[version_1.0]

Note

The exercises in this course will have an associated charge in your AWS account. In this exercise, you will create the following resources:

- AWS Cloud9 environment
- Amazon Simple Storage Service (Amazon S3) bucket
- Amazon API Gateway REST API
- Amazon DynamoDB table
- AWS Step Functions state machine
- AWS Lambda functions

The final exercise includes instructions to delete all the resources that you create in the exercises.

Familiarize yourself with **AWS Cloud9 Pricing**, **Amazon S3 pricing**, **Amazon API Gateway pricing**, **Amazon DynamoDB pricing**, **AWS Step Functions pricing**, **AWS Lambda Pricing**, and the **AWS Free Tier**.

Exercise: Setting Up Your Development Environment

In a production environment, you would normally create a customer managed policy in AWS Identity and Access Management (IAM). Customer managed policies provide more precise control over your policies than policies that AWS manages. This policy would then have permissions that are specific to the AWS resources you need. You would also attach this policy to a new user and log in to the AWS Management Console with that new user. However, because you are working in an exercise environment, those steps were omitted.

In this exercise, you will start by creating an AWS Cloud9 environment for your development environment. In this environment, you will download and extract the source code that you will use in this course. You will deploy the application's backend infrastructure by using the AWS Serverless Application Model Command Line Interface (AWS SAM CLI). You will use AWS SAM to create all the resources that host the backend of the application: an Amazon API Gateway gateway, AWS Lambda functions, an Amazon DynamoDB table, and an AWS Step Functions state machine. You will also deploy the application frontend: a React web application that's hosted on an S3 bucket, which acts as the web server. After the application is up and running, you will put the application under source control by using AWS CodeCommit.

Task 1: Creating an AWS Cloud9 environment and downloading the application

For this task, you create an AWS Cloud9 environment for use as your development environment. You also download the pre-built application on to the AWS Cloud9 instance.

1. Choose **Services** and search for **Cloud9**. *Make sure you are in the Oregon **(US-West-2)** Region*.

2. Choose **Create environment**.

3. For **Name**, enter `trivia-app`, and choose **Next step**.

4. Keep the default **Environment settings** and choose **Next step**.

5. Choose **Create environment**.

6. Download the application through the **AWS Cloud9** terminal by running the following command:

   ```
   wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/DEV-AWS-MO-DevOps-C1/dow
   unzip -o ~/trivia-app.zip
   ```

Task 2: Creating the backend infrastructure

In this task, you deploy the application stack by using the AWS SAM and the AWS CLI. You also make a few minor configuration changes.

1. In the AWS Cloud9 terminal, change the directory to the `trivia-app` folder and use AWS SAM to build the `template.yaml` file.

   ```
   cd trivia-app
   sam build
   sam deploy --guided
   ```

   AWS SAM settings: For the **Stack Name**, make sure to copy `trivia-app` and paste it. However, for the other entries, accept the default selections by pressing Enter or Return.

   ```
   Stack Name [sam-app]: trivia-app
   AWS Region [us-west-2]: <Press Enter or Return>
   #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
   Confirm changes before deploy [y/N]: <Press Enter or Return>
   #SAM needs permission to be able to create roles to connect to the resources in your
   Allow SAM CLI IAM role creation [Y/n]: <Press Enter or Return>
   Save arguments to configuration file [Y/n]: <Press Enter or Return>
   SAM configuration file [samconfig.toml]: <Press Enter or Return>
   SAM configuration environment [default]: <Press Enter or Return>
   ```

   You should get confirmation that the stack was created successfully.

   ```
   Successfully created/updated stack - trivia-app in us-west-2
   ```

2. From the **Outputs** table in the terminal output, copy the Websocket **Value**. It should look similar to the following example:

```
wss://xxxxxxxxxx.execute-api.us-west-2.amazonaws.com/Prod
```

Task 3: Testing the frontend on AWS Cloud9

In this task, you install Node package manager (NPM) dependencies. You also preview the application in the AWS Cloud9 environment.

1. Update the `trivia-app/front-end-react/src/config.js` file with the Websocket endpoint that you copied previously. Make sure to save the file.

2. In **AWS Cloud9**, set the Node version to the v16 (codename: Gallium).

   ```
   nvm install lts/gallium
   nvm alias default lts/gallium
   ```

3. Change to the `front-end-react` directory and install the NPM dependencies.

   ```
   cd front-end-react/
   npm install
   npm run start
   ```

   Note: There is currently a known issue with the dependency webpack under Node v17. If you see an error containing `code: 'ERR_OSSL_EVP_UNSUPPORTED'` ensure you are running the exercise with Node v16.

   After the NPM installation finishes, you might see NPM warnings, but it's OK to proceed.

4. In the AWS Cloud9 menu bar, choose **Preview** and choose **Preview Running Application**. Feel free to play around with the game.

5. Finally, in the **AWS Cloud9** terminal, close the previewed application and end the running NPM server by sending a SIGINT (Ctrl+C).

Task 4: Deploying the frontend to an S3 bucket

For this task, you deploy the application to a new S3 bucket, which will act as the web server.

1. Create a uniquely named Amazon Simple Storage Service (Amazon S3) bucket. For the bucket name, you could use your initials, a set of numbers, and the string *trivia-app-bucket*, similar to this example: `<your_initials><numbers>-trivia-app-bucket`.

   ```
   aws s3 mb s3://<your_initials><numbers>-trivia-app-bucket/
   npm run build
   aws s3 sync --acl public-read build s3://<your_initials><numbers>-trivia-app-bucket/
   ```

   Note: If the bucket name you entered already exists, change the numbers and try to create the bucket again.

2. View the application that's hosted on your bucket by using the application URL, which includes the bucket name and *index.html*. The URL should look similar to the following

example: `https://<your_initials><numbers>-trivia-app-bucket.s3.amazonaws.com/index.html`.

Task 5: Putting the application under source control

In this final task, you add source control to the application by using AWS CodeCommit to set up a new repository. You then use Git within AWS Cloud9 to commit the code changes to the repository.

1. At the left on the menu bar, choose **AWS Cloud9** and choose **Go To Your Dashboard**.

2. Choose **Services** and search for **CodeCommit**.

3. Choose **Create repository**.

4. For the **Repository name**, enter `trivia-app` and then choose **Create**.

5. Switch back to the **AWS Cloud9** terminal window.

6. Configure Git with your name and email address.

```
git config --global user.name "<REPLACE_WITH_YOUR_NAME>"
git config --global user.email <REPLACE_WITH_YOUR_EMAIL>
```

7. In the terminal, initialize your repository, create a branch and a commit, and push the code to the repository.

```
cd ~/environment/trivia-app/
git init
git checkout -b main
git add .
git commit -m "initial commit"
git remote add origin codecommit://trivia-app
git push origin main
```

The following list provides explanations of each Git command:

- `git init` - Initialize a Git repository.
- `git checkout -b main` - Create a branch named `main`, and check out (switch to) the branch.
- `git add` - Adds all the files to the Git index. The files are now staged for a commit.
- `git commit` - Creates a commit in the Git log with a commit message.
- `git remote add origin` - Create a remote named `origin`. A remote is how Git tracks the *remote* repository that is hosted in CodeCommit. The `codecommit://` prefix tells Git to use the CodeCommit remote helper.
- `git push` - Update the remote named `origin` with the `main` branch.