Frontend is for Everyone

**SPL-TF-300-MBFIFE-1 - Version 1.0.10**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

As a technology professional with scripting experience and a new idea, building a frontend for a proof of concept or minimum lovable product may seem out of reach. In this scenario, your team depends on you to provision Linux instances for their use. You have already improved this process by writing an AWS Lambda function that launches Elastic Compute Cloud (EC2) instances with appropriate tagging. What if there was a web-based self-service tool that doesn’t require logging into the AWS console? In this lab, you build a serverless, full-stack single-page app with authentication that allows users to launch and terminate EC2 instances.

You learn about the React library’s features for building user interfaces. Then you use AWS Amplify to add AWS services and integrate them with your frontend. This includes adding authentication via Amazon Cognito and creating a secure REST API powered by Amazon API Gateway. You further extend the backend with a serverless function via AWS Lambda that interacts with the Amazon EC2 service.

Finally, you use AWS Amplify to add hosting your app, allowing it to be delivered to end-users on the AWS content delivery network using Amazon CloudFront. All application development happens within the AWS Cloud9 IDE.

OBJECTIVES

By the end of this lab, you will be able to:

* Create and host a full-stack web app on AWS.
* Implement React concepts such as states, effects and props.
* Create auth, API, function, and hosting resources on AWS using AWS Amplify.
* Integrate your frontend with backend services.

DURATION

This lab requires approximately *60* minutes to complete.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.
* **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
* **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
* **Knowledge check:** An opportunity to check your knowledge and test what you have learned.
* **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
* **Task complete:** A conclusion or summary point in the lab.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

SERVICES USED IN THIS LAB

**AWS Amplify**

AWS Amplify is a set of purpose-built tools and features that lets frontend web and mobile developers quickly and easily build full-stack applications on AWS. With Amplify, developers are able to deliver rich cloud-enabled experiences by integrating a broad range of capabilities into their apps including authentication, cloud storage, databases, REST & GraphQL APIs, online/offline data, serverless business logic, location, push notifications and real-time interactive experiences.

Amplify applications are powered by AWS managed services like AWS Lambda, AWS AppSync, Amazon API Gateway, Amazon Cognito, Amazon DynamoDB and Amazon Location Service. And Amplify Hosting is a fully-managed web hosting service that serves content with low latency from Amazon’s reliable content delivery network (CDN) with hundreds of points of presence globally.

**Task 1: Create and run a blank React app in AWS Cloud9**

React is a declarative, component-based JavaScript library used to build user interfaces. Create React App (CRA) is a command line tool used to quickly create single-page React applications.

In this task, you use CRA to generate the React development environment for your EC2 Vendor app. You also start a local debugging session to preview your running app.

**Note:** You use **React functional components** throughout this lab. Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen.

TASK 1.1: LAUNCH THE AWS CLOUD9 IDE

1. From the **Lab Information** section to the left of these instructions, copy the **Cloud9Environment** URL link and in a new browser tab paste the link.

The browser takes you to the AWS Cloud9 environment that you use during this lab.

You don’t need the **Cloud9 Welcome screen** or any of the other default tabs that appear when you first launch **AWS Cloud9**.

1. Close each tab by choosing the **X**.

This section of the IDE is where you update various file throughout this lab.

**Consider:** Take a moment to familiarize yourself with the **AWS Cloud9** IDE interface.

* In the middle of the screen, a single terminal session is open in the editor. You can open multiple tabs in this window to edit files and run terminal commands.
* The file navigator appears on the left side of the screen. As you build out your AWS Cloud Development Kit (AWS CDK) environment and application, additional directories and files appear here.
* A gear icon is on the right side of the screen. Choosing this icon opens the AWS Cloud9 **Settings** panel.

**Note:** Every *AWS Cloud9* workspace is automatically assigned *AWS Identity and Access Management (IAM)* credentials. These credentials provide the workspace with limited access (based on your federated role) to some AWS services in your account. These are known as AWS managed temporary credentials.

TASK 1.2: GENERATE THE REACT DEVELOPMENT ENVIRONMENT

1. **Command:** In the AWS Cloud9 terminal, run the following commands to invoke Create React App to generate the React development environment for your app:

cd ~/environment

npx create-react-app ec2-vendor

* For **Ok to proceed? (y)**, answer **y** to install the package and CRA creates the environment.

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Creating a new React app in /home/ec2-user/environment/ec2-vendor.

Success! Created ec2-vendor at /home/ec2-user/environment/ec2-vendor

Inside that directory, you can run several commands:

npm start

Starts the development server.

npm run build

Bundles the app into static files for production.

npm test

Starts the test runner.

npm run eject

Removes this tool and copies build dependencies, configuration files

and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

cd ec2-vendor

npm start

Happy hacking!

1. After the process is complete, you see the **ec2-vendor** directory in the file browser. This is your React development environment, where your app is located.

**Consider:** Explore the directory structure of the app. Expand the *ec2-vendor* directory in the file browser.

* *node\_modules* contains dependencies used by the app.
* *public* contains your static files. *index.html* is the page template and must exist in this directory.
* *src* contains dynamic components.
  + *index.js* is the JavaScript entry point and must exist in this directory. It provides Document Object Model (DOM) specific methods that can be used at the top level of your app and as an escape hatch to get outside the React model if you need to.
  + By default, a React root is created for an element with the *root* ID, i.e. all the compiled React content goes into an HTML element with id of root. *Hint: look for this element in index.html.*
  + Within *index.js*, the script imports a **component** named **App** and React renders this component inside its root.
* *package.json* contains the overall configuration for the React project.

**Note:** There is one *@babel* dependency required that was not installed by default. Update the *package.json* file with this dependency.

1. Update the **package.json** file to include the missing **@babel** dependency with the following command:

jq '.dependencies = {"@babel/plugin-proposal-private-property-in-object": "^7.14.5"} + .dependencies' /home/ec2-user/environment/ec2-vendor/package.json > /home/ec2-user/environment/ec2-vendor/package.json.tmp && mv /home/ec2-user/environment/ec2-vendor/package.json.tmp /home/ec2-user/environment/ec2-vendor/package.json

**Expected output:**

*None, unless an error occurs.*

TASK 1.3: START APP SESSION

1. **Command:** Run the following commands to start the app locally on your AWS Cloud9 instance:

cd ~/environment/ec2-vendor

npm run start

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Compiled successfully!

You can now view ec2-vendor in the browser.

Local: http://localhost:8080

On Your Network: http://10.0.1.70:8080

Note that the development build is not optimized.

To create a production build, use npm run build.

webpack compiled successfully

**Note:** For the remainder of this lab, leave this terminal running the app. For future commands, start a new terminal by selecting **Window** in the AWS Cloud9 toolbar, then **New Terminal**.

TASK 1.4: PREVIEW APP

1. In the AWS Cloud9 toolbar, choose **Preview**, and then choose **Preview Running Application**.
2. This opens up a new window in the AWS Cloud9 IDE with a preview of the React app you just created. You should see a spinning logo.

**Hint:** AWS Cloud9 tabs can be moved around the IDE by dragging and dropping the tab header.

**Task complete:** You successfully created and ran a blank React app in AWS Cloud9.

**Task 2: Update the React app to use traditional HTML elements for the EC2 Vendor app**

In this task, you remove the CRA-created content and use standard HTML elements to build an outline of your app.

1. **Command:** In the second AWS Cloud9 terminal, run the following commands to remove the existing components:

cd ~/environment/ec2-vendor/src

rm -f \*

**Preview Window Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Compiled with problems:

×

ERROR

Module not found: Error: Can't resolve '/home/ec2-user/environment/ec2-vendor/src/index.js' in '/home/ec2-user/environment/ec2-vendor'

**Note:** If you switch back to the first terminal window, notice how the compiler returns errors. This is because the local running app compiles itself whenever the app changes and you have just removed the JavaScript entry point.

**First Terminal Window Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Failed to compile.

Module not found: Error: Can't resolve '/home/ec2-user/environment/ec2-vendor/src/index.js' in '/home/ec2-user/environment/ec2-vendor'

ERROR in main

Module not found: Error: Can't resolve '/home/ec2-user/environment/ec2-vendor/src/index.js' in '/home/ec2-user/environment/ec2-vendor'

webpack compiled with 1 error

1. **Command:** Go back to the second terminal window and run the following command to copy the HTML outline code into the app:

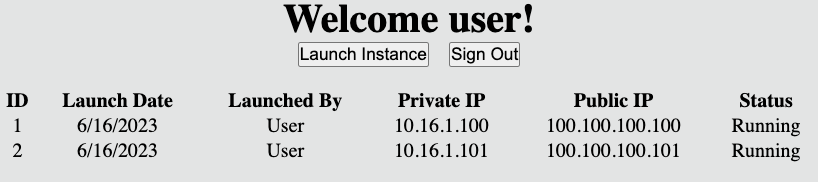
cp ~/environment/lab-resources/frontend/example1/\* ~/environment/ec2-vendor/src

**Expected output:**

*None, unless an error occurs.*

1. To refresh the preview, select the refresh icon in the upper-left section of the preview window.

**Expected output:**



*The image depicts the React application interface with a Welcome user! message followed by a Launch Instance button and a Sign Out button. Below that is a table with colum names for ID, Launch Date, Launched By, Private IP, Public Ip, and Status. There are two entries in the table.*

**Consider:** Explore the code in *ec2-vendor/src/App.js*:

* You return a centered *<div />* that contains several HTML elements.
* A *<header />* element at the top welcomes the user. Below it are two *<button />* elements that shows alert boxes when selected.
* Next you have a *<table />* with mock EC2 instance information.

**Task complete:** You successfully updated the React app to use traditional HTML elements for the EC2 Vendor app.

**Task 3: Update the app to use Amplify UI components**

In this task, you use Amplify UI, a **component library**, instead of HTML elements to build your app. A component library is a repository of customizable and reusable components that allow developers and designers to work consistently and save time. For example, instead of using *<button />* with CSS styling, you can import *<Button />* from a component library that provides additional features and styling built in.

1. **Command:** Run the following commands to install the Amplify UI component library:

cd ~/environment/ec2-vendor

npm install @aws-amplify/ui-react@^5

**Note:** The warnings can be ignored.

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

added 1183 packages, changed 1 package, and audited 2681 packages in 43s

1. **Command:** Run the following command to update your code to use the component library:

cp ~/environment/lab-resources/frontend/example2/\* ~/environment/ec2-vendor/src

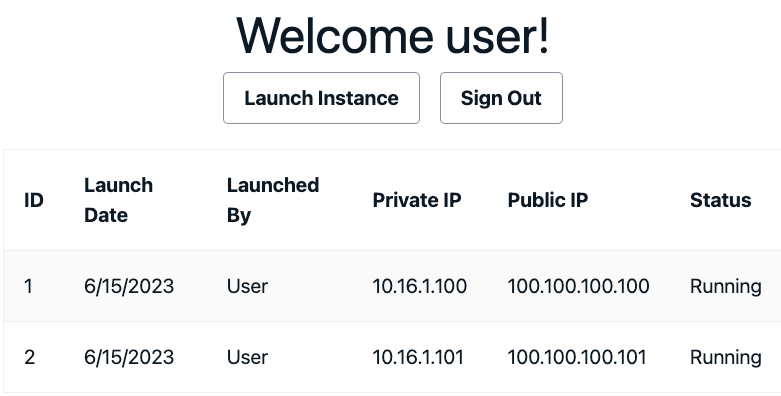
**Expected output:**

*None, unless an error occurs.*

1. Refresh the preview.

**Note:** The refresh may take a a while to load and/or display a timeout error. This is because the compilation is taking longer than usual due to having to compile the new component library and its dependencies. If you get a timeout error, try refreshing the preview again.

**Expected output:**



*The image depicts the React application interface with a Welcome user! message followed by a Launch Instance button and a Sign Out button. Below that is a table with colum names for ID, Launch Date, Launched By, Private IP, Public Ip, and Status. There are two entries in the table.*

**Consider:** Explore the code in *<App />*:

* You import several components from *@aws-amplify/ui-react*. Component libraries provide documentation on their components’ use.
* Instead of *<div />*, you wrap everything in a *<Card />*. Instead of *<header />* you use the component library’s *<Heading />*. While some HTML elements have a 1:1 mapping to a component library component, some do not. It’s best to look through the component library’s documentation to find the appropriate component to use.

**Task complete:** You successfully updated the app to use Amplify UI components.

**Task 4: Update the app to include states**

A React state is a built-in object that is used to store data belonging to that component. It should be used if the component needs to modify this data at some point. It should *not* be used if this value is derived from another value. When a component’s state changes, the component and any child components rerender.

For example, many websites provide an option to switch between a light or dark theme. When you first load the website, a theme state is created and contains an initial value. When you switch themes, you update the state value, and the website’s components are rerendered with the new theme options.

In this task, you update your app to support a dark or light theme, allow loading states in the buttons, and implement conditional rendering.

1. **Command:** Run the following command to update your code to use the states:

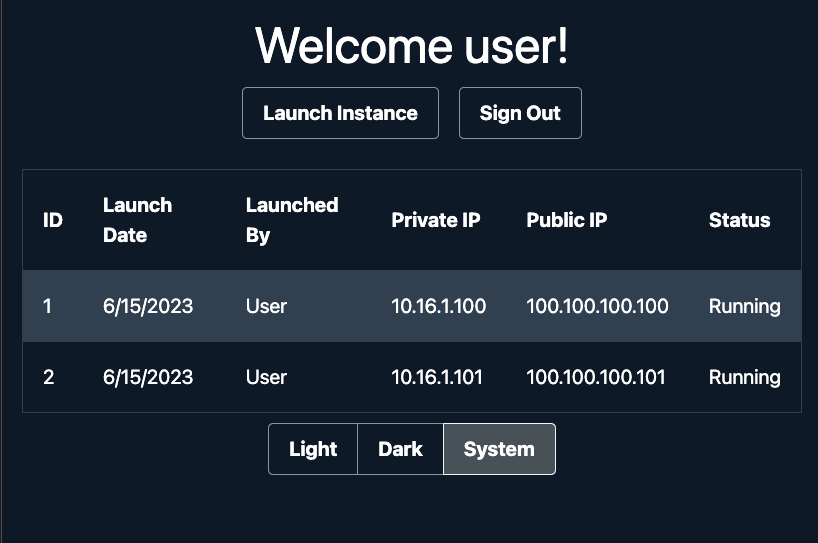
cp ~/environment/lab-resources/frontend/example3/\* ~/environment/ec2-vendor/src

**Expected output:**

*None, unless an error occurs.*

1. Refresh the preview.

**Expected output:**



*The image depicts a dark background for the React application interface with a Welcome user! message followed by a Launch Instance button and a Sign Out button. Below that is a table with colum names for ID, Launch Date, Launched By, Private IP, Public Ip, and Status. There are two entries in the table. Below the table are three buttons reading Light, Dark, and System.*

**Consider:** Explore the app:

* A toggle button group at the bottom of the page allows you to change between light, dark and system themes. Note the system theme was selected initially. This is the default value.
* When you select the **Launch Instance** button, the button becomes disabled and displays *Launching…*.
* When you select the **Sign Out** button, the button becomes disabled and displays *Signing Out…*.
* When either of the above buttons are selected, a **Reset** button appears. Selecting this button resets the other two buttons to their original state.

**Consider:** Explore the code in *<App />*:

* To be able to use states, you import *useState* from *react* at the top of the file.
* The states are defined immediately after the function definition. The syntax is *const [value, setValue] = useState(defaultValue)*. Note that *setvalue* is a function that allows updates to *value*.
* The *isLaunching* and *isSigningOut* states are set to false initially. Notice on their respective buttons’ *onClick* actions, you use *setIsLaunching* and *setIsSigningOut* to set the states to true. The Amplify UI component library’s *<Button />* component has a *isLoading* property, which when set to true, disables the button and displays the text in the *loadingText* property with a spinning circle.
* The reset button is only displayed when *isLaunching* or *isSigningOut* is set to true. This is conditional rendering. When this button is chosen, both states are set back to false. This causes those two buttons to revert to their original state, and the reset button is removed from view.
* The *colorMode* state uses the Amplify UI component library’s built-in theming to render the page based on its value. Notice the *<Card />* component is now wrapped in a *<ThemeProvider />* component. This is a feature of Amplify UI that allows you to dynamically change the theme of your app.

**Task complete:** You successfully updated the app to include states.

**Task 5: Update the app to include effects, props and a reusable component**

React effect hooks, implemented by *useEffect*, adds the ability to perform side effects from a function component. Examples of this include fetching data and timers.

React props are arguments passed into React components.

Reusable components is one of the core concepts of atomic design. As atoms are the building blocks of matter, in UI development, you can think of these atoms as basic UI components. For example, in your app’s table, the *<TableBody />* section contains the same few rows of *<TableCell />* surrounded by a *<TableRow />*. What if you created a reusable component that returned the table row given the row’s parameters?

In this task, you update your app with the ability to dynamically add and remove instances.

1. **Command:** Run the following command to update your code to use the effects and reusable components:

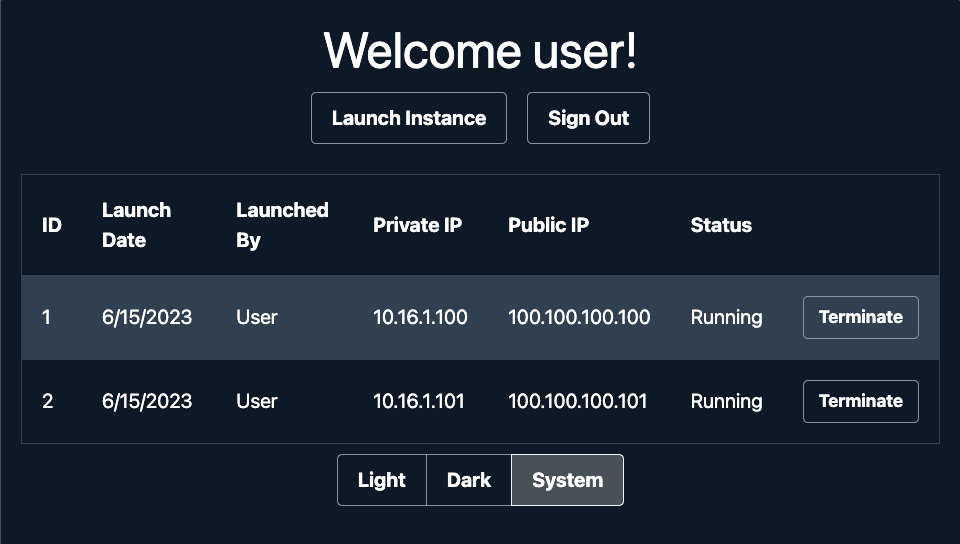
cp ~/environment/lab-resources/frontend/example4/\* ~/environment/ec2-vendor/src

**Expected output:**

*None, unless an error occurs.*

1. Refresh the preview.

**Expected output:**



*The image depicts a dark background for the React application interface with a Welcome user! message followed by a Launch Instance button and a Sign Out button. Below that is a table with colum names for ID, Launch Date, Launched By, Private IP, Public Ip, and Status. There are two entries in the table. To the right of the Status column is an unnamed column and in each entry in the table is a Terminate button. Below the table are three buttons reading Light, Dark, and System.*

**Consider:** Explore the app:

* Selecting the **Launch Instance** button adds a new row to the table.
* The table now has a **Terminate** button for every row. Selecting this button removes the row from the UI.

**Consider:** Explore the code in *<App />*:

* To be able to use effects, you import *useEffect* from *react* at the top of the file.
* A new state, *instances*, defaults to an empty array. This mimics a real-life scenario where on page load, you do not know the state of your instances.
* An *useEffect* is implemented that sets the instances state to an array containing mock EC2 instance data. This mimics a real-life scenario where you load the page first, then query an API to get EC2 data.

**Note:** The array at the end of the *useEffect* is the effect’s *dependencies*. If any variable listed in this array changes, the effect runs. Having an empty dependency list, i.e. an empty array, means that the effect only runs when the component first loads.

**Knowledge check:** Can you think of a scenario where the dependency list is not empty?

**Answer**

**Task complete:** You successfully updated the app to include effects, props and a reusable component.

**Task 6: Create an Amplify app using the CLI and add authentication services**

So far, you’ve built a frontend with mock data. Let’s begin adding AWS services to your app.

In this task, you install the Amplify CLI, create an Amplify app with an AWS Identity and Access Management (IAM) **permissions boundary**, and add authentication services based on Amazon Cognito.

A permissions boundary is an IAM feature that helps centralized cloud IAM teams to safely empower application developers to create new IAM roles and policies in AWS. Instead of logging tickets to your cloud team to create roles on your behalf, you are allowed to create roles yourselves, but only if you attach a permissions boundary to the role. This increases your development agility and allows you to bring features to market faster.

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Apple, Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0 and OpenID Connect.

1. **Command:** Run the following command to install the Amplify CLI:

npm install -g @aws-amplify/cli@^12

**Note:** The -g flag tells npm to install the package globally, instead of at the project level.

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

added 1182 packages, changed 1 package, and audited 2680 packages in 31s

Next you create the Amplify application using CLI commands. The first command is used to create a variable for the *ARN* of a *permissions boundary policy* with AmplifyPermissionBoundary in the name. Then it changes directories into the *ec2-vendor* directory and issues the command to create the Amplify application.

**Note:** For security purposes this requires specifying a permissions boundary policy. Since the creation of a role and an inline policy is required as part of creating the Amplify application, this permissions boundary policy restricts the permissions that can be used in the inline policy to ensure security of the lab.

1. **Command:** Run the following commands to create the Amplify app:

policy\_arn=$(aws iam list-policies --scope Local --query 'Policies[?contains(PolicyName, `AmplifyPermissionBoundary`)].Arn | [0]' --output text) && echo -e "\n Policy ARN: $policy\_arn\n"

cd ~/environment/ec2-vendor

amplify init --permissions-boundary $policy\_arn

* Accept the default for the project name: **ec2vendor**.
* Answer **Y** to initialize the project with the predefined configuration.
* Use **AWS profile** for authentication.
* Select the **default** profile.

**Expected output:** Output has been truncated.

Note: It is recommended to run this command from the root of your app directory

? Enter a name for the project ec2vendor

The following configuration will be applied:

Project information

| Name: ec2vendor

| Environment: dev

| Default editor: Visual Studio Code

| App type: javascript

| Javascript framework: react

| Source Directory Path: src

| Distribution Directory Path: build

| Build Command: npm run-script build

| Start Command: npm run-script start

? Initialize the project with the above configuration? Yes

Using default provider awscloudformation

? Select the authentication method you want to use: AWS profile

For more information on AWS Profiles, see:

https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

? Please choose the profile you want to use default

Adding backend environment dev to AWS Amplify app: dryf2k7c2wzr

Deployment completed.

Deploying root stack ec2vendor [ ---------------------------------------- ] 0/4

amplify-ec2vendor-dev-210943 AWS::CloudFormation::Stack CREATE\_IN\_PROGRESS Wed Mar 22 2023 21:09:44…

UnauthRole AWS::IAM::Role CREATE\_IN\_PROGRESS Wed Mar 22 2023 21:09:47…

DeploymentBucket AWS::S3::Bucket CREATE\_IN\_PROGRESS Wed Mar 22 2023 21:09:48…

AuthRole AWS::IAM::Role CREATE\_IN\_PROGRESS Wed Mar 22 2023 21:09:48…

✔ Help improve Amplify CLI by sharing non sensitive configurations on failures (y/N) · no

Deployment state saved successfully.

✔ Initialized provider successfully.

✅ Initialized your environment successfully.

Your project has been successfully initialized and connected to the cloud!

Some next steps:

"amplify status" will show you what you've added already and if it's locally configured or deployed

"amplify add <category>" will allow you to add features like user login or a backend API

"amplify push" will build all your local backend resources and provision it in the cloud

"amplify console" to open the Amplify Console and view your project status

"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and provision it in the cloud

Pro tip:

Try "amplify add api" to create a backend API and then "amplify push" to deploy everything

1. **Command:** Run the following command to add authentication services to the Amplify app:

**Note:** If at any time you select or input the wrong option, you can use **ctrl** + **c** to break out of the CLI and attempt it again.

amplify add auth

* Select **Default configuration** to use the preconfigured default.
* Select **Username** as the method of signing in.
* Select **No, I am done.** to complete the configuration.

**Expected output:**

Using service: Cognito, provided by: awscloudformation

The current configured provider is Amazon Cognito.

Do you want to use the default authentication and security configuration? Default configuration

Warning: you will not be able to edit these selections.

How do you want users to be able to sign in? Username

Do you want to configure advanced settings? No, I am done.

✅ Successfully added auth resource ec2vendorbee520f7 locally

✅ Some next steps:

"amplify push" will build all your local backend resources and provision it in the cloud

"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and provision it in the cloud

1. **Command:** Run the following command to deploy authentication services to AWS:

amplify push --yes

**Expected output:**

✔ Successfully pulled backend environment dev from the cloud.

Current Environment: dev

┌──────────┬───────────────────┬───────────┬───────────────────┐

│ Category │ Resource name │ Operation │ Provider plugin │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Auth │ ec2vendor1e2344a8 │ Create │ awscloudformation │

└──────────┴───────────────────┴───────────┴───────────────────┘

Deployment completed.

Deploying root stack ec2vendor [ ====================-------------------- ] 1/2

amplify-ec2vendor-dev-210943 AWS::CloudFormation::Stack UPDATE\_IN\_PROGRESS Wed Mar 22 2023 21:14:23…

authec2vendor1e2344a8 AWS::CloudFormation::Stack CREATE\_COMPLETE Wed Mar 22 2023 21:16:44…

Deployed auth ec2vendor1e2344a8 [ ======================================== ] 10/10

UserPool AWS::Cognito::UserPool CREATE\_COMPLETE Wed Mar 22 2023 21:14:37…

UserPoolClientWeb AWS::Cognito::UserPoolClient CREATE\_COMPLETE Wed Mar 22 2023 21:14:42…

UserPoolClient AWS::Cognito::UserPoolClient CREATE\_COMPLETE Wed Mar 22 2023 21:14:42…

UserPoolClientRole AWS::IAM::Role CREATE\_COMPLETE Wed Mar 22 2023 21:15:05…

UserPoolClientLambda AWS::Lambda::Function CREATE\_COMPLETE Wed Mar 22 2023 21:15:18…

UserPoolClientLambdaPolicy AWS::IAM::Policy CREATE\_COMPLETE Wed Mar 22 2023 21:15:41…

UserPoolClientLogPolicy AWS::IAM::Policy CREATE\_COMPLETE Wed Mar 22 2023 21:16:04…

UserPoolClientInputs Custom::LambdaCallout CREATE\_COMPLETE Wed Mar 22 2023 21:16:12…

IdentityPool AWS::Cognito::IdentityPool CREATE\_COMPLETE Wed Mar 22 2023 21:16:18…

IdentityPoolRoleMap AWS::Cognito::IdentityPoolRol… CREATE\_COMPLETE Wed Mar 22 2023 21:16:23…

Deployment state saved successfully.

**Consider:** Explore the resources created by Amplify:

* In AWS Cloud9, in the project’s *src* directory, open *aws-exports.js*. This file is generated by Amplify and allows the frontend easy access to the backend by referencing specific IDs.
* In your browser, switch to the *AWS Cloud9* tab. Use the *AWS search bar* to search for *Cognito*. Choose the service from the list of results.
* If available, select the *Try out the new interface* link.
* In the user pools list, select the *ec2vendor user pool*. After a user signs up, their information appears here.

**Task complete:** You successfully created an Amplify app using the CLI and added authentication services.

**Task 7: Update the app to use authentication**

Now that you have AWS authentication services configured on the backend, you use the *aws-amplify* library to link the frontend with the backend. This library provides a declarative interface across different categories of operations in order to make common tasks easier to add into your application.

In this task, you update your app with the authentication capabilities.

1. Switch back to the browser tab open to the **Cloud9-Lab-IDE** environment.
2. **Command:** Run the following commands to install the aws-amplify library:

cd ~/environment/ec2-vendor

npm install aws-amplify@^5

**Note:** The warnings can be ignored.

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

up to date, audited 2680 packages in 6s

1. **Command:** Run the following command to update your code to use authentication:

cp ~/environment/lab-resources/frontend/example5/\* ~/environment/ec2-vendor/src

**Expected output:**

*None, unless an error occurs.*

1. Refresh the preview.
2. You are now prompted with a sign in screen. To create an account, select the **Create Account** tab

* Input a username, password and **valid** email address
* Choose **Create Account**
* Wait for the validation code to arrive at your email, and enter the code at the next screen
* Choose **Confirm**. You are now taken to the app.

**Note:** It may take a few minutes for the verification code to arrive at your email address.

**Consider:** Explore the app:

* There is a text box that shows the user object. This object is passed in from the Amplify authenticator component and includes user-level attributes as well as session details, such as the JWT token. This object is typically used programmatically and is only shown here for learning purposes.
* The **Sign Out** button is functional and signs you out of the app.

**Consider:** Explore the code in *<App />*:

* You use a combination of the *aws-amplify*, Amplify UI libraries and *aws-exports.js* to implement authentication.
* *withAuthenticator* is a higher order component that wraps around the *App* component. If you are not authenticated, then the app is hidden and the authentication component is shown. If you are authenticated, then the app is shown and the authenticator passes in the user object and the sign out function to *App*.

**Note:** There are other options for using the authentication component directly or even writing your own components.

* The table component has been moved into its own file, named *InstanceTable.jsx*. It takes in props and returns the entirety of the instance table.

**Task complete:** You successfully updated the app to use authentication.

**Task 8: Create an AWS Lambda function**

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

In this task, you use Amplify to create a Node.js Lambda function that takes input from an API and interact with the EC2 service using the AWS SDK. This includes getting Amazon Machine Image (AMI) data, querying, starting and terminating EC2 instances.

TASK 8.1: CREATE THE FUNCTION

1. **Command:** Run the following command to add a Lambda function via Amplify:

cd ~/environment/ec2-vendor

amplify add function

* Select **Lambda function (serverless function)** for the capability.
* Enter **ec2VendorApi** for the function name.
* Select **NodeJS** for the runtime.
* Select the **Hello World** template.
* Answer **N** to configuring advanced settings and press **Enter**.
* Answer **N** to editing the local Lambda function now and press **Enter**.

**Expected output:**

? Select which capability you want to add: Lambda function (serverless function)

? Provide an AWS Lambda function name: ec2VendorApi

? Choose the runtime that you want to use: NodeJS

? Choose the function template that you want to use: Hello World

Available advanced settings:

- Resource access permissions

- Scheduled recurring invocation

- Lambda layers configuration

- Environment variables configuration

- Secret values configuration

? Do you want to configure advanced settings? No

? Do you want to edit the local lambda function now? No

Successfully added resource ec2VendorApi locally.

Next steps:

Check out sample function code generated in <project-dir>/amplify/backend/function/ec2VendorApi/src

"amplify function build" builds all of your functions currently in the project

"amplify mock function <functionName>" runs your function locally

To access AWS resources outside of this Amplify app, edit the /home/ec2-user/environment/ec2-vendor/amplify/backend/function/ec2VendorApi/custom-policies.json

"amplify push" builds all of your local backend resources and provisions them in the cloud

"amplify publish" builds all of your local backend and front-end resources (if you added hosting category) and provisions them in the cloud

1. **Command:** The Lambda code has been pre-written. Run the following command to update your local Lambda code:

cp -rp ~/environment/lab-resources/backend/ec2VendorApi/\* ~/environment/ec2-vendor/amplify/backend/function/ec2VendorApi

**Expected output:**

*None, unless an error occurs.*

**Consider:** Explore the code in the *ec2VendorApi* Lambda function:

* In the AWS Cloud9 file explorer, navigate to the following path: *ec2-vendor/amplify/backend/function/ec2VendorApi*. This is where the local Lambda function is located.
* *custom-policies.json* contains IAM statements that are given to the Lambda function execution role.

**Note:** Remember the IAM permissions boundary you specified when you ran *amplify init*? This permissions boundary is applied to every role that Amplify creates, including this Lambda execution role.

* The *src* directory contains the Lambda function code. The exported *handler* function in *index.js* is the entry into the function.
* Your example function contains simple conditional statements to call functions depending on the input:
  + GET HTTP method
    - */vendor/ami*: query EC2 for Amazon Linux 2 AMIs based on the architecture
    - */vendor/instance*: query EC2 for running EC2 instances based on the owner tag
  + POST HTTP method
    - */vendor/instance/launch*: launch an EC2 instance with appropriate tagging
    - */vendor/instance/terminate*: terminate an EC2 instance given the instance ID
* The function always returns a JSON with an HTTP status code of 200 and a variable body from the EC2 API output.
* The *src/event.json* file provides a sample input that you can use to test the function locally. This is covered in the next subtask.

TASK 8.2: TEST THE FUNCTION LOCALLY

In order to quickly test and debug without pushing all changes in your project to the cloud, Amplify supports Local Mocking and Testing for certain categories, including AWS Lambda.

1. **Command:** Before you can test the Lambda function locally, you must install the function’s dependencies. Run the following command to do so:

cd ~/environment/ec2-vendor/amplify/backend/function/ec2VendorApi/src/

npm install aws-sdk@^2

**Expected output:** Output is truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

added 32 packages, and audited 33 packages in 3s

1. **Command:** Run the following command to test the AMI query logic of the Lambda function:

amplify mock function ec2VendorApi --event "src/event.json"

**Expected output:** Output has been truncated.

Ensuring latest function changes are built...

Starting execution...

Result:

{

"statusCode": 200,

"headers": {

"Access-Control-Allow-Origin": "\*",

"Access-Control-Allow-Headers": "\*"

},

"body": "{\"Images\":[{\"Architecture\":\"x86\_64\",...\"Tags\":[],\"VirtualizationType\":\"hvm\",\"DeprecationTime\":\"2023-07-27T06:22:54.000Z\"}]}"

}

Finished execution.

**Task complete:** You successfully created an AWS Lambda function.

**Task 9: Create a secure REST API backed by an AWS Lambda function**

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the “front door” for applications to access data, business logic, or functionality from your backend services.

In this task, you use Amplify to create a secure REST API powered by Amazon API Gateway. This is backed by the Lambda function you created in the previous task.

1. **Command:** Run the following command to add a secure REST API via Amplify:

cd ~/environment/ec2-vendor

amplify add api

* Select **REST** for the API type.
* Enter **vendorApi** for the API friendly name.
* Enter **/vendor** for the path.
* Use the arrow down key to select **Use a Lambda function already added in the current Amplify project** for the Lambda source. Since you only have one Lambda function, the CLI selects this automatically.
* Answer **Y** to restrict API access.
* Select **Authenticated users only** for who should have access.
* Use **ctrl** + **a** to grant all options to authenticated users.
* Answer **N** to not add another path.

**Expected output:**

? Select from one of the below mentioned services: REST

✔ Provide a friendly name for your resource to be used as a label for this category in the project: · vendorApi

✔ Provide a path (e.g., /book/{isbn}): · /vendor

✔ Choose a Lambda source · Use a Lambda function already added in the current Amplify project

Only one option for [Choose the Lambda function to invoke by this path]. Selecting [ec2VendorApi].

✔ Restrict API access? (Y/n) · yes

✔ Who should have access? · Authenticated users only

✔ What permissions do you want to grant to Authenticated users? · create, read, update, delete

✔ Do you want to add another path? (y/N) · no

✅ Successfully added resource vendorApi locally

✅ Some next steps:

"amplify push" will build all your local backend resources and provision it in the cloud

"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and provision it in the cloud

1. **Command:** Run the following command to deploy the secure REST API and Lambda function to AWS:

amplify push --yes

**Expected output:**

✔ Successfully pulled backend environment dev from the cloud.

Current Environment: dev

┌──────────┬───────────────────┬───────────┬───────────────────┐

│ Category │ Resource name │ Operation │ Provider plugin │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Function │ ec2VendorApi │ Create │ awscloudformation │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Api │ vendorApi │ Create │ awscloudformation │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Auth │ ec2vendora1b2c3d4 │ No Change │ awscloudformation │

└──────────┴───────────────────┴───────────┴───────────────────┘

⚠️ Warning: You've specified "\*" as the "Resource" in ec2VendorApi's custom IAM policy.

This will grant ec2VendorApi the ability to perform ec2:DescribeImages,ec2:DescribeInstances on ALL resources in this AWS Account.

Deployment completed.

Deployed root stack ec2vendor [ ======================================== ] 4/4

amplify-ec2vendor-dev-173538 AWS::CloudFormation::Stack UPDATE\_COMPLETE

functionec2VendorApi AWS::CloudFormation::Stack CREATE\_COMPLETE

authec2vendora1b2c3d4 AWS::CloudFormation::Stack UPDATE\_COMPLETE

apivendorApi AWS::CloudFormation::Stack CREATE\_COMPLETE

Deployed function ec2VendorApi [ ======================================== ] 4/4

LambdaExecutionRole AWS::IAM::Role CREATE\_COMPLETE

LambdaFunction AWS::Lambda::Function CREATE\_COMPLETE

CustomLambdaExecutionPolicy AWS::IAM::Policy CREATE\_COMPLETE

lambdaexecutionpolicy AWS::IAM::Policy CREATE\_COMPLETE

Deployed api vendorApi [ ======================================== ] 5/5

vendorApi AWS::ApiGateway::RestApi CREATE\_COMPLETE

vendorApiDefault5XXResponse AWS::ApiGateway::GatewayRespo… CREATE\_COMPLETE

vendorApiDefault4XXResponse AWS::ApiGateway::GatewayRespo… CREATE\_COMPLETE

functionec2VendorApiPermissio… AWS::Lambda::Permission CREATE\_COMPLETE

DeploymentAPIGWvendorApi9cf8d… AWS::ApiGateway::Deployment CREATE\_COMPLETE

REST API endpoint: https://a1b2c3d4e5.execute-api.us-west-2.amazonaws.com/dev

**Note:** The following warning can be safely ignored.

🛑 Failed to upload AMPLIFY\_function\_ec2VendorApi\_deploymentBucketName to ParameterStore

**Consider:** In the terminal, select the URL for the REST API endpoint and select *Open*. This opens the API URL in a new window. You should get a message of *Missing Authentication Token*. This is because you chose to create a secure API that uses IAM authentication. This prevents anonymous access to the API. You can close this new tab.

**Task complete:** You successfully created a secure REST API backed by an AWS Lambda function.

**Task 10: Update the app to use the API to query Amazon Linux 2 AMI data**

Before you implement launching and terminating EC2 instances, let’s implement something simpler: querying Amazon Machine Images (AMIs) from the frontend using the API and Lambda function.

An Amazon Machine Image (AMI) is a supported and maintained image provided by AWS that provides the information required to launch an instance.

In this task, you update your app to interact with the secure REST API and query for Amazon Linux 2 AMI data based on x86\_64 or arm64 architecture.

1. **Command:** Run the following command to update your code to use the REST API:

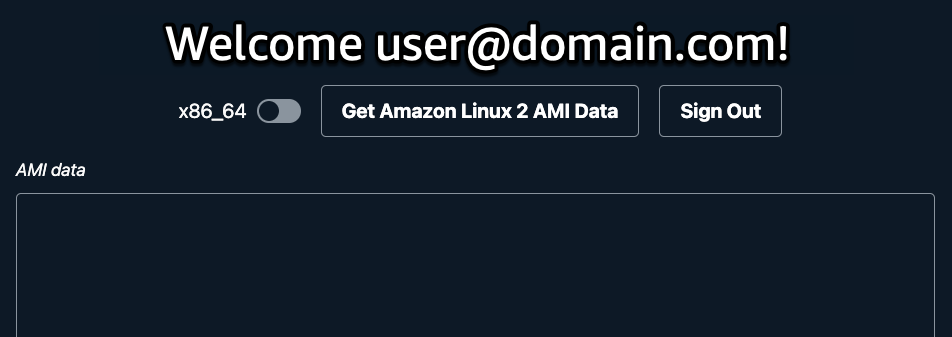
cp ~/environment/lab-resources/frontend/example6/\* ~/environment/ec2-vendor/src

**Expected output:**

*None, unless an error occurs.*

1. Refresh the preview.

**Expected output:**



*The image depicts the Amplify app interface that welcomes the logged in user by their email address. Below this is a toggle switch that lets you pick between x86\_64 or arm64 architectures for an AMI. To the right of this toggle switch is a button labeled Get Amazon Linux 2 AMI Data and to the right of this button is a Sign Out button. Below this is a window with a title of AMI data that will be populated with AMI details in JSON format when you choose the Get Amazon Linux 2 AMI Data button.*

**Consider:** Explore the app:

* There is a toggle switch that lets you pick between x86\_64 or arm64 architectures.

**Knowledge check:** What is causing the toggle button text to switch between x86\_64 and arm64?

**Answer**

**Consider:** Explore the code in *<App />*:

* You import the *API* interface from the aws-amplify library to interact with the API. Behind the scenes, the interface is using an HTTP client library to execute the HTTP requests.
* An async function enables asynchronous, promise-based behavior. *handlGetAmi* is an async function that:
  + Sets *setIsLaunching* to true. This causes the button to become disabled and displays the spinning circle, which provides feedback to the user that the app is doing something.
  + Calls the REST API with a GET request using the */vendor/ami* path. This returns a JSON object of AMI data, same as your Lambda test in the previous task.
  + Sets the amiData state to the JSON object. This causes the app to rerender and display the JSON object in the text field.
  + Sets *setIsLaunching* back to false.
* *getLatestAmiId* is a helper function that parses through the AMI JSON object and returns the latest AMI ID. You use this to decide which AMI to launch your EC2 instances from in the next task.

**Task complete:** You successfully updated the app to use the API to query Amazon Linux 2 AMI data.

**Task 11: Update the app to launch and terminate Amazon EC2 instances**

In this task, you update your app to launch and terminate EC2 instances.

1. **Command:** Run the following command to update your app:

cp ~/environment/lab-resources/frontend/example7/\* ~/environment/ec2-vendor/src

**Expected output:**

*None, unless an error occurs.*

1. Refresh the preview.

**Consider:** Explore the app:

* The instance list state is initially empty. When the page loads, the instance list is refreshed.
* The **Launch** button launches an EC2 instance and returns EC2 metadata. The EC2 instance is also tagged with the user who launched it.
* The **Terminate** button becomes active when the EC2 instance is in a running state. Selecting this button terminates the instance.

**Consider:** Explore the code in *<App />*:

* *refreshInstances*, *handleLaunchInstance*, and *handleTerminateInstance* have the same general concept:
  + Set the “loading” state to true.
  + Call the REST API to refresh instances, or launch or terminate an instance.
  + Set the “loading” state to false.
* On initial page load, the instance list is refreshed. This is because you call *refreshInstances* in an useEffect hook.

**Task complete:** You successfully updated the app to launch and terminate Amazon EC2 instances.

**Task 12: Host the app on Amazon CloudFront**

AWS Amplify Hosting is a fully managed CI/CD and hosting service for fast, secure, and reliable static and server-side rendered apps that scale with your business. Your app is deployed to Amazon CloudFront’s content delivery network (CDN) with hundreds of points of presence globally.

In this task, you use Amplify to add hosting and deploy your app to Amazon CloudFront.

1. **Command:** Run the following command to add hosting via Amplify:

cd ~/environment/ec2-vendor

amplify add hosting

* Select **Hosting with Amplify Console (Managed hosting with custom domains, Continuous deployment)** for the module to execute.
* Select **Manual deployment** for the type.

**Expected output:**

✔ Select the plugin module to execute · Hosting with Amplify Console (Managed hosting with custom domains, Continuous deployment)

? Choose a type Manual deployment

You can now publish your app using the following command:

Command: amplify publish

1. **Command:** Run the following command to publish the app. During this process, Amplify creates an optimized build of your app for hosting.

amplify publish --yes

**Expected output:**

✔ Successfully pulled backend environment dev from the cloud.

Current Environment: dev

┌──────────┬───────────────────┬───────────┬───────────────────┐

│ Category │ Resource name │ Operation │ Provider plugin │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Hosting │ amplifyhosting │ Create │ awscloudformation │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Auth │ ec2vendor1e2344a8 │ No Change │ awscloudformation │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Function │ ec2VendorApi │ No Change │ awscloudformation │

├──────────┼───────────────────┼───────────┼───────────────────┤

│ Api │ vendorApi │ No Change │ awscloudformation │

└──────────┴───────────────────┴───────────┴───────────────────┘

Deployment completed.

Deploying root stack ec2vendor [ ================================-------- ] 4/5

amplify-ec2vendor-dev-210943 AWS::CloudFormation::Stack UPDATE\_IN\_PROGRESS Wed Mar 22 2023 21:36:58…

authec2vendor1e2344a8 AWS::CloudFormation::Stack UPDATE\_COMPLETE Wed Mar 22 2023 21:37:05…

hostingamplifyhosting AWS::CloudFormation::Stack CREATE\_COMPLETE Wed Mar 22 2023 21:37:17…

functionec2VendorApi AWS::CloudFormation::Stack UPDATE\_COMPLETE Wed Mar 22 2023 21:37:06…

apivendorApi AWS::CloudFormation::Stack UPDATE\_COMPLETE Wed Mar 22 2023 21:37:09…

Deployed hosting amplifyhosting [ ======================================== ] 1/1

Deployment state saved successfully.

Publish started for amplifyhosting

> ec2-vendor@0.1.0 build

> react-scripts build

Creating an optimized production build...

Compiled successfully.

File sizes after gzip:

261.32 kB build/static/js/main.f356e6ca.js

33.01 kB build/static/css/main.746f56ac.css

The project was built assuming it is hosted at /.

You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.

You may serve it with a static server:

npm install -g serve

serve -s build

Find out more about deployment here:

https://cra.link/deployment

✔ Zipping artifacts completed.

✔ Deployment complete!

https://dev.dryf2k7c2wzr.amplifyapp.com

1. To access the public facing app, copy and paste the URL from the published output into a new tab.
2. You are taken to the authentication page, log in with the user name and password you used earlier.

**Note:** You are allowed spin up to 2 instances in this lab. Any additional instances will be deleted automatically.

**Task complete:** You successfully hosted the app on Amazon CloudFront.

**Task 13: Cleanup**

When you are done with resources you are no longer going to use, it is always best to remove those resources to keep from incurring charges.

In this task, you use the AWS Amplify CLI to remove all resources you built during the lab: Cognito user and identity pools, API Gateway REST API, Lambda function and CloudFront hosting.

**Note:** After this task, the app no longer functions.

1. **Command:** Run the following command to update your app:

cd ~/environment/ec2-vendor

amplify delete

* Answer **Y** to delete all Amplify resources.

**Expected output:**

✔ Are you sure you want to continue? This CANNOT be undone. (This will delete all the environments of the project from the cloud and wipe out all the local files created by Amplify CLI) (y/N) · yes

⠋ Deleting resources from the cloud. This will take a few minutes.

Deleting env: dev.

✔ Project deleted in the cloud.

✅ Project deleted locally.

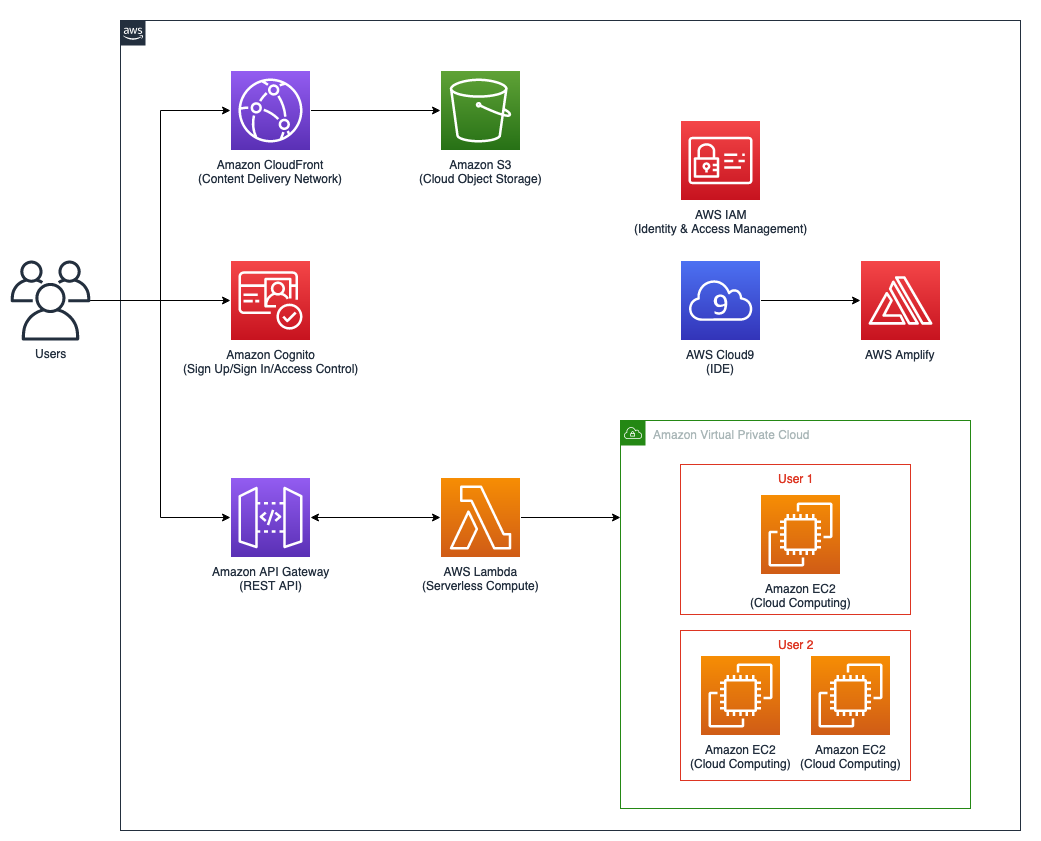
**Task complete:** You successfully used the AWS Amplify CLI to remove all resources you built during the lab.

**Conclusion**

**Task complete:** You now have successfully:

* Created a full-stack web app on AWS.
* Implemented React concepts such as states, effects and props.
* Created auth, API, function, and hosting resources on AWS using AWS Amplify.
* Integrated your frontend with backend services.

The architecture that is built during this lab is as follows:



*The diagram depicts the use of the following resources:*

* *Amazon Cognito used to sign up/sign in to the app*
* *Amazon CloudFront*
* *Amazon S3*
* *Amazon API Gateway*
* *AWS Lambda*
* *Amazon EC2*
* *Amazon VPC*
* *AWS IAM*
* *AWS Cloud9*
* *AWS Amplify*

You used AWS Amplify to minimize heavy lifting and created a first iteration of a PoC. By using Amplify, you gain development agility and can focus more on business needs.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional resources**

* [AWS Amplify Documentation](https://docs.aws.amazon.com/amplify/)
* [Amazon API Gateway Documentation](https://docs.aws.amazon.com/apigateway/index.html)
* [AWS Lambda Documentation](https://docs.aws.amazon.com/lambda/index.html)
* [AWS Identity and Access Management Documentation](https://docs.aws.amazon.com/iam/index.html)
* [Permissions boundaries for IAM entities](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html)

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).