notepad++ • Sublime • Vim • Visual Studio Code
Get help	• Troubleshooting Amplify • Troubleshooting Amazon Bedrock • Learn about Auth

Overview

Now that you have a React web app, you will configure an authentication resource for the app using AWS Amplify Auth, powered by Amazon Cognito. Cognito is a powerful user directory service that manages user registration, authentication, account recovery, and more.

You will use the AWS Management Console to enable access to Amazon Bedrock and Claude 3 Sonnet foundation model, which the app will use to generate recipes.

What you will accomplish

In this tutorial, you will:

- Set up Amplify Authentication
- Set up access to Claude 3 Sonnet from Anthropic

Implementation

Step 1: Set up Amplify Auth

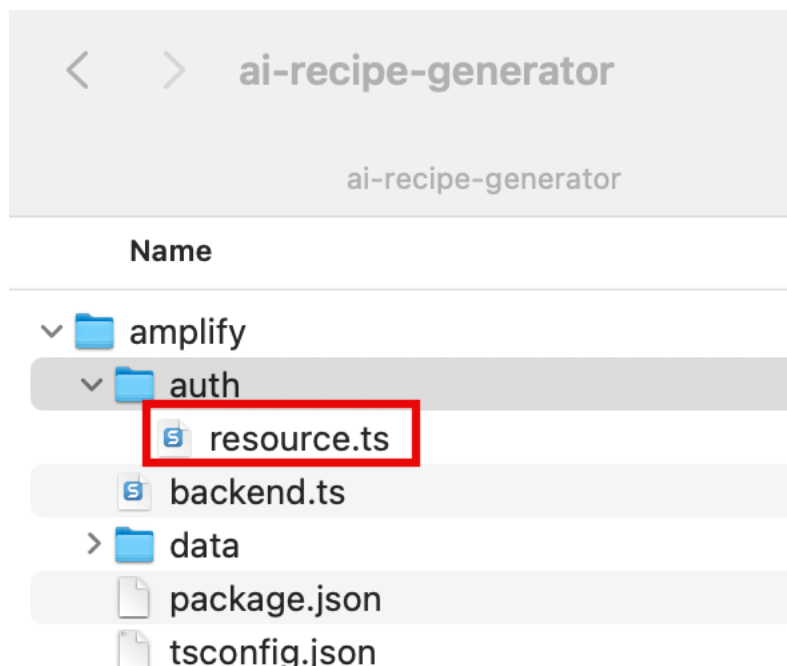
The app uses email as the default login mechanism. When the users sign up, they receive a verification email. In this step, you will customize the verification email that is sent to users.

1. Modify the resource file

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

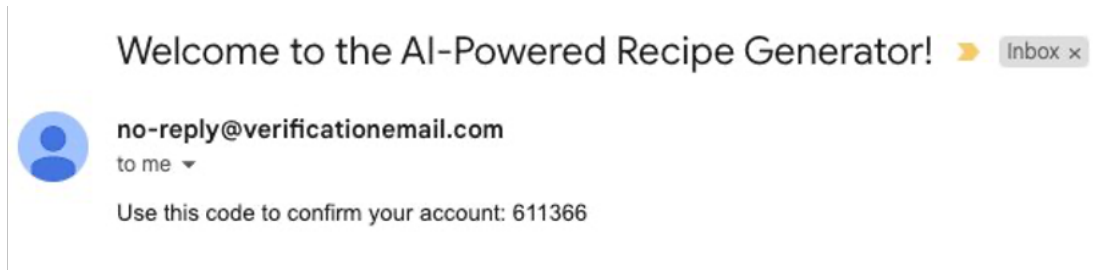
```
import { defineAuth } from "@aws-amplify/backend";

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



2. View the customized email

The following image is an example of the customized verification email.



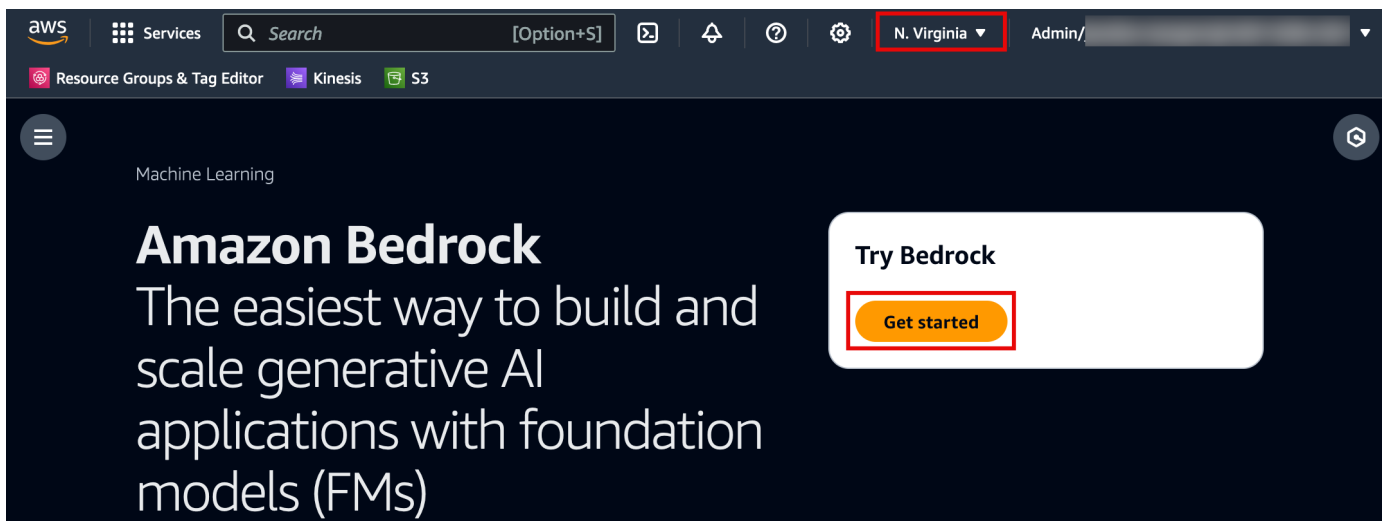
Step 2: Set up Amazon Bedrock Model Access

Amazon Bedrock enables users to request access to a variety of Generative AI models. In this tutorial, you will need access to Claude 3 Sonnet from Anthropic.

1. Open the Bedrock console

Sign in to the AWS Management console in a new browser window, and **open** the AWS Amazon Bedrock console at <https://console.aws.amazon.com/bedrock/>.

Verify that you are in the **N. Virginia us-east-1** region, and choose **Get started**.



2. Select the Claude model

In the **Foundation models** section, choose the **Claude model**.

[Amazon Bedrock](#) > Overview

Overview [Info](#)

[Explore & Learn](#)[Build & Test](#)

Foundation models

Amazon Bedrock supports foundation models from industry-leading providers. Choose the model that is best suited to achieving your unique goals.



Jamba-Instruct
By AI21
Labs



Titan
By Amazon



Claude
By
Anthropic



Command
By Cohere



Llama 3
By Meta



Mistral
By Mistral
AI



**Stable
Diffusion**
By Stability
AI

3. Request access to Claude 3 Sonnet

Scroll down to the **Claude models** section, and choose the **Claude 3 Sonnet** tab, and select **Request model access**.

Note

If you already have access to some models, then the button will display **Manage model access**.

Claude models

[Request model access](#)

Anthropic's Claude family of models – Haiku, Sonnet, and Opus – allow customers to choose the exact combination of intelligence, speed, and cost that suits their business needs. Claude 3 Opus, the company's most capable model, has set a market standard on benchmarks. All of the latest Claude models have vision capabilities that enable them to process and analyze image data, meeting a growing demand for multimodal AI systems that can handle diverse data formats. While the family offers impressive performance across the board, Claude 3 Haiku is one of the most affordable and fastest options on the market for its intelligence category.

< | Claude 3 Opus | **Claude 3 Sonnet** | Claude 3 Haiku | Claude 2.1 | Cla >

 This account does not currently have access to this model. Request access in [Model access](#).

4. Request model access

In the Base models section, for **Claude 3 Sonnet**, choose **Available to request**, and select **Request model access**.

Base models (29)



Expand all

Collapse all

Not seeing a model you're interested in? Check out all supported models by region [here](#).

Group by provider

Models	Access status	Modality	EULA
▶ AI21 Labs (2)	0/2 access granted		
▶ Amazon (6)	0/6 access granted		
▼ Anthropic (5)	0/5 access granted		
Claude 3 Opus	 Unavailable	Text & Vision	EULA
Claude 3 Sonnet	 Available to request		
Claude 3 Haiku	 Available to request		
Claude	 Available to request		

Available to request

×

You can request access to this model. Billing will start after you are granted access and start using the model in Bedrock.

Request model access

5. Choose Next

On the **Edit model access** page, choose **Next**.

Edit model access

Base models (1/29)

Expand all Collapse all

Not seeing a model you're interested in? Check out all supported models by region [here](#).

Group by provider

	Models	Access status	Modality	EULA
☐	▶ AI21 Labs (2)	0/2 access granted		
☐	▶ Amazon (6)	0/6 access granted		
☐	▼ Anthropic (5)	0/5 access granted		
☐	Claude 3 Opus	Unavailable	Text & Vision	EULA
☒	Claude 3 Sonnet	Available to request	Text & Vision	EULA
☐	Claude 3 Haiku	Available to request	Text & Vision	EULA
☐	Claude	Available to request	Text	EULA
☐	Claude Instant	Available to request	Text	EULA
☐	▶ Cohere (6)	0/6 access granted		
☐	▶ Meta (6)	0/6 access granted		
☐	▶ Mistral AI (3)	0/3 access granted		
☐	▶ Stability AI (1)	0/1 access granted		

Cancel
Next

6. Submit request

On the **Review and Submit** page, choose **Submit**.

Edit model access

Base models (1/29)

Expand all

Collapse all

Not seeing a model you're interested in? Check out all supported models by region [here](#).

Group by provider ▼

	Models	Access status	Modality	EULA
☐	▶ AI21 Labs (2)	0/2 access granted		
☐	▶ Amazon (6)	0/6 access granted		
☐	▼ Anthropic (5)	0/5 access granted		
☐	Claude 3 Opus	Unavailable	Text & Vision	EULA
☒	Claude 3 Sonnet	Available to request	Text & Vision	EULA
☐	Claude 3 Haiku	Available to request	Text & Vision	EULA
☐	Claude	Available to request	Text	EULA
☐	Claude Instant	Available to request	Text	EULA
☐	▶ Cohere (6)	0/6 access granted		
☐	▶ Meta (6)	0/6 access granted		
☐	▶ Mistral AI (3)	0/3 access granted		
☐	▶ Stability AI (1)	0/1 access granted		

Cancel

Next

Conclusion

You have configured Amplify for authentication and customized the verification email, and enabled access to Amazon Bedrock and Claude 3 Sonnet.

Task 3: Build a Serverless Backend

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 Lambda Functions

Overview

In this task, you will configure a serverless function using AWS Amplify and AWS Lambda. This function takes an input parameter i.e. ingredients to generate a prompt. It then sends this prompt to Amazon Bedrock via an HTTP POST request to the Claude 3 Sonnet model. The body of the request includes the prompt string within a messages array.

What you will accomplish

In this tutorial, you will:

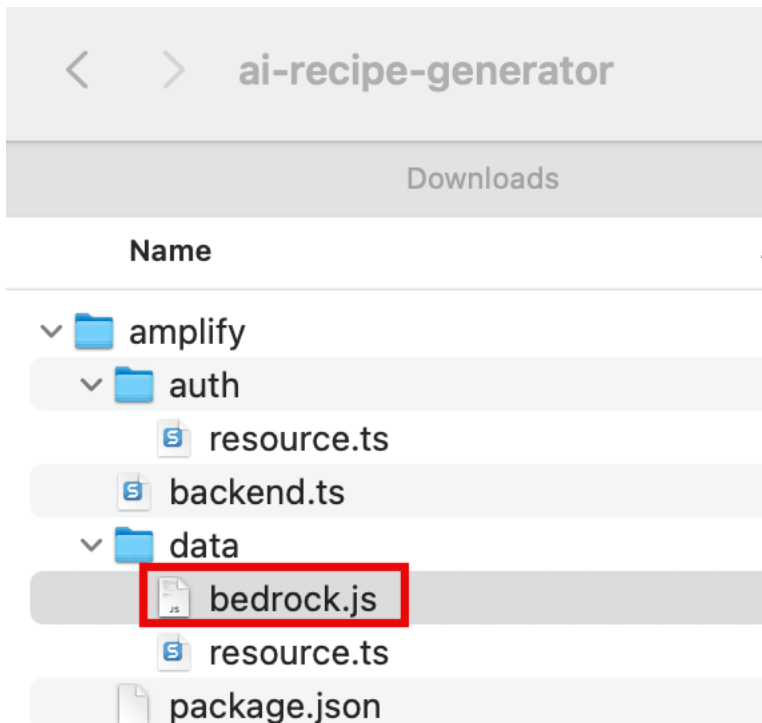
- Add Amazon Bedrock as a data source
- Configure custom business logic handler code

Implementation

Step 1: Create a Lambda function for handling requests

1. Create a Lambda function

On your local machine, navigate to the **ai-recipe-generator/amplify/data** folder, and **create** a file named **bedrock.js**.



2. Add the function code

Then, **update** the file with the following code:

```
export function request(ctx) {
  const { ingredients = [] } = ctx.args;

  // Construct the prompt with the provided ingredients
  const prompt = `Suggest a recipe idea using these ingredients:
${ingredients.join(", ")}`;

  // Return the request configuration
  return {
    resourcePath: `/model/anthropic.claude-3-sonnet-20240229-v1:0/invoke`,
    method: "POST",
    params: {
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        anthropic_version: "bedrock-2023-05-31",
```

```

        max_tokens: 1000,
        messages: [
          {
            role: "user",
            content: [
              {
                type: "text",
                text: `\n\nHuman: ${prompt}\n\nAssistant:`,
              },
            ],
          },
        ],
      },
    ],
  },
);
};
}

export function response(ctx) {
  // Parse the response body
  const parsedBody = JSON.parse(ctx.result.body);
  // Extract the text content from the response
  const res = {
    body: parsedBody.content[0].text,
  };
  // Return the response
  return res;
}

```

This code defines a request function that constructs the HTTP request to invoke the Claude 3 Sonnet foundation model in Amazon Bedrock. The response function parses the response and returns the generated recipe.

Step 2: Add Amazon Bedrock as a data source

- Update the backend file

Update the **amplify/backend.ts** file with the following code. Then, save the file.

```

import { defineBackend } from "@aws-amplify/backend";
import { data } from "../data/resource";
import { PolicyStatement } from "aws-cdk-lib/aws-iam";
import { auth } from "../auth/resource";

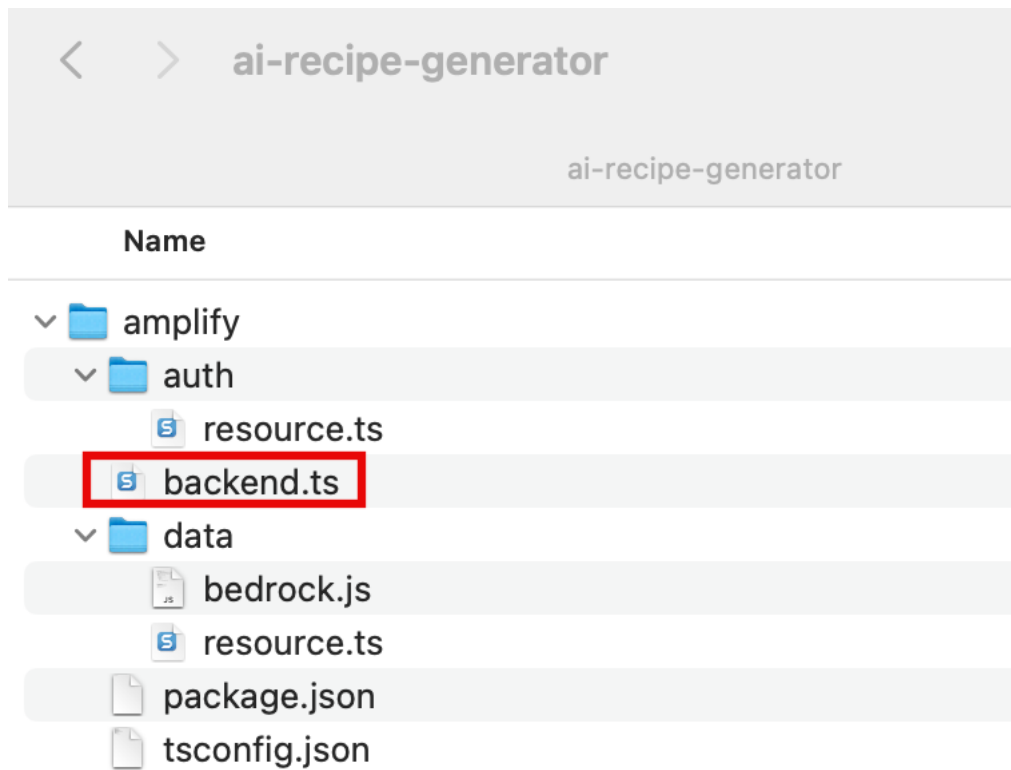
```

```
const backend = defineBackend({
  auth,
  data,
});

const bedrockDataSource = backend.data.resources.graphqlApi.addHttpDataSource(
  "bedrockDS",
  "https://bedrock-runtime.us-east-1.amazonaws.com",
  {
    authorizationConfig: {
      signingRegion: "us-east-1",
      signingServiceName: "bedrock",
    },
  },
);

bedrockDataSource.grantPrincipal.addToPrincipalPolicy(
  new PolicyStatement({
    resources: [
      "arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-3-sonnet-20240229-v1:0",
    ],
    actions: ["bedrock:InvokeModel"],
  })
);
```

- The code adds an HTTP data source for Amazon Bedrock to your API and grant it permissions to invoke the Claude model.



Conclusion

You have defined a Lambda function using Amplify, and added Amazon Bedrock as an HTTP data source.

Task 4: Deploy the Backend API

Overview

In this task, you will configure a custom query that will reference the data source and the resolver you defined the previous model to produce a recipe based on a list of ingredients. This query will use a custom type to structure the response from Amazon Bedrock.

What you will accomplish

In this tutorial, you will:

- Define a GraphQL Query that takes an array of strings
- Define a custom type to be used to structure the response from the query

Implementation

Time to complete	5 minutes
Get help	Troubleshooting Amplify Learn about Data

Step 1: Set up Amplify Data

1. Update the `resource.ts` file

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

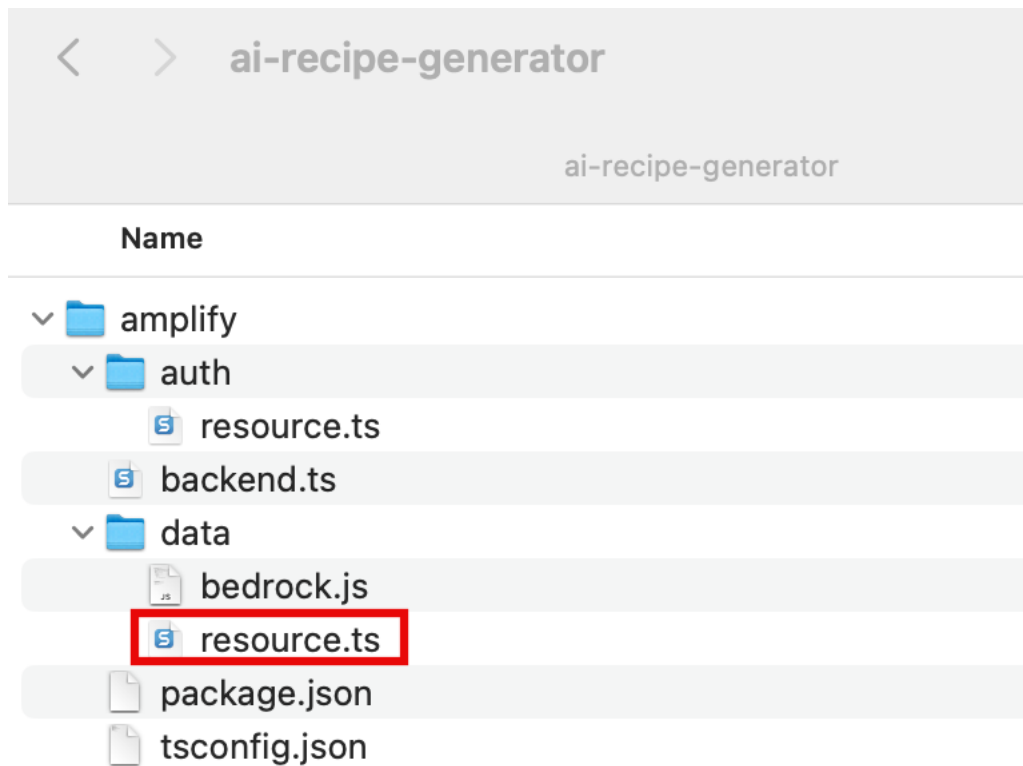
- The following code defines the `askBedrock` query that takes an array of strings called `ingredients` and returns a `BedrockResponse`. The `.handler(a.handler.custom({ entry: ". / bedrock.js", dataSource: "bedrockDS" })))` line sets up a custom handler for this query, defined in `bedrock.js`, using `bedrockDS` as its data source.

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

const schema = a.schema({
  BedrockResponse: a.customType({
    body: a.string(),
    error: a.string(),
  }),
  askBedrock: a
    .query()
    .arguments({ ingredients: a.string().array() })
    .returns(a.ref("BedrockResponse"))
    .authorization((allow) => [allow.authenticated()])
    .handler(
      a.handler.custom({
        entry: "./bedrock.js",
        dataSource: "bedrockDS"
      })
    ),
});

export type Schema = ClientSchema<typeof schema>;

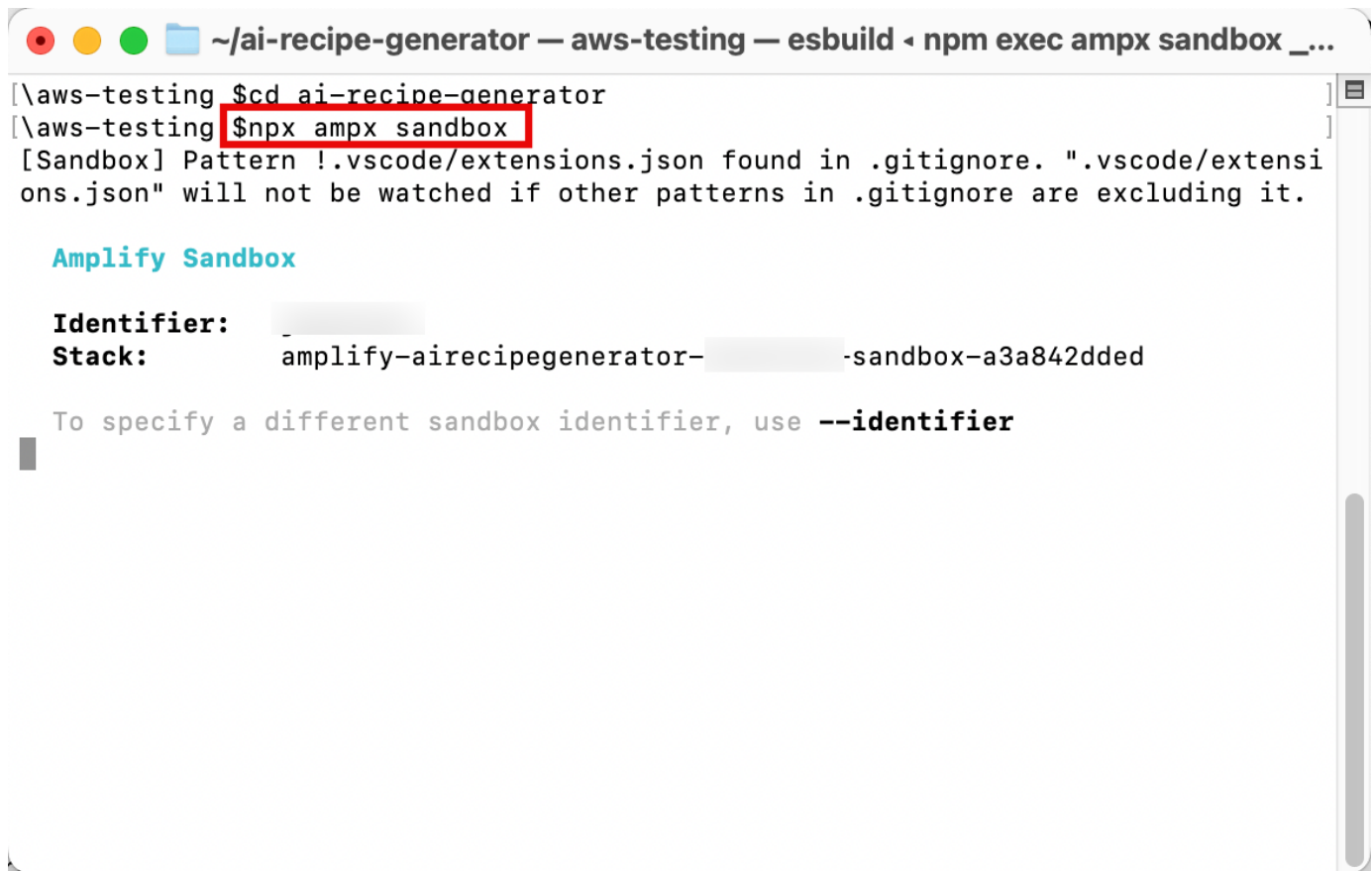
export const data = defineData({
  schema,
  authorizationModes: {
    defaultAuthorizationMode: "apiKey",
    apiKeyAuthorizationMode: {
      expiresInDays: 30,
    },
  },
});
```



2. Deploy resources

Open a new terminal window, **navigate** to your apps project folder (ai-recipe-generator), and **run** the following command to deploy cloud resources into an isolated development space so you can iterate fast.

```
npx ampx sandbox
```



```
~/ai-recipe-generator — aws-testing — esbuild ◀ npm exec ampx sandbox _...  
[\aws-testing $cd ai-recipe-generator  
[\aws-testing $npm exec ampx sandbox  
[Sandbox] Pattern !.vscode/extensions.json found in .gitignore. ".vscode/extension  
ons.json" will not be watched if other patterns in .gitignore are excluding it.  
  
Amplify Sandbox  
  
Identifier:   
Stack:      amplify-ai-recipe-generator- -sandbox-a3a842dded  
  
To specify a different sandbox identifier, use --identifier
```

3. View confirmation message

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

```
~/ai-recipe-generator — aws-testing — node ◀ npm exec ampx sandbox __CFBundleIdentifier...

amplify-airecipegenerator-sandbox-a3a842dded.oauthRedirectSignIn = https://example.com
amplify-airecipegenerator-sandbox-a3a842dded.oauthRedirectSignOut =
amplify-airecipegenerator-sandbox-a3a842dded.oauthResponseType = code
amplify-airecipegenerator-sandbox-a3a842dded.oauthScope = ["profile", "phone", "email", "openid", "aws.cognito.signinUserAttribute", "aws.cognito.signinUserPassword"]
amplify-airecipegenerator-sandbox-a3a842dded.passwordPolicyMinLength = 8
amplify-airecipegenerator-sandbox-a3a842dded.passwordPolicyRequirements = ["REQUIRES_NUMBERS", "REQUIRES_LOWERCASE", "REQUIRES_UPPERCASE", "REQUIRES_SYMBOLS"]
amplify-airecipegenerator-sandbox-a3a842dded.region = us-east-1
amplify-airecipegenerator-sandbox-a3a842dded.signupAttributes = ["email"]
amplify-airecipegenerator-sandbox-a3a842dded.socialProviders =
amplify-airecipegenerator-sandbox-a3a842dded.userPoolId = us-east-1_cUkDxFZWE
amplify-airecipegenerator-sandbox-a3a842dded.usernameAttributes = ["email"]
amplify-airecipegenerator-sandbox-a3a842dded.verificationMechanisms = ["email"]
amplify-airecipegenerator-sandbox-a3a842dded.webClientId = 4r3c5vvtcring64u1urv5cooo9

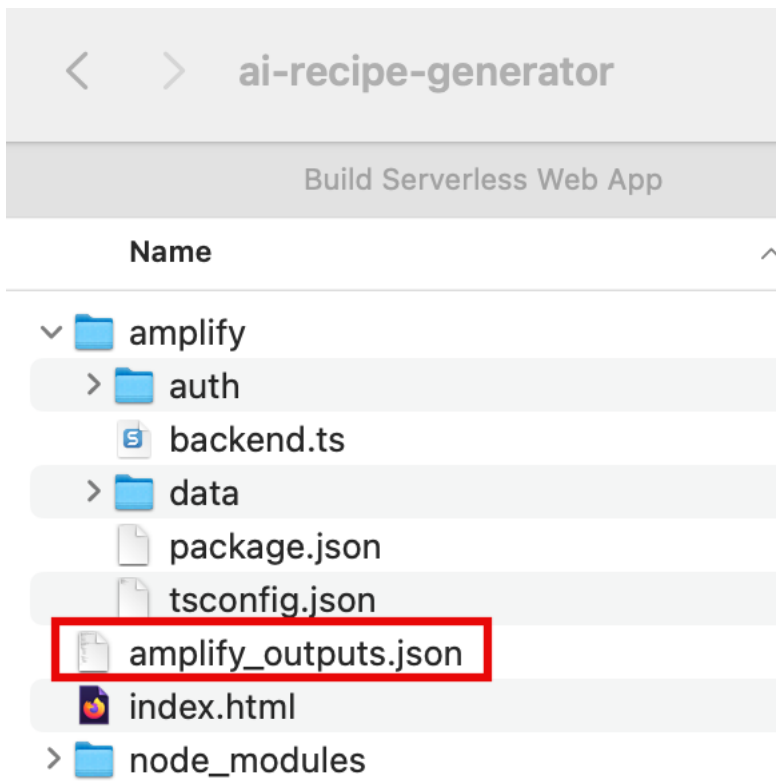
Stack ARN:
arn:aws:cloudformation:us-east-1:640702963591:stack/amplify-airecipegenerator-sandbox-a3a842dded/4a39ece0-...

🌟 Total time: 118.98s

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

4. Verify outputs file creation

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



Conclusion

You have configured a GraphQL API to define a custom query to connect to Amazon Bedrock and generate recipes based on a list of ingredients.

Task 5: Build the Frontend

Time to complete

5 minutes

Get help

[Troubleshooting Amplify](#)

[Troubleshooting common issues using Amplify UI](#)

Overview

In this task, you will update the website you created in module 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 to use the customer query for generating recipe based on a list of ingredients.

What you will accomplish

In this tutorial, you will:

- Install the Amplify client libraries
- Configure your React app to add the authentication flow and invoke the GraphQL API

Implementation

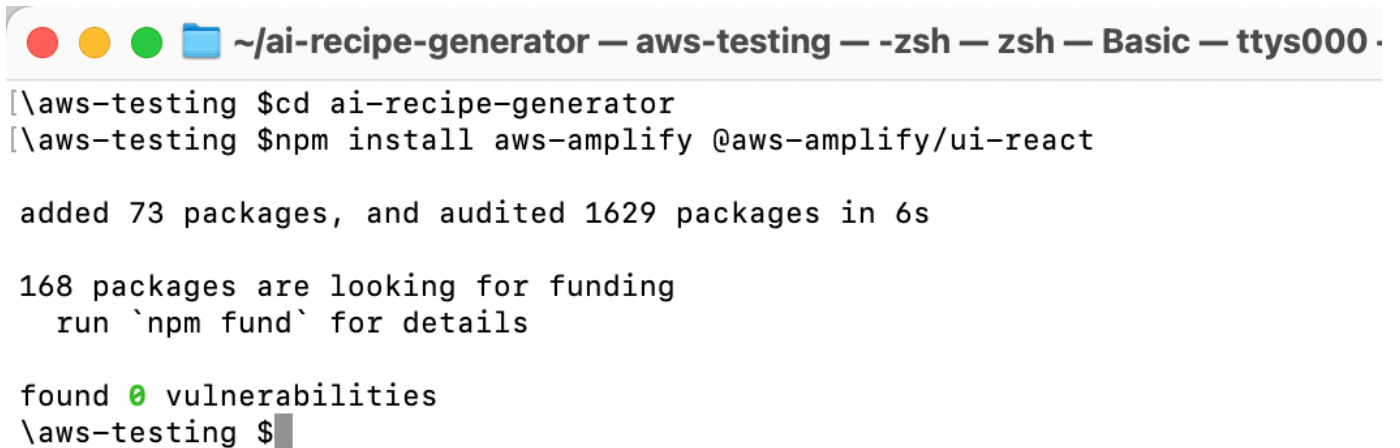
Step 1: Install the Amplify libraries

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

Open a new terminal window, **navigate** to your projects root folder (**ai-recipe-generator**), and **run** the following command to install the libraries.

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



```
~/ai-recipe-generator — aws-testing — -zsh — zsh — Basic — ttys000 .
[\aws-testing $cd ai-recipe-generator
[\aws-testing $npm install aws-amplify @aws-amplify/ui-react

added 73 packages, and audited 1629 packages in 6s

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

found 0 vulnerabilities
[\aws-testing $
```

Step 2: Style the App UI

1. Modify the Index CSS

On your local machine, navigate to the **ai-recipe-generator/src/index.css** file, and **update** it with the following code to center the App 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;

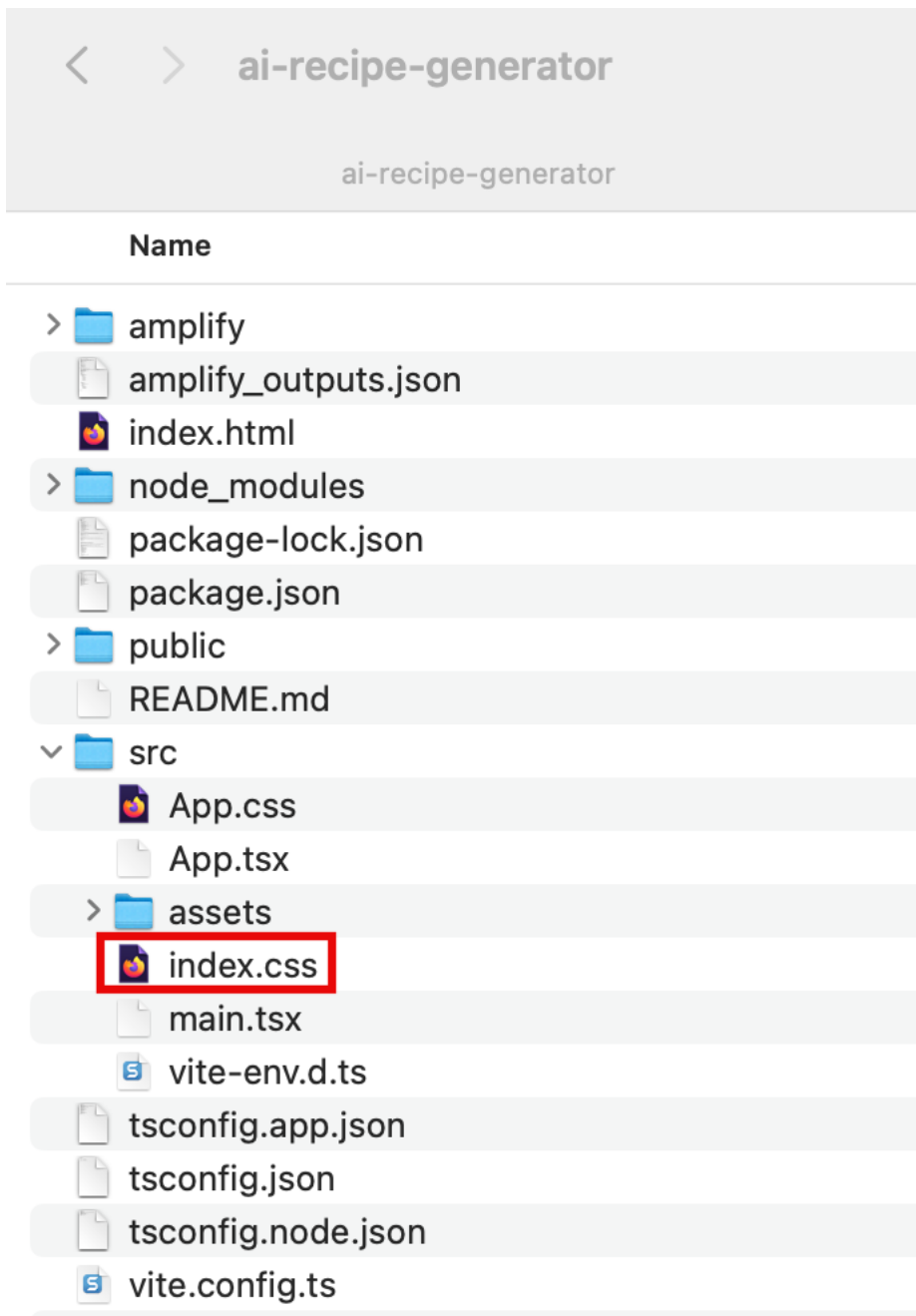
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
```

```
}

.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;
}
```



2. Modify the App CSS

Update the `src/App.css` file with the following code to style the ingredients form. Then, **save** the file.

```
.app-container {  
  
  margin: 0 auto;  
  padding: 20px;  
}
```

```
    text-align: center;
  }

  .header-container {
    padding-bottom: 2.5rem;
    margin: auto;
    text-align: center;

    align-items: center;
    max-width: 48rem;

  }

  .main-header {
    font-size: 2.25rem;
    font-weight: bold;
    color: #1a202c;
  }

  .main-header .highlight {
    color: #2563eb;
  }

  @media (min-width: 640px) {
    .main-header {
      font-size: 3.75rem;
    }
  }

  .description {

    font-weight: 500;
    font-size: 1.125rem;
    max-width: 65ch;
    color: #1a202c;
  }

  .form-container {
    margin-bottom: 20px;
  }

  .search-container {
    display: flex;
```

```
    flex-direction: column;
    gap: 10px;
    align-items: center;
}

.wide-input {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

.search-button {
    width: 100%; /* Make the button full width */
    max-width: 300px; /* Set a maximum width for the button */
    padding: 10px;
    font-size: 16px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

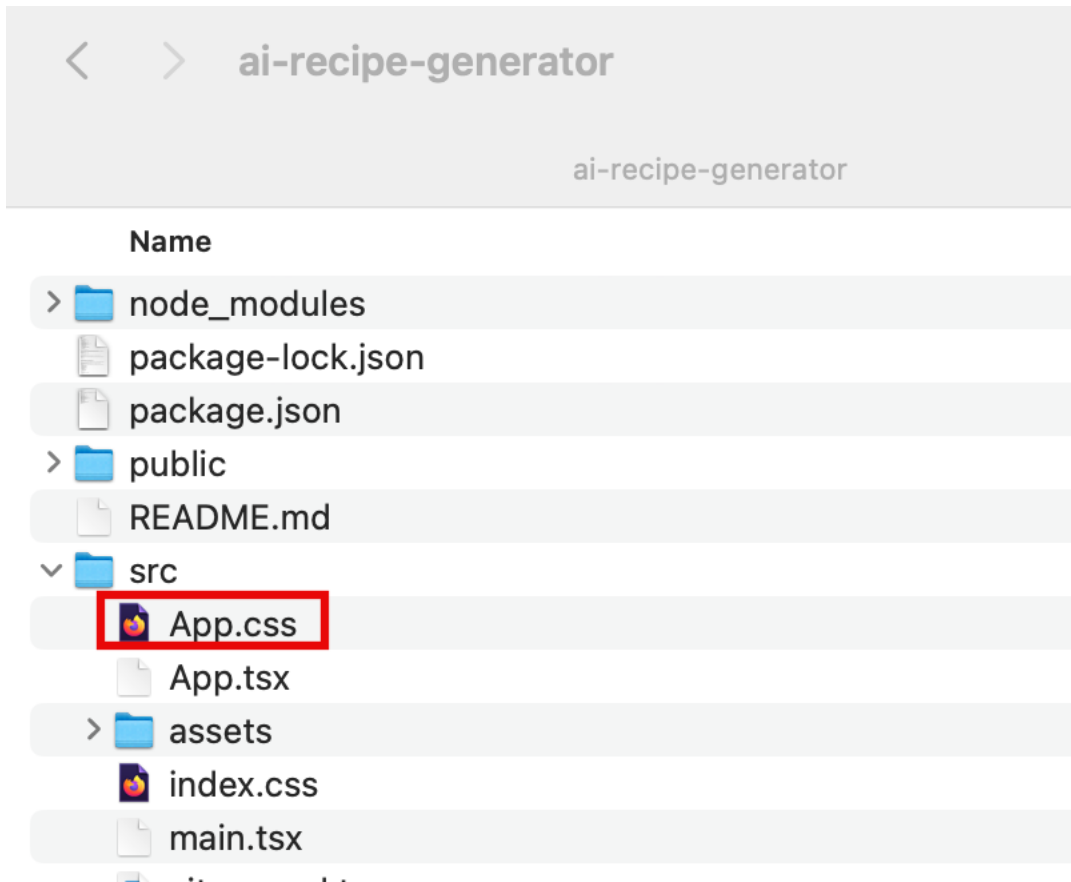
.search-button:hover {
    background-color: #0056b3;
}

.result-container {
    margin-top: 20px;
    transition: height 0.3s ease-out;
    overflow: hidden;
}

.loader-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 10px;
}

.result {
    background-color: #f8f9fa;
```

```
border: 1px solid #e9ecef;  
border-radius: 4px;  
padding: 15px;  
white-space: pre-wrap;  
word-wrap: break-word;  
color: black;  
font-weight: bold;  
text-align: left; /* Align text to the left */  
}
```



Step 3: Implement the UI

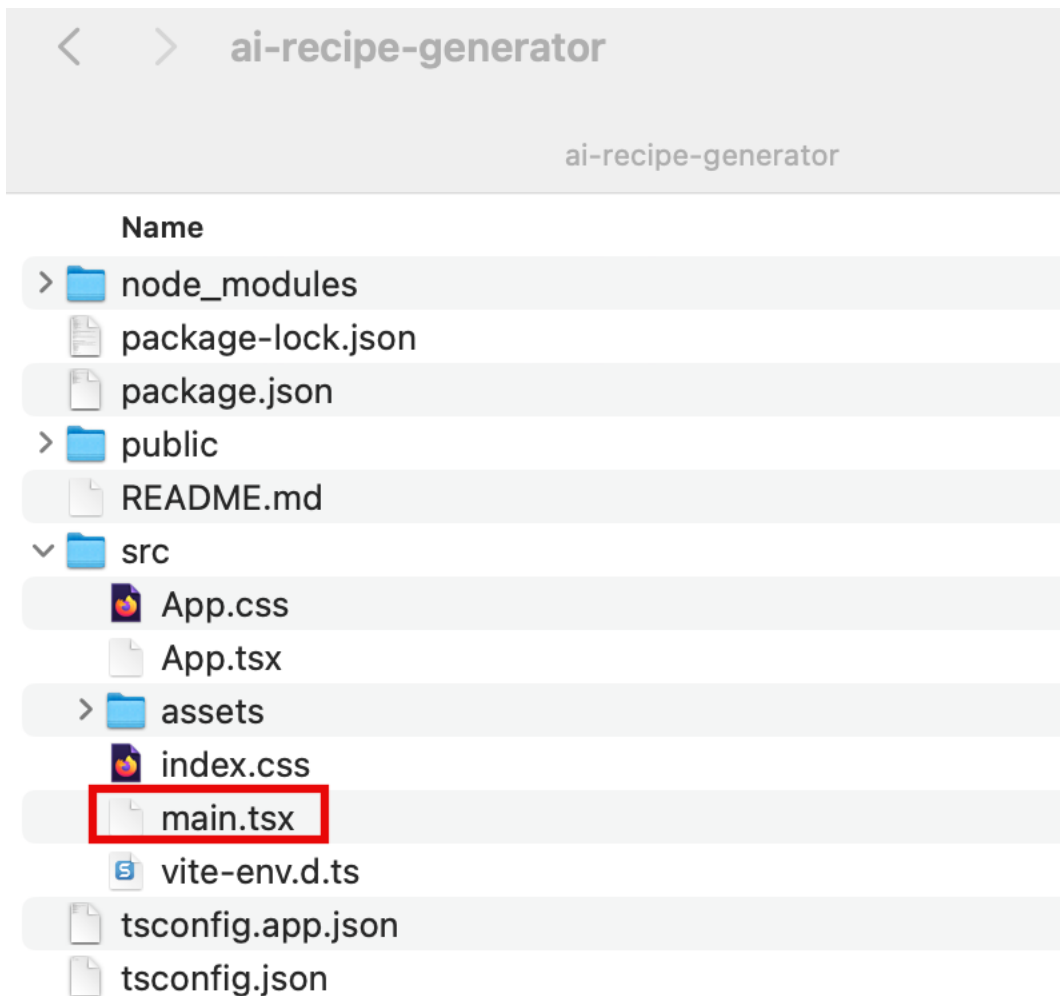
1. Add authentication

On your local machine, navigate to the **ai-recipe-generator/src/main.tsx** file, and **update** it with the following 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).

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import "./index.css";
import { Authenticator } from "@aws-amplify/ui-react";

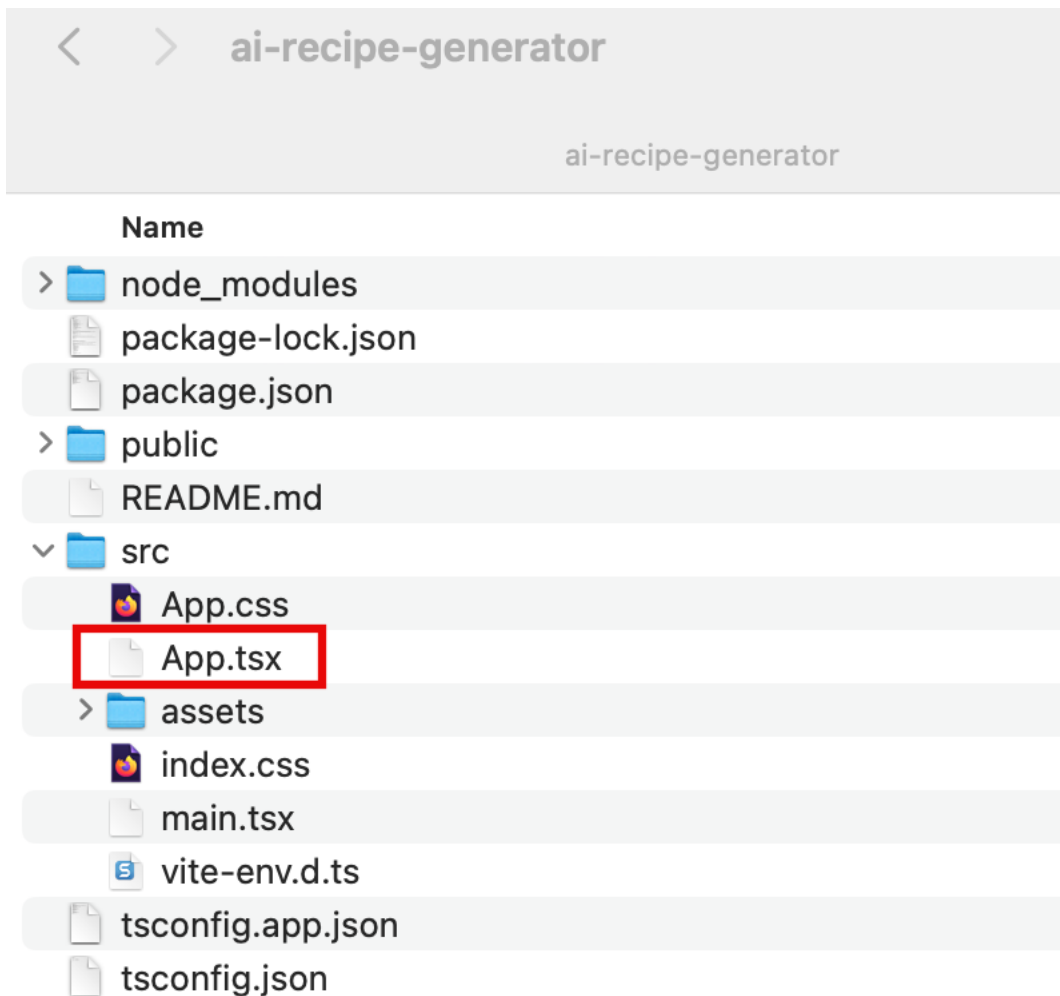
ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <Authenticator>
      <App />
    </Authenticator>
  </React.StrictMode>
);
```



2. Configure the Amplify library

Open the **ai-recipe-generator/src/App.tsx** 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 presents a form to users for submitting a list of ingredients. Once submitted, it will use the data client to pass the list to the **askBedrock** query and retrieve the generated recipe then display it to the user.



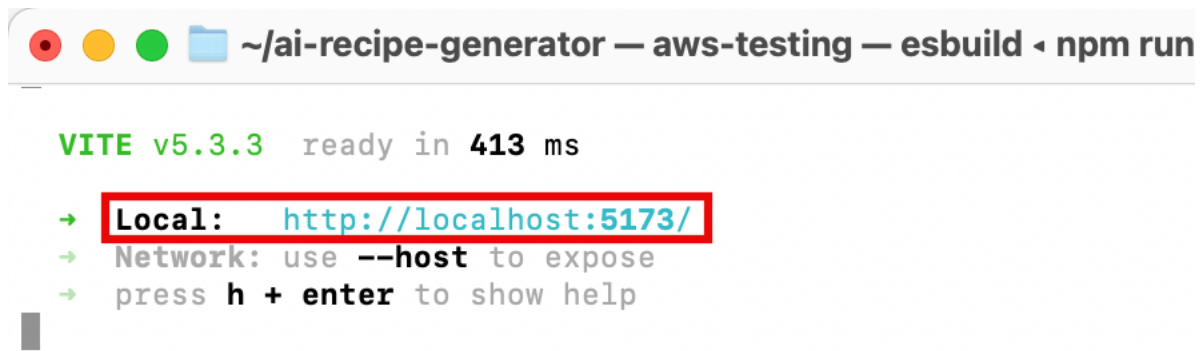
3. Launch the app

Open a new terminal window, **navigate** to your projects root directory (**ai-recipe-generator**), and **run** the following command to launch the app:

```
npm run dev
```

4. Open the app

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



```
~/ai-recipe-generator — aws-testing — esbuild ◀ npm run  
  
VITE v5.3.3 ready in 413 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

5. 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

6. Enter verification code

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

123456

Confirm

Resend Code

7. Generate recipes

When signed in, you can start **inputting ingredients** and **generating recipes**.

Meet Your Personal Recipe AI

Simply type a few ingredients using the format ingredient1, ingredient2, etc., and Recipe AI will generate an all-new recipe on demand...

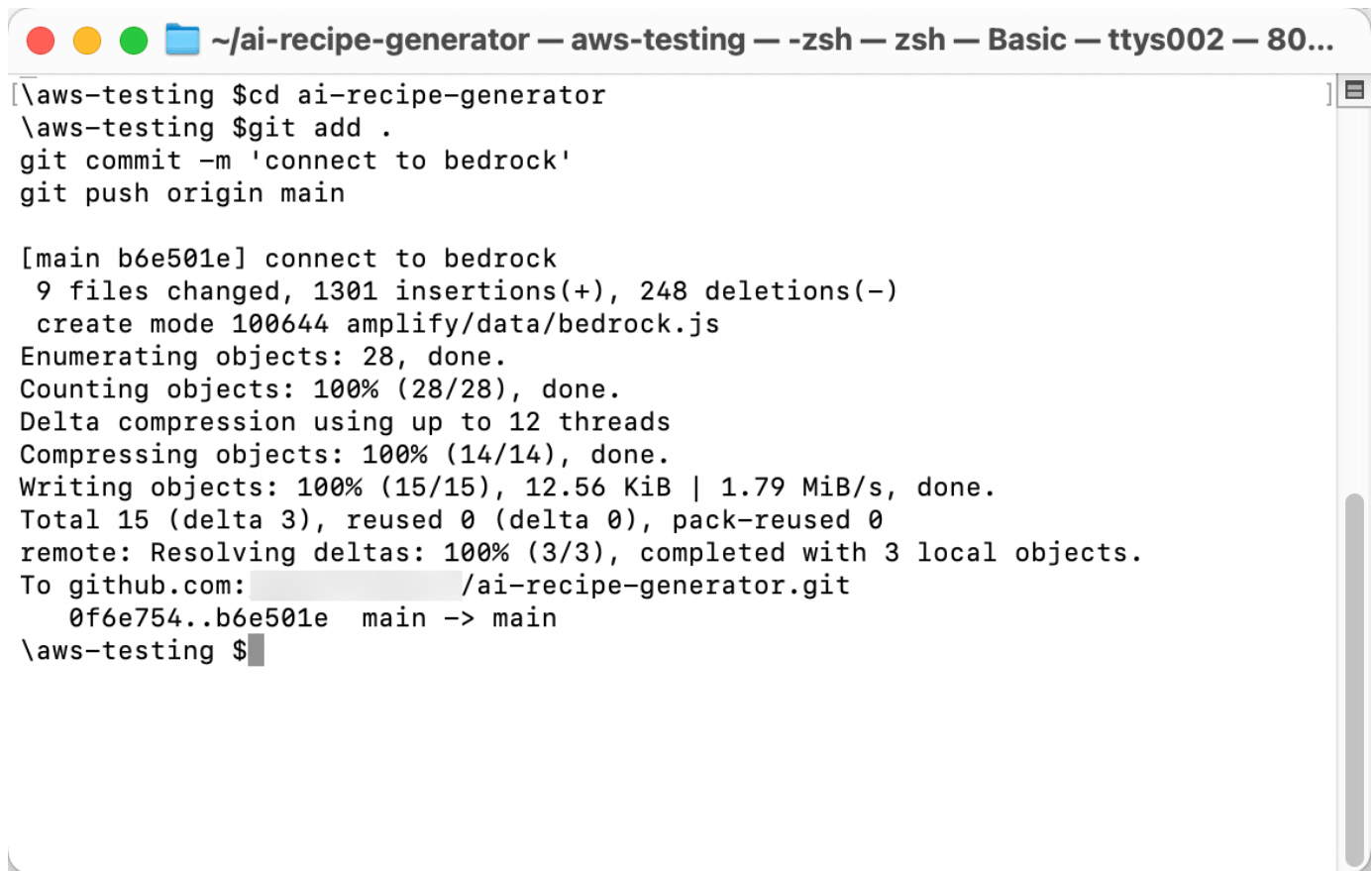
Chicken, white rice, yellow squash, onion

Generate

8. Push changes

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

```
git add .  
git commit -m 'connect to bedrock'  
git push origin main
```



```
~/ai-recipe-generator — aws-testing — -zsh — zsh — Basic — ttys002 — 80...

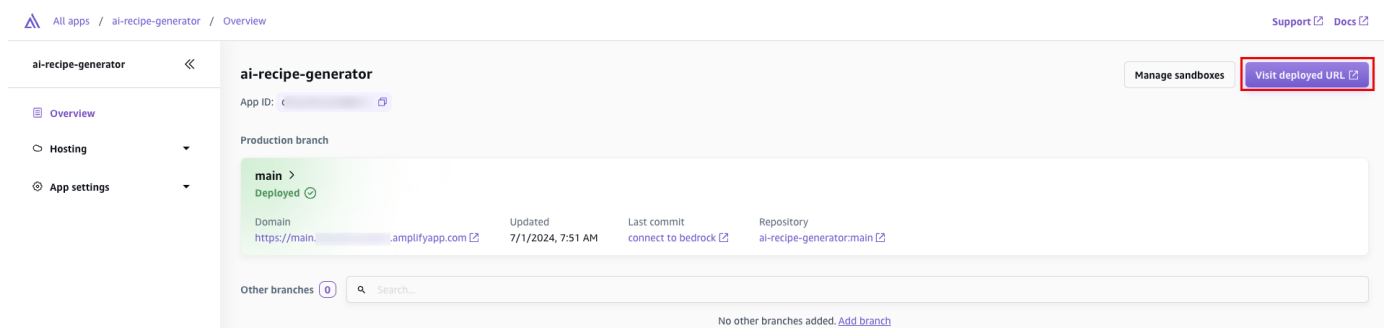
[\aws-testing $cd ai-recipe-generator
\aws-testing $git add .
git commit -m 'connect to bedrock'
git push origin main

[main b6e501e] connect to bedrock
 9 files changed, 1301 insertions(+), 248 deletions(-)
 create mode 100644 amplify/data/bedrock.js
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 12.56 KiB | 1.79 MiB/s, done.
Total 15 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com: [redacted]/ai-recipe-generator.git
   0f6e754..b6e501e  main -> main
\aws-testing $
```

9. View your changes

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.



Conclusion

You have now connected your app to the Amplify backend and built a frontend to generate a recipe based on a list of ingredients submitted by the user.

Task 6: Clean up Resources

Time to complete

<2 minutes

Overview

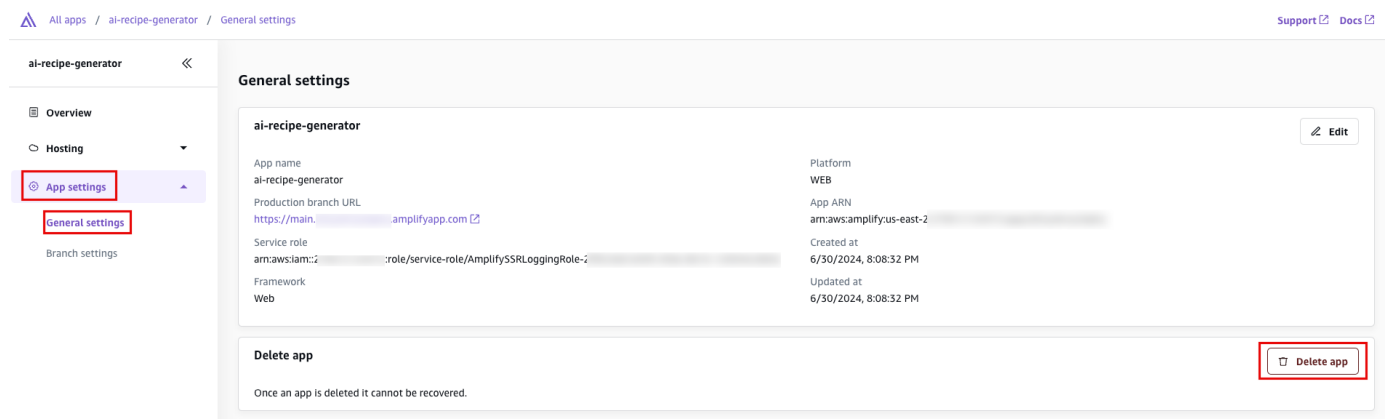
In this task, you will go through the steps to delete all the resources you created throughout this tutorial. It is a best practice to delete resources you are no longer using to avoid unwanted charges.

1. Open general settings

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

2. Delete the app

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



Congratulations

You have created a React web app and used Amplify and Amazon Bedrock to develop an AI-powered Recipe Generator App. Additionally, you've deployed the app on AWS using Amplify Hosting.