AWS Services are loading.

aws training and certification

Lab 6: Capstone - Complete the Application Build

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* .

Duration

This lab will require **60 minutes** to complete.

Prerequisites

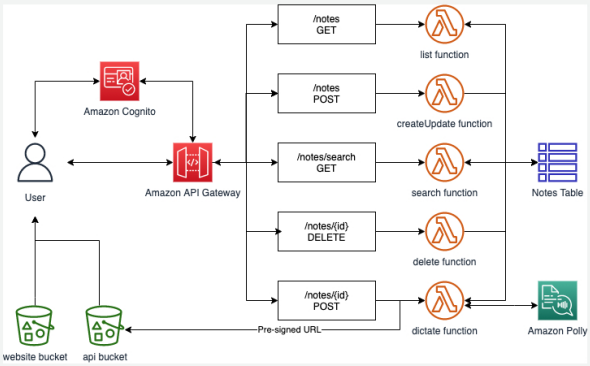
This lab requires:

- Access to a Microsoft Windows or MacOS notebook computer with a WI-FI connection.
- An Internet browser such as Chrome, Firefox, or IE9+.
- **Note:** Previous versions of Internet Explorer are not supported.
- **Note:** You can use an iPad or tablet device to access these directions in the lab console.
- **Additional information:** Review additional lab environment specific details in the Appendix.

Overview

You have all of the components created for your application and have learned how to allow access to them from the web. Before adding any more API methods for the remaining functions you need to be able to secure access. In this lab, you will learn how to set up Amazon Cognito to authenticate and authorize users to use your API. With that in place, you will then configure the front-end web application to complete your fully functional application.

Once you have completed the lab, your deployment will look like following diagram:



OBJECTIVES

After completing this lab, you will be able to:

- Create a user pool and an app client for your web application using Amazon Cognito.
- Add new users and confirm their ability to sign-in using the Amazon Cognito CLI.
- Configure API Gateway methods to use Amazon Cognito as an authorizer.
- Verify JWT authentication tokens are generated during API calls.
- Develop API Gateway resources rapidly using Swagger importing strategy.
- Set up your web application frontend to use Amazon Cognito and API Gateway configurations and verify the entire application functionality.

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.

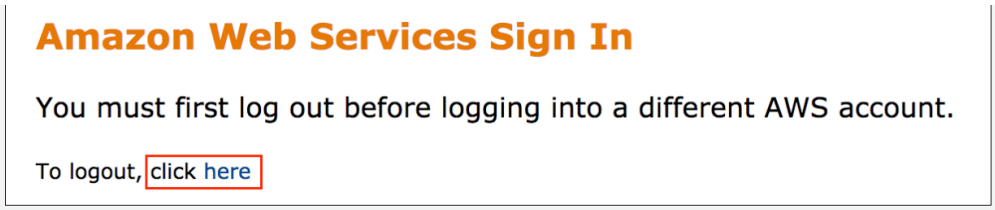
2. To open the lab, choose Open Console .

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.

Task 1: Configure Amazon Cognito

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. You will use Amazon Cognito to manage a directory of users and grant access to your API. In this task you will create a user pool, user, and confirm that the user can successfully authenticate.

Consider: This lab is designed for both experienced and newer developers:

- For more experienced developers who enjoy a challenge, there are **High-Level Instructions** before each task that should provide you enough information to help you complete the task.
- Once you complete the updates test your code to ensure it works, troubleshoot if needed, and then move on to the next task.
- For newer developers, there are **Detailed Instructions** to guide you through each step of the lab.

High-Level Instructions

- Create an Amazon Cognito user pool named PollyNotesP... with only a username attribute and minimal password requirements for testing purpose.
- Create an Amazon Cognito application client for your application without a secret key.
- In the AWS Cloud9 terminal, add a new user named stud... and approve it in the user pool using the cognito-idp CLI commands sign-up and admin-confirm-sign-up.
- Test authentication using the URL from the **testLoginWebsite** output to the left of these instructions.
- Review JWT token created by your application using .getIdToken().getJwtToken() methods.

Detailed Instructions

TASK 1.1: CREATE AN AMAZON COGNITO USER POOL WITH AN APP CLIENT

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Your users can also sign in through social identity providers like Google, Facebook, Login withAmazon, or Apple, and through Security Assertion Markup Language (SAML) 2.0 identity providers. Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through a Software Development Kit (SDK).

3. From the **AWS Management Console** , use the **AWS search bar** to search for Cogn... and then choose the service from the list of results.

Note: This lab uses the **latest** Cognito interface. You may need to choose to use the new interface from the top-right link **Try out the new interface** if the instructions do not match what you see in the console.

4. Choose **Create user pool** and make the following selections:

Authentication providers:

- **Provider types:** Select ☑ Cognito user pool (This should already be selected and grayed out).
- **Cognito user pool sign-in options:** Select ☑ User name

5. Choose **Next** .

In the **Configure security requirements** section, you set the specifics required for the password. The password used is for testing purposes and you will be entering it many times so it has limited security settings.

6. Choose the following configurations:

Password policy:

- **Password policy mode:** ◉ Custom
- **Password minimum length** 6 character(s)

Password requirements: (Unselect all)

- ☐ **Contains at least 1 number**
- ☐ **Contains at least 1 special character (^ $ * . [ ] { } ( ) ? - " ! @ # % & / \ , > < ' : ; | _ ~ ` + =)**
- ☐ **Contains at least 1 uppercase letter**
- ☐ **Contains at least 1 lowercase letter**
- Set the *Temporary passwords set by administrators expire in* value to 7 day(s)

Multi-factor authentication:

- Select ◉ No MFA

User account recovery:

- You will not need account recovery in this lab. Ensure this box is de-selected.
- ☐ Enable self-service account recovery - Recommended

7. Choose **Next** .

Self-service sign-up:

- Enable self-registration by checking the box provided (it should be checked by default).
- **Self-registration:** ☑ Enable self-registration

Attribute verification and user account confirmation:

- You will not need to verify messages for this lab. You may deselect the box.
- ☐ Allow Cognito to automatically send messages to verify and confirm - Recommended

8. Choose **Next** .

9. Set the following options for **Configure message delivery:**

Email:

- Configure your user pool to use Cognito to send email messages to users.
- **Email provider:** ◉ Send email with Cognito

10. Choose **Next** .

11. Set the following options for **Integrate your app:**

User pool name:

- **User pool name:** PollyNotesPool

Hosted authentication pages:

- ☐ Use the Cognito Hosted UI

Initial app client:

- **App type:** ◉ Public client
- **App client name:** PollyNotesWeb
- **Client secret:** ◉ Don't generate a client secret

Note: The App Client secret would be used in situations where there isn't a username and password used, like machine to machine communication. This secret isn't supported or needed in the AWS Javascript SDK.

12. Choose **Next** .

Review and create:

13. Review selections and once satisfied, choose **Create user pool** .

An app is an entity within a user pool that has permission to call unauthenticated API operations (operations that do not have an authenticated user). Unauthenticated API operation examples include operations to register, sign in, and handle forgotten passwords. To call these API operations, you need an app client ID and an optional client secret. It is your responsibility to secure any app client IDs or secrets so that only authorized client apps can call these unauthenticated operations.

14. Choose the PollyNotesPool link.

15. Save the **User Pool Id** and the **User Pool ARN** from your Amazon Cognito Pool information into a separate file for use later in this lab.

Note: Make sure to note that it is for **Amazon Cognito Pool Id** and **Amazon Cognito Pool ARN** as well.

For example, in the Oregon region, the **User Pool Id** would look like **us-west-2_XXXXXXXXX** and the **User Pool ARN** would look like **arn:aws:cognito-idp:us-west-2:012345678901:userpool/us-west-2_XXXXXXXXX** .

16. Choose the **App Integration** tab.

17. Scroll down to the **App client list** section.

18. Save the **Client ID** in the same file. Make sure to note that it is for the **Amazon Cognito App Client ID** .

TASK 1.2: ADD A NEW USER TO THE USER POOL

After you create your user pool, you can create users using the AWS Management Console, the AWS Command Line Interface (CLI), or the Amazon Cognito API. In a production environment, you usually create a profile for a new user in a user pool manually or with sign-up functionality built into the application. A welcome message with confirmation instructions would then be sent to the user via SMS or email.

For this lab, you will create and confirm the user with the AWS CLI instead of requiring SMS or email. Confirming the user as an administrator cannot be done with the AWS Management Console. To access the AWS CLI and the lab code, you will use AWS Cloud9.

19. From the **AWS Management Console** , use the **AWS search bar** to search for Cloud9 and then choose the service from the list of results.

20. On the **Environments** page, next to the **Lab6** environment listing, choose **Open** .

Note: Make sure the Cloud9 environment is **ready** . You can verify this by checking the folder structure for folders named **api** , **web** , and **Lab-Is-Ready** . If you don't see all three of them, just wait a few moments for the final configurations to complete.

Note: If you see the following message choose **Accept** .

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

21. In the terminal window, create three variables to use in later commands.

- ⌨ **Command:** Replace *[apiURL]* with the value from the left of these instructions and run the command below:

```
apiURL='[apiURL]'
```

Example:

apiURL='https://1gfw6ya7le.execute-api.us-east-1.amazonaws.com/Prod'

Expected output:

None, unless there is an error.

```
CognitoPoolId='[User Pool Id]'
```

Example:

CognitoPoolId='us-west-2_HtiTqUaLg'

Expected output:

None, unless there is an error.

- ⌨ **Command:** Replace *[App Client Id]* with information from Amazon Cognito and run the command below:

```
AppClientId='[App Client Id]'
```

Example:

AppClientId='7obbqbpsbu606efniqma1qceng'

Expected output:

None, unless there is an error.

22. ⌨ **Command:** Create a new Amazon Cognito user named **student** with the command below:

```
aws cognito-idp sign-up --client-id $AppClientId --username student --password student
```

Expected output:

```
*******************************
**** This is OUTPUT ONLY ****
```

```
******************************

{
    "UserConfirmed": false,
    "UserSub": "d946bb86-1691-4620-9690-65940fb033af"
}
```

You have now created a user in Amazon Cognito with the username `student` and a password of `student`. Even though this isn't secure, you are using a simple username and password for simplicity during testing.

23. **Command:** Next, confirm the user that you created with the command below:

```
aws cognito-idp admin-confirm-sign-up --user-pool-id $CognitoPoolId --username student
```

Expected output:

None, unless there is an error.

24. From the **AWS Management Console**, use the **AWS search bar** to search for and choose **Cognito**.

25. Choose the hamburger menu icon ☰ in the top left corner and then choose **User pools**.

26. Choose the PollyNotesPool link.

27. Under the **Users** tab you should see one new user named student with a confirmation status of **Confirmed**.

Note: If you do not see it, choose the refresh button to refresh the browser.

Congratulations! You have successfully created an Amazon Cognito User Pool, manually added a user, and confirmed the user.

TASK 1.3: TEST THE USER AUTHENTICATION IN A SIMPLE WEB PAGE

A test login page has been created for you to confirm the Amazon Cognito user and pool.

28. To the left of these instructions, copy the **testLoginWebsite** value, open that URL in a new web browser tab.

This page, takes the values that you copied earlier and the username and password **student**. As an example of how the login process works, there is JavaScript code on the right, which uses the Amazon Cognito JavaScript SDK. As you enter values in the login form, you will see where those values are used in this sample code.

29. Enter, the **User Pool Id** and **App Client Id** in the first two input fields.

Note: This information is used to create the **poolData** variable.

30. Enter `stud...` in both the **Username** and **Password** input fields.

Note: These values are used in the **CognitoUser** and **AuthenticationDetails** objects.

31. Choose **Login**.

Using the **.getIdToken().getJwtToken()** methods on the return of successful authentication, a **JSON Web Token** or **JwtToken** is retrieved. JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties. You can decode the token at JWT.IO if you would like to verify the claim.

When you make API calls that require authentication you have to pass this Authorization token in a header with the request.

32. Leave this window open, you will use this token in the next step for a test.

Task 2: Configure Amazon API Gateway to use Amazon Cognito as an authorizer

Now that you have created your Amazon Cognito user pool, you can configure an Amazon API Gateway authorizer to secure your methods with. After the API is deployed, the client must first sign the user in to the user pool, obtain an identity or access token for the user. The client then calls the API method with one of the tokens, which are typically set to the request's Authorization header. The API call succeeds only if the required token is supplied and the supplied token is valid.

High-Level Instructions

- Create an Amazon API Gateway authorizer `PollyNotesPool` for your existing API **PollyNotesAPI** that expects `Authorization` as the token source
- Configure the existing **/notes** GET AND POST methods to use the new Authorizer
- Update the mapping templates to transform the request using the Amazon Cognito username (`"$context.authorizer.claims['cognito:username...`) as the UserId

Detailed Instructions

TASK 2.1: CREATE AN AMAZON API GATEWAY AUTHORIZER FOR THE POLLYNOTESAPI USING THE CONSOLE

In this task, you create the Amazon Cognito authorizer. The authorizer will allow you to control access to the API with the Amazon Cognito User Pool you created in the last task.

33. From the **AWS Management Console**, use the **AWS search bar** to search for `API Gateway` and then choose the service from the list of results.

Note: This lab uses the **latest** API Gateway interface. If you find a dialog prompting you to switch to the new console, choose **Try out the new console**.

34. Choose the name link for **PollyNotesAPI**.

35. In the left navigation menu, choose **Authorizers**.

36. Choose **Create authorizer**.

37. Enter the following information:

- **Authorizer Name**: `PollyNotesPool`
- **Type**: `Cognito`
- **Cognito User Pool**: `PollyNotesPool`
- **Token source**: `Authorization`
- **Token Validation**: Leave empty

38. Choose **Create authorizer**.

39. Choose **PollyNotesPool**.

40. Under **Test authorizer** section, in the **Token value** input box, paste the **Authorization Token** generated in the last task.

41. Choose **Test authorizer**.

The Response contains the claim information that was returned from Amazon Cognito that you can use in your application. You will be using the **cognito:username** value in future steps.

TASK 2.2: CONFIGURE THE EXISTING METHODS TO USE THE NEW AUTHORIZER

The next few procedures walk you through the steps to secure the two existing API methods with the authorizer. You will also replace the static UserId mapping template with dynamic information from the authorization claim.

42. In the left navigation menu, choose **Resources**.

43. Refresh your browser to refresh the list of authorizers for the next set of steps.

44. In the **Resources** column, under **/notes**, choose **GET**.

45. Choose the **Method Request** tab.

46. Choose **Edit** .

47. For **Authorization** , choose **PollyNotesPool** under **Cognito user pool authorizers** .

CAUTION: If you don't see **PollyNotesPool** , refresh your browser window.

48. Choose **Save** .

49. Choose the **Integration request** tab.

50. Choose **Edit** .

51. Expand the **Mapping templates** section.

This mapping template was previously hard coded with the UserId **student** to work with the Lambda functions. Now that the users will be authenticated you can pull this dynamically from the request.

52. For the **Template body** , copy and paste the code below.

```
{
    "UserId": "$context.authorizer.claims['cognito:username']"
}
```

53. Choose **Save** .

Add the authorizer to the POST method as well.

54. In the **Resources** column, under **/notes** , choose **POST** .

55. Choose the **Method request** tab.

56. Choose **Edit** .

57. For **Authorization** , choose **PollyNotesPool** under **Cognito user pool authorizers** .

58. Choose **Save** .

59. Choose the **Integration request** tab.

60. Choose **Edit** .

61. Expand the **Mapping templates** section.

Change the mapping template to pull dynamically from the request. Notice that the NoteId and note is also passed to the Lambda function for this method.

62. For the **Template body** , copy and paste the code below.

```
{
    "UserId": "$context.authorizer.claims['cognito:username']",
    "NoteId": $input.json('$.NoteId'),
    "Note": $input.json('$.Note')
}
```

63. Choose **Save** .

Congratulations! The API is now secured. You will deploy this configuration in the next task.

Task 3: Create the remaining API resources using a Swagger file

In this task, you will create all the remaining resources and methods for the application. Your Lambda functions were already created in previous labs and the code will not need to be modified.

API gateway allows you to use the Swagger specification to export and import configuration information. This is a much quicker method than using the AWS Management Console.

High-Level Instructions

- Review the Swagger file. `~/environment/api/PollyNotesAPI-swagger.yaml` It includes all the remaining API resource definitions needed for your application
- Replace the placeholders in the Swagger file to customize it for your lab
- Merge the Swagger file to create Amazon API Gateway resources
- Update the AWS Lambda function permissions for the newly created API resources
- Verify that the resources imported are correct using the Amazon API Gateway console

Detailed Instructions

TASK 3.1: CUSTOMIZE THE SWAGGER FILE FOR YOUR LAB

Before you can use this file you have to replace three placeholders to customize it. The three variables are below.

- [Cognito_Pool_Arn]
- [AWS_Region]
- [AWS_AccountId]

Instead of replacing them manually you can use a script to do it. This is useful if you want to deploy your API in multiple regions or AWS accounts.

64. **Command:** In the **"AWS Cloud9"** web browser tab, run the commands below in the terminal to replace the variables for your lab:

```
region=$(curl http://169.254.169.254/latest/meta-data/placement/region -s)

acct=$(aws sts get-caller-identity --output text --query "Account")

poolId=$(aws cognito-idp list-user-pools --max-results 1 --output text --query "UserPools[].Id")

poolArn="arn:aws:cognito-idp:$region:$acct:userpool/$poolId"
```

Expected output:

None, unless there is an error.

65. Run the commands below to update the swagger file place holders with the variables you just created:

```
sed -i "s~\[Cognito_Pool_ARN\]~$poolArn~g" ~/environment/api/PollyNotesAPI-swagger.yaml

sed -i "s~\[AWS_Region\]~$region~g" ~/environment/api/PollyNotesAPI-swagger.yaml

sed -i "s~\[AWS_AccountId\]~$acct~g" ~/environment/api/PollyNotesAPI-swagger.yaml
```

Expected output:

None unless there are errors.

Now that the Swagger file has been customized to work with your lab, explore the contents.

66. Expand the api folder and open **PollyNotesAPI-swagger.yaml** .

Swagger allows you to describe the structure of your APIs. This file is readable by both humans and machines, so it is used as a way to document your API as well as a way to back up your configuration. The file that you will use has all of the remaining resources and methods defined that your application needs.

- /notes/search
  - GET
  - OPTIONS
- /notes/{id}
  - POST
  - DELETE
  - OPTIONS

The Swagger file also has the Amazon Cognito authorizer defined so that it can be referenced to define the security of these methods.

Notice the sections that start with **x-amazon-apigateway-** ; these are API Gateway extensions that allow you to add Amazon specific configuration. In this case, you are specifying the Lambda function uri along with request templates and response configuration.

TASK 3.2: MERGE THE SWAGGER FILE TO THE EXISTING API

You can merge the Swagger file to your API with the console or AWS CLI. Use the AWS CLI for this lab. You have to put the new resources and then create a new deployment to make them available. Notice how the mode "merge" is used. This mode will not remove any resources that are not in the Swagger file. This leaves the existing **/note** methods in place while adding the new ones. The authorizer which is existing and defined in the Swagger file will be updated if there are any differences.

67. ⌨ **Command:** Run the command below change directories into ~/environment/api directory :

```
cd ~/environment/api
```

Expected output:

None, unless there is an error.

68. ⌨ **Command:** Run the command below to set the **apiId** variable to the PollyNotesAPI ID:

```
apiId=$(aws apigateway get-rest-apis --query "items[?name == 'PollyNotesAPI'].id" --output text)
```

Expected output:

None, unless there is an error.

69. ⌨ **Command:** Run the command below to import the new resources to API Gateway:

```
aws apigateway put-rest-api --rest-api-id $apiId --mode merge --body 'fileb://PollyNotesAPI-swagger.yaml'
```

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

{
    "id": "zt77702fy8",
    "name": "PollyNotesAPI",
    "createdDate": "2022-03-29T16:45:17+00:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ]
    },
    "tags": {
        "aws:cloudformation:logical-id": "pollyNotesAPI",
        "aws:cloudformation:stack-id": "arn:aws:cloudformation:ap-southeast-1:710606863198:stack/LabStack-a6qh92vr7LcXDF2CKQ8VhF-0/94792f00-af7f-11ec-897f-06c217005cc0",
        "aws:cloudformation:stack-name": "LabStack-a6qh92vr7LcXDF2CKQ8VhF-0"
    },
    "disableExecuteApiEndpoint": false
}
```

Note: To escape from the output, you may need to type **Ctrl+C** or **q** depending on your OS.

70. ⌨ **Command:** Run the command below to deploy the new resources to API Gateway:

```
aws apigateway create-deployment --rest-api-id $apiId --stage-name Prod
```

Expected output:

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

{
    "id": "w67bc0",
    "createdDate": "2022-03-29T20:24:14+00:00"
}
```

If you were to add a method manually in the console, as you did in the last lab, the Lambda permissions would be configured for you. Since you added the functions by importing a Swagger file, you now need to grant the API Gateway service access to invoke the three Lambda functions you imported methods for.

71. ⌨ **Command:** Run the commands below to **add Lambda permissions** to the functions:

```
aws lambda add-permission --function-name delete-function --statement-id apiInvoke --action lambda:InvokeFunction --principal apigateway.amazonaws.com

aws lambda add-permission --function-name dictate-function --statement-id apiInvoke --action lambda:InvokeFunction --principal apigateway.amazonaws.com

aws lambda add-permission --function-name search-function --statement-id apiInvoke --action lambda:InvokeFunction --principal apigateway.amazonaws.com
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

{
    "Statement": "{\"Sid\":\"apiInvoke\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"arn:aws
}

{
    "Statement": "{\"Sid\":\"apiInvoke\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"arn:aws
}

{
    "Statement": "{\"Sid\":\"apiInvoke\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":\"arn:aws
}
```

72. In the **"API Gateway"** web browser tab, in the left navigation pane, choose resources.

Explore the new resources created by the Swagger import. Everything is available and configured to use the authorizer if appropriate. Using Swagger files is a great way to back up and document your configuration. Regularly exporting the Swagger file will allow you to have a consistent point in time snapshot of your configuration. If you need to expand your deployment to another AWS Region or Account, you can easily import the latest wagger file in those locations. Regularly import the latest file to maintain consistency between deployments. Just be aware that you will need to update the three unique values (AWS_Region, AWS_AccountId, and Cognito_Pool_ARN) that you did here to localize the file.

Task 4: Configure the frontend web application

Now that the API has been configured and secured, you can deploy the web front end. This web application is serverless and hosted on Amazon S3. All processing is performed by the client's web browser. The framework that it was written in requires you to build the application to optimize it for web distribution. Before you build and deploy, you need to customize the configuration to use your API.

High-Level Instructions

- Update your application `web/src/Config.js` file to refer to the Amazon Cognito user pool and API.
- Build the web application and download dependencies with npm.
- Update the Amazon S3 website bucket with the latest application files.

73. **Command:** Run the command below to create a bucket variable. Be sure to replace *WEBBUCKETNAME* with the **WebBucketName** value from the left side of the lab page:

```
webBucket=WEBBUCKETNAME
```

Detailed Instructions

TASK 4.1: CONFIGURE THE WEB APPLICATION FOR YOUR LAB

For portability, all of the configuration settings that you need to modify are located in the **Config.js** file. After you update the configuration, you will copy the files to Amazon S3. The Amazon S3 bucket has already been configured to host your website.

74. Run the commands below to update the **Config.js** file place holders with the variables you created earlier:

```
sed -i "s~\[UserPoolId\]~$CognitoPoolId~g" ~/environment/web/src/Config.js

sed -i "s~\[AppClientId\]~$AppClientId~g" ~/environment/web/src/Config.js

sed -i "s~\[ApiURL\]~$apiURL~g" ~/environment/web/src/Config.js
```

Expected output:

None, unless there is an error.

75. Expand **web/src** and open **Config.js** to ensure the updates were made. If not, update the values then save and close the **Config.js** .

76. **Command:** Run the command below to install change into the `~/environment/web` directory:

```
cd ~/environment/web
```

Expected output:

None, unless there is an error.

77. **Command:** Run the command below to install dependencies:

```
npm install
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************

added 1993 packages, and audited 1994 packages in 3m

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

35 vulnerabilities (18 moderate, 16 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

The `npm install` command downloaded any dependencies that the web application needs for build or production.

78. **Command:** Run the commands below to build the web application. While it is building scroll down to read more about these steps.

```
npm run test+build
```

Expected output:

```
******************************
```

```
**** This is OUTPUT ONLY. ****
*******************************

> react-web@0.1.0 test+build
> CI=true npm test && npm run build


> react-web@0.1.0 test
> react-scripts test

PASS src/App.test.js
  ✓ renders App component (147 ms)

  console.log
    undefined

      at src/Routes/Notes.js:33:21

  console.log
    redirecting to login

      at src/Routes/Notes.js:34:21

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.613 s
Ran all test suites.

> react-web@0.1.0 build
> react-scripts build

Creating an optimized production build...
Browserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db

Why you should do it regularly:
https://github.com/browserslist/browserslist#browsers-data-updating
Compiled successfully.

File sizes after gzip:

  88.59 KB  build/static/js/2.16e7aa1e.chunk.js
  23.25 KB  build/static/css/2.4c97ca4f.chunk.css
  4.64 KB   build/static/js/main.98523804.chunk.js
  775 B     build/static/js/runtime-main.7e0376b9.js
  277 B     build/static/css/main.c56fd950.chunk.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
```

Note: If you did not replace the variables in the Config.js file, you will see and error that states **Invalid UserPoolId format** and the test will fail. If the test does not pass the application will not be built.

The `npm run test+build` command runs unit tests to make sure the application builds correctly. If the tests are successful, it runs scripts that optimizes the web application for distribution and places the files for deployment into the build folder.

79. **Command:** Run the command below to create a variable for website bucket and then copy the files to Amazon S3 for hosting. Be sure to replace *WEBBUCKETNAME* with the **WebBucketName** value from the left side of the lab page:

```
webBucket=WEBBUCKETNAME
```

Expected output:

None, unless there is an error.

80. **Command:** Run the command below to copy the files to Amazon S3 for hosting:

```
aws s3 sync --delete build/. s3://$webBucket
```

Expected output:

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

upload: build/asset-manifest.json to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/asset-manifest.json
upload: build/index.html to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/index.html
upload: build/manifest.json to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/manifest.json
upload: build/robots.txt to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/robots.txt
upload: build/favicon.ico to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/favicon.ico
upload: build/static/css/main.c56fd950.chunk.css to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/css/main.c56fd950.chunk.css
upload: build/static/css/main.c56fd950.chunk.css.map to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/css/main.c56fd950.chunk.css.map
upload: build/static/js/runtime-main.7e0376b9.js.map to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/runtime-main.7e0376b9.js.map
upload: build/static/js/2.16e7aa1e.chunk.js.LICENSE.txt to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/2.16e7aa1e.chunk.js.LICENSE.txt
upload: build/static/js/main.98523804.chunk.js to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/main.98523804.chunk.js
upload: build/static/js/runtime-main.7e0376b9.js to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/runtime-main.7e0376b9.js
upload: build/static/css/2.4c97ca4f.chunk.css to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/css/2.4c97ca4f.chunk.css
upload: build/static/js/main.98523804.chunk.js.map to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/main.98523804.chunk.js.map
upload: build/static/js/2.16e7aa1e.chunk.js to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/2.16e7aa1e.chunk.js
upload: build/static/css/2.4c97ca4f.chunk.css.map to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/css/2.4c97ca4f.chunk.css.map
upload: build/static/js/2.16e7aa1e.chunk.js.map to s3://labstack-a6qh92vr7lcxdf2ck-pollynotesweb-1g5f9zg6rek5g/static/js/2.16e7aa1e.chunk.js.map
```

The `aws s3 sync --delete` command was used here to recursively update files from the source build directory the Amazon S3 bucket and remove any files that are no longer in the source directory. A local file will require uploading if the size of the local file is different than the size of the s3 object, the last modified time of the local file is newer than the last modified time of the s3 object, or the local file does not exist under the specified bucket and prefix.

Task 5: Test the web application functionality

Now that your application has been fully deployed, it's time to test it.

81. **Copy/Paste:** Paste the **PollyNotesWebsite** URL, from the left of these instructions, to a new web browser tab and test the deployed website.

82. For **Username** and **Password**, enter `stud`

83. In the **Note** text box, enter a new note and choose **Add** .

84. Choose the edit button beside one of the notes.

85. In the Note text box, edit the note text and choose **Update** .

86. Enter text into the **Search Notes** box to confirm search functionality.

87. Press the speak button to convert the note to speech and listen to the notes.

88. Choose the delete button to delete the note.

Summary

Congratulations! You have successfully deployed a serverless application using Amazon S3, Amazon API Gateway, Amazon Cognito, AWS Lambda, and Amazon DynamoDB.

You can now:

- Create a user pool and an app client for your web application using Amazon Cognito.
- Add new users and confirm their ability to sign-in using the Amazon Cognito CLI.
- Configure API Gateway methods to use Amazon Cognito as an authorizer.
- Verify JWT authentication tokens are generated during API calls.
- Develop API Gateway resources rapidly using Swagger importing strategy.
- Set up your web application frontend to use Amazon Cognito and API Gateway configurations and verify the entire application functionality.

End lab

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

89. Return to the **AWS Management Console** .

90. At the upper-right corner of the page, choose **AWSLabsUser** , and then choose **Sign out** .

91. Choose **End lab** and then confirm that you want to end your lab. For more information about AWS Training and Certification, see *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* .

Additional Resources

- Control access to a REST API using Amazon Cognito user pools as authorizer

---

Appendix

AWS SERVICES NOT USED IN THIS LAB

AWS services that are not used in this lab are deactivated in the lab environment. In addition, the capabilities of the services used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. While not all of the icons will be used, 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 note, tip, or important guidance.
- **Additional information:** Where to find more information.
- **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).
- **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).
- **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.
- **Copy/Paste:** A code block that displays the contents of a script or file you need to copy and paste that has been pre-created for you. When you need to copy only a certain part of a code block, there will be numbered **TODO** comments in the code.
- **Knowledge check:** An opportunity to check your knowledge and test what you have learned.
- **Security:** An opportunity to incorporate security best practices.
- **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
- **Copy command:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
- **Hint:** A hint to a question or challenge.
- **Answer:** An answer to a question or challenge.
- **Group effort:** A time when you must work together with another student to complete a task.

Return to the instructions