[*Version 1.0.39*]

Lab 13.1: Automating Code Deployments with a CI/CD Pipeline

Lab overview and objectives

In this lab, you will create an AWS CodeCommit repository (also known as a repo) and an AWS CodePipeline pipeline. You will configure the pipeline to automatically apply updates to the café website as changes are saved to the repository.

After completing this lab, you should be able to:

- Create a new CodeCommit repository
- Clone and update a CodeCommit repository
- Create a pipeline by using CodePipeline

Duration

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

AWS service restrictions

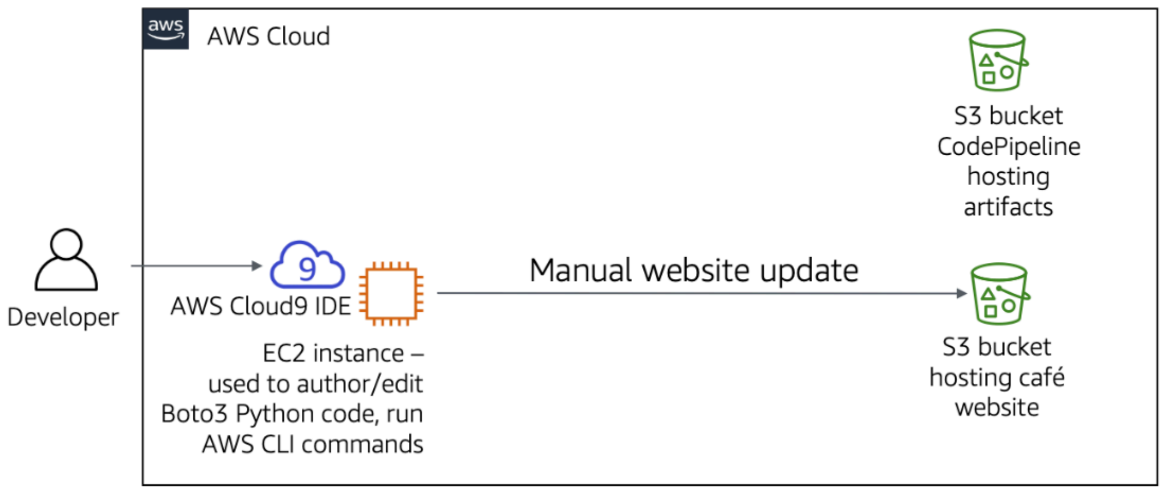
In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

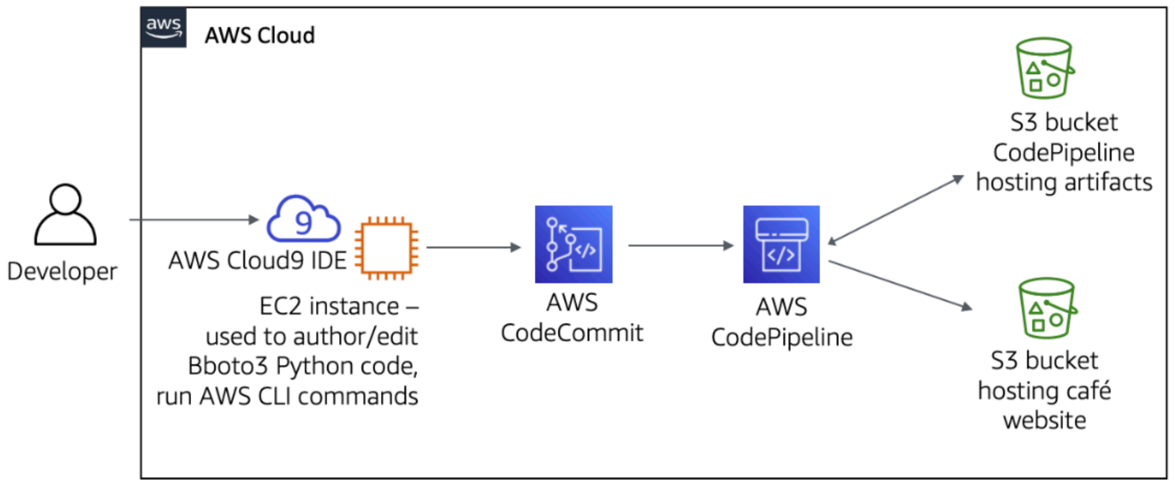
Now that the café website is in production, Frank wants a reliable process in place to track code changes and update the site when changes are made. He asked Sofía to find a way to centralize the website code and add version control. He has also asked if it's possible to automatically update the website instead of manually running scripts and uploading files when changes are made.

Mateo, a café regular and AWS consultant who specializes in automating repeatable processes, was chatting with Sofía about his work. He mentioned using CodeCommit to collaborate on projects with other developers. Sofía shared that she was researching CodeCommit to centralize the café website's code and asked Mateo for suggestions about automating updates to the site. Mateo suggested using CodePipeline because it easily integrates with both CodeCommit and Amazon Simple Storage Service (Amazon S3).

When you *start* the lab, many services are deployed for you. In this lab, you will focus on the services that are represented in the following diagram. As in past labs, the AWS Cloud9 instance is used as the development environment. The developer runs commands from AWS Cloud9 to update the code in the S3 bucket that hosts the website. The second bucket in the diagram will be used to set up a continuous integration and continuous delivery (CI/CD) pipeline.


AWS Cloud
S3 bucket CodePipeline hosting artifacts
Developer
AWS Cloud9 IDE
Manual website update
S3 bucket hosting café website
EC2 instance – used to author/edit Boto3 Python code, run AWS CLI commands

By the *end* of this lab, you will have created the architecture in the following diagram. You will have created a CodeCommit repository to store the website code. You will have also created a CodePipeline pipeline to automate updates to the website when changes are pushed to the CodeCommit repository. Artifacts that CodePipeline uses to deploy and update your website application will be hosted on an S3 bucket that is separate from the bucket that hosts the café website.


AWS Cloud
S3 bucket CodePipeline hosting artifacts
Developer
AWS Cloud9 IDE
AWS CodeCommit
AWS CodePipeline
S3 bucket hosting café website
EC2 instance – used to author/edit Bboto3 Python code, run AWS CLI commands

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.

   A **Start Lab** panel opens, and it displays the lab status.

   Tip: If you need more time to complete the lab, choose the **Start Lab** button again to restart the timer for the environment.

2. Wait until you see the message *Lab status: ready*, then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose AWS.

   This opens the AWS Management Console in a new browser tab. The system will automatically log you in.

   Tip: If a new browser tab does not open, a banner or icon is usually at the top of your browser with a message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon and then choose **Allow pop ups**.

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time so that you can follow the lab steps more easily.

   Tip: If you want the lab instructions to display across the entire browser window, you can hide the terminal in the browser panel. In the top-right area, clear the **Terminal** ☐ check box.

Task 1: Preparing the development environment

In this first task, you will configure your AWS Cloud9 environment so that you can use Python and the AWS Command Line Interface (AWS CLI) to interact with AWS services.

5. Before proceeding to the next step, verify that the AWS CloudFormation stack creation process for the lab has successfully completed.
   - In a new browser tab, navigate to the CloudFormation console.
   - In the left navigation pane, choose **Stacks**.
   - For *all three* stacks, verify that the **Status** says *CREATE_COMPLETE*.

     ⚠️ If any stack doesn't yet have that status, wait until it does. It might take about 12 minutes to complete from the time that you chose **Start Lab**.

6. In the AWS Management Console, connect to the AWS Cloud9 integrated development environment (IDE) named *Cloud9 Instance*.

7. Download and extract the files that you need for this lab.
   - In the Cloud9 terminal (at the bottom of the IDE), run the following command:

     ```
     wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-
     ACCDEV-2-91558/13-lab-ci-cd/code.zip -P /home/ec2-user/environment
     ```

   - Notice that a **code.zip** file was downloaded to the AWS Cloud9 instance. The file is listed in the Environment window.
   - Extract the file:

```
unzip code.zip
```

8. Run a script to upgrade the version of Python and the AWS CLI that are installed on the Cloud9 instance. The script also re-creates the work that you completed in earlier labs into this AWS account.

   Note: This script deploys the architecture that you created across the previous labs. This includes populating and configuring the S3 bucket that the café website uses. The script creates the Amazon DynamoDB table and populates it with data. The script also re-creates the REST API that you created in the Amazon API Gateway lab and deploys the containerized coffee suppliers AWS Elastic Beanstalk application.

   - Set permissions on the script so that you can run it, and then run it:

     ```
     chmod +x ./resources/setup.sh && ./resources/setup.sh
     ```

   - When prompted for an IP address, enter the IPv4 address that the internet uses to contact your computer. You can find your IPv4 address at https://whatismyipaddress.com.

     Note: The IPv4 address that you set is the one that will be used in the bucket policy. Only requests that originate from this IPv4 address will be allowed to load the website pages. Do not set it to 0.0.0.0 because the S3 bucket's block public access settings will prevent access.

   - When prompted for an email address, enter one that you have access to as you complete the lab.

9. Verify the version of AWS CLI installed.

   - In the AWS Cloud9 terminal, run the following command:

     ```
     aws --version
     ```

     The output should indicate that version 2 is installed.

10. Verify that the SDK for Python is installed.

    - Run the following command:

      ```
      pip show boto3
      ```

      Note: If you see a message about not using the latest version of pip, ignore the message.

    Keep the AWS Cloud9 environment open in your browser. You will use it later in this lab.

Task 2: Creating a CodeCommit repository

When a developer manages code in their local working environment, it is difficult to track and manage changes. This approach also limits the ability for multiple developers to collaborate on the same codebase.

AWS CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit makes it easy for teams to securely collaborate on code with contributions encrypted in transit and at rest.

In this task, you will create a CodeCommit repository to host the café website code.

11. Create a CodeCommit repository to host the codebase.
    - On the menu bar in the AWS Cloud9 IDE, choose **AWS Cloud9**, and then choose **Go To Your Dashboard**.
    - Navigate to the CodeCommit console.
    - Choose **Create repository**.
    - Configure the following settings:
        - **Repository name:** Enter `front_end_website`
        - **Description:** Enter `Repository for the cafe website front end`
        - Keep the rest of the default settings.
    - Choose **Create**.

12. Create a test file.
    - In the **front_end_website** section, choose **Create file**.
    - Copy and paste the following code into the **front_end_website** text box:

```
<!DOCTYPE html>
<html>
    <head>
    <title>Test page</title>
    </head>
    <body>
        <h1>
           This is a sample HTML page.
        </h1>
    </body>
</html>
```

    - In the **Commit changes to main** section, configure the following settings:
        - **File name:** Enter `test.html`
        - **Author name:** Enter your name.
        - **Email address:** Enter the same email address that you used in Task 1.
        - **Commit message:** Enter `This is my first commit.`
        - Choose **Commit changes**.

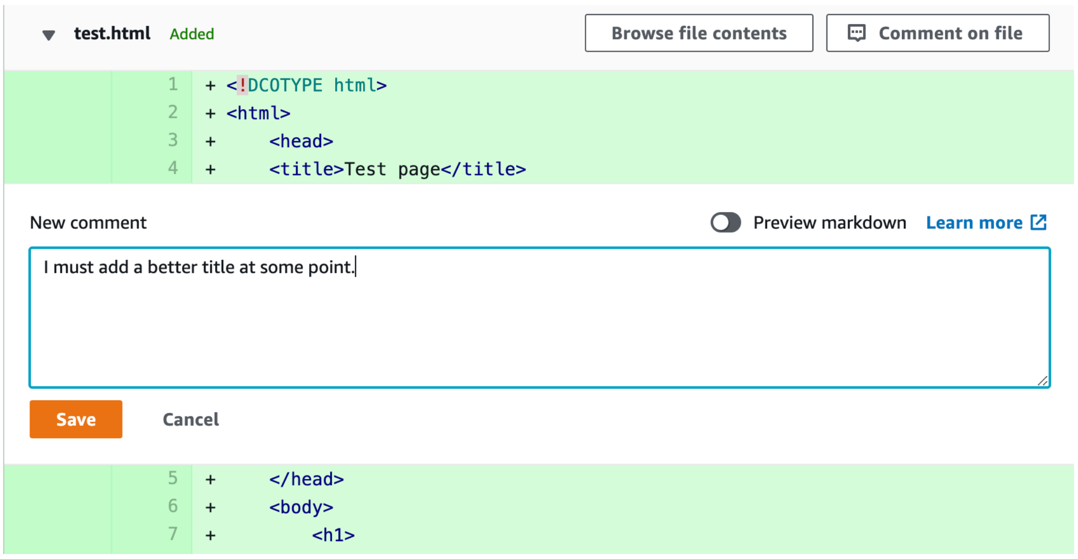
13. Review your commit.
    - In the left navigation pane, under **Repositories**, choose **Commits**.
    - Choose the link for the commit ID. Only one should be listed.
    - Review the information about this commit.

Note: On this page, you see the author of the commit, the commit message, the date of the commit, and the name and contents of the file that was added.

14. Add a comment to the committed file.
    - In the **test.html** section, hover on the plus sign (+) icon on line 4.
    - To the left of the line number, a comment icon appears. Choose the icon.

      Note: The comment icon is a small box that contains an ellipsis.
    - In the **New comment** text box, enter: `I must add a better title at some point.`

      The following image shows the comment text box.



```
1  + <!DCOTYPE html>
2  + <html>
3  +     <head>
4  +     <title>Test page</title>
```

I must add a better title at some point.

```
5  +     </head>
6  +     <body>
7  +         <h1>
```

    - Choose **Save**.

Now that you have created a CodeCommit repository to host and manage your code changes, you can use it as a source to automate publishing updates to the café website.

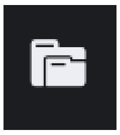
Task 3: Creating a pipeline to automate website updates

So far, you have been running commands from AWS Cloud9 to update the code in the S3 bucket for the website. In this task, you will configure a CI/CD pipeline by using CodePipeline to automate the website updates. The pipeline will use the code in your CodeCommit repository to deploy changes to the website's S3 bucket.

15. Return to the AWS Cloud9 IDE.

16. Choose the files icon, which is located in the upper-left corner.

    The icon looks like the following image:



17. Expand the **resources** folder, and open the file named **cafe_website_front_end_pipeline.json**.

    This file defines configuration that will be used to deploy your new pipeline. Review the following code snippets to understand how the pipeline is configured.

    The following lines declare the AWS Identity and Access Management (IAM) role that will be associated with the pipeline.

    ```
    "pipeline": {
      "roleArn": "arn:aws:iam::<FMI_1>:role/RoleForCodepipeline",
    ```

    The following code snippet defines the *Source* that your pipeline will use to create and update your application. In this case, the source is your CodeCommit repository, *front_end_website*. Note that the pipeline will be configured to use the *main* branch.

    ```
    {
       "name": "Source",
       "actions": [
        {
          "inputArtifacts": [],
          "name": "Source",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeCommit"
          },
          "outputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "configuration": {
            "RepositoryName": "front_end_website",
            "BranchName": "main"
          },
          "runOrder": 1
        }
      ]
    }
    ```

    The following section of code defines the *Deploy* stage. The deployment will update code in Amazon S3. The *configuration* settings define details about the deployment target. In this case, this section defines the S3 bucket name, configures files to be extracted from a .zip file, and sets a caching policy.

    ```
    {
        "name": "Deploy",
    ```

```
    "actions": [
        {
            "inputArtifacts": [
                {
                    "name": "MyApp"
                }
            ],
            "name": "CafeWebsite",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "version": "1",
                "provider": "S3"
        },
            "outputArtifacts": [],
            "configuration": {
                "BucketName": "<FMI_2>",
                "Extract": "true",
                "CacheControl": "max-age=14"
            },
            "runOrder": 1
        }
    ]
}
```

The final section of the code defines the *artifactStore*. This is the S3 bucket where CodePipeline artifacts will be stored.

```
    "artifactStore": {
        "type": "S3",
        "location": "codepipeline-us-east-1-<FMI_1>-website"
    },
    "name": "cafe_website_front_end_pipeline",
    "version": 1
}
```

18. Update the cafe_website_front_end_pipeline.json file:
    - Replace the two *<FMI_1>* placeholders with the your AWS account ID.

      Note: To find your account ID, run the following command: `aws sts get-caller-identity`
    - Replace the *<FMI_2>* placeholder with the name of the bucket that has *s3bucket* in the name.

      Note: To retrieve a list of the S3 buckets in your account, run the following command: `aws s3 ls`
    - Save the changes to the file.

19. To create the pipeline, run the following commands.

```
cd ~/environment/resources
aws codepipeline create-pipeline --cli-input-json
file://cafe_website_front_end_pipeline.json
```

The command returns output similar to the following example:

```
{
    "pipeline": {
    "name": "cafe_website_front_end_pipeline",
        "roleArn": "arn:aws:iam::111122223333:role/RoleForCodepipeline",
        "artifactStore": {
            "type": "S3",
            "location": "codepipeline-us-east-1-111122223333-website"
        },
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "CodeCommit",
                            "version": "1"
...
```

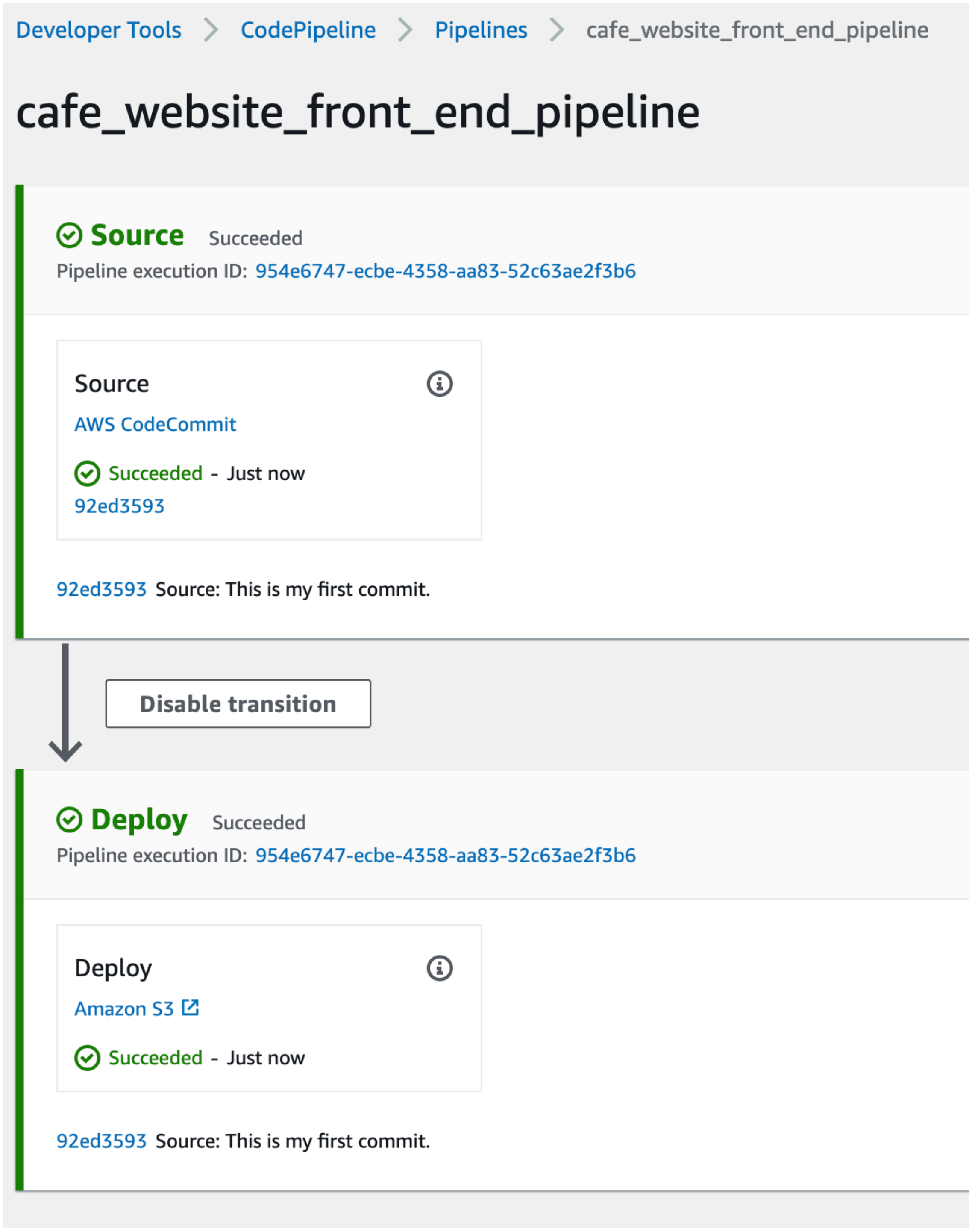
You may need to press the Q key to return to the command prompt.

If you receive an error, double check your account ID and the bucket name that you used to update the placeholders. Correct any typos in the cafe_website_front_end_pipeline.json file and run the command again.

20. Navigate to the CodePipeline console.

    The **Pipelines** section lists the **cafe_website_front_end_pipeline.json** file.

21. Choose the **cafe_website_front_end_pipeline** hyperlink and review the pipeline status, as shown in the following image.


Developer Tools > CodePipeline > Pipelines > cafe_website_front_end_pipeline

cafe_website_front_end_pipeline

Source Succeeded
Pipeline execution ID: 954e6747-ecbe-4358-aa83-52c63ae2f3b6

Source
AWS CodeCommit
Succeeded - Just now
92ed3593

92ed3593 Source: This is my first commit.

Deploy Succeeded
Pipeline execution ID: 954e6747-ecbe-4358-aa83-52c63ae2f3b6

Deploy
Amazon S3
Succeeded - Just now

92ed3593 Source: This is my first commit.

The pipeline should deploy successfully. Code was deployed using CodeCommit as the source and the café website S3 bucket as the target. This means the bucket should have been updated with the *test.html* file.

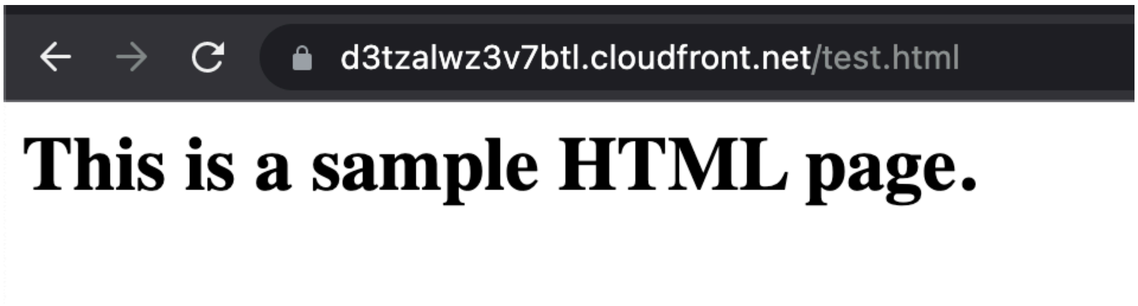
22. Verify the automated deployment.
    - Return to the AWS Cloud9 terminal.
    - To find your Amazon CloudFront distribution domain name, run the following command:

    ```
    aws cloudfront list-distributions --query
    DistributionList.Items[0].DomainName --output text
    ```

- Update the following URL by replacing <*cloudfront_domain*> with the value that was returned by the previous command: `https://<cloudfront_domain>/test.html`

  The updated URL is similar to *https://aaabbb111222.cloudfront.net/test.html*.
- Open a new browser tab, and enter the URL that you just created.

  You reach a sample webpage similar to the following:


d3tzalwz3v7btl.cloudfront.net/test.html

This is a sample HTML page.

- Keep this browser tab open. You will return to it later.

Well done! Now, when you update the repository, the pipeline will automatically update your website.

Next, you will clone the repository to your AWS Cloud9 instance. This will provide the ability to edit the files locally and synchronize with your centralized CodeCommit repository.

Task 4: Cloning a repository in AWS Cloud9

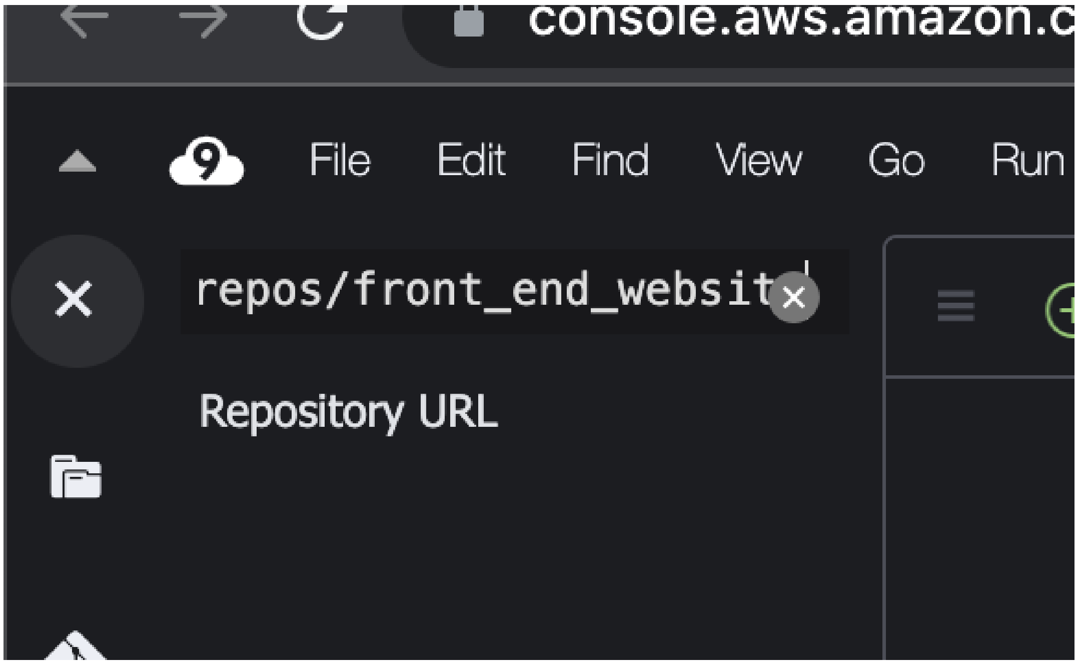
It's more efficient to edit code in an IDE than it is to use the CodeCommit console. In this task, you will clone a local copy of the repository in your AWS Cloud9 IDE work environment.

23. Retrieve the SSH clone URL for your repository.
    - Navigate to the CodeCommit console.
    - In the navigation pane, choose **Repositories**.
    - In the **Clone URL** column, choose **SSH** to copy the URL to your clipboard.

24. Clone your repository using the SSH URL.
    - Return to the AWS Cloud9 IDE.
    - Choose the Git source control icon, which is located on the left side of the page.

      The icon looks like the following image:

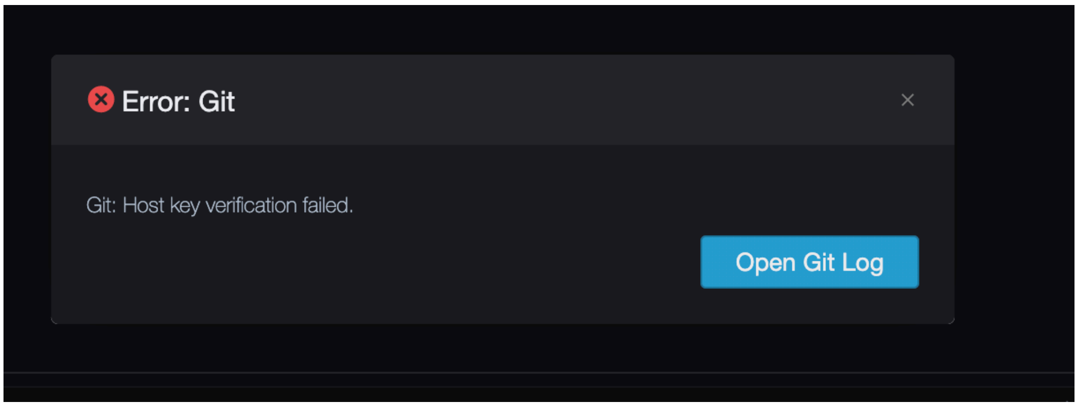
    - Choose **Clone Repository**.
    - You are prompted for a **Repository URL**. Paste your URL into the text box as shown in the following image:



- Press Enter.

  Note: You *might* be asked which directory to use. Use the root (environment).

- You receive the following error message:



Error: Git

Git: Host key verification failed.

  Note: This error occurs because Git must be able to ensure that the Git client (the AWS Cloud9 environment) has the correct permissions to read and write to the CodeCommit repository. You could set up SSH authentication, but there's an easier way to connect your IDE to CodeCommit.

- Close the error message.

CodeCommit simplifies Git client setup by offering the HTTPS (GRC) connectivity option. This works perfectly with AWS Cloud9 because this IDE uses temporary AWS credentials in the background. Because your AWS account in this lab has full access to CodeCommit, you can use HTTPS (GRC) to have the same level of access to your AWS Cloud9 environment.

25. Retrieve the HTTP (GRC) Clone URL for your repository.
    - Return to the CodeCommit console.
    - Under **Clone URL**, choose **HTTPS (GRC)** to copy the URL to your clipboard.

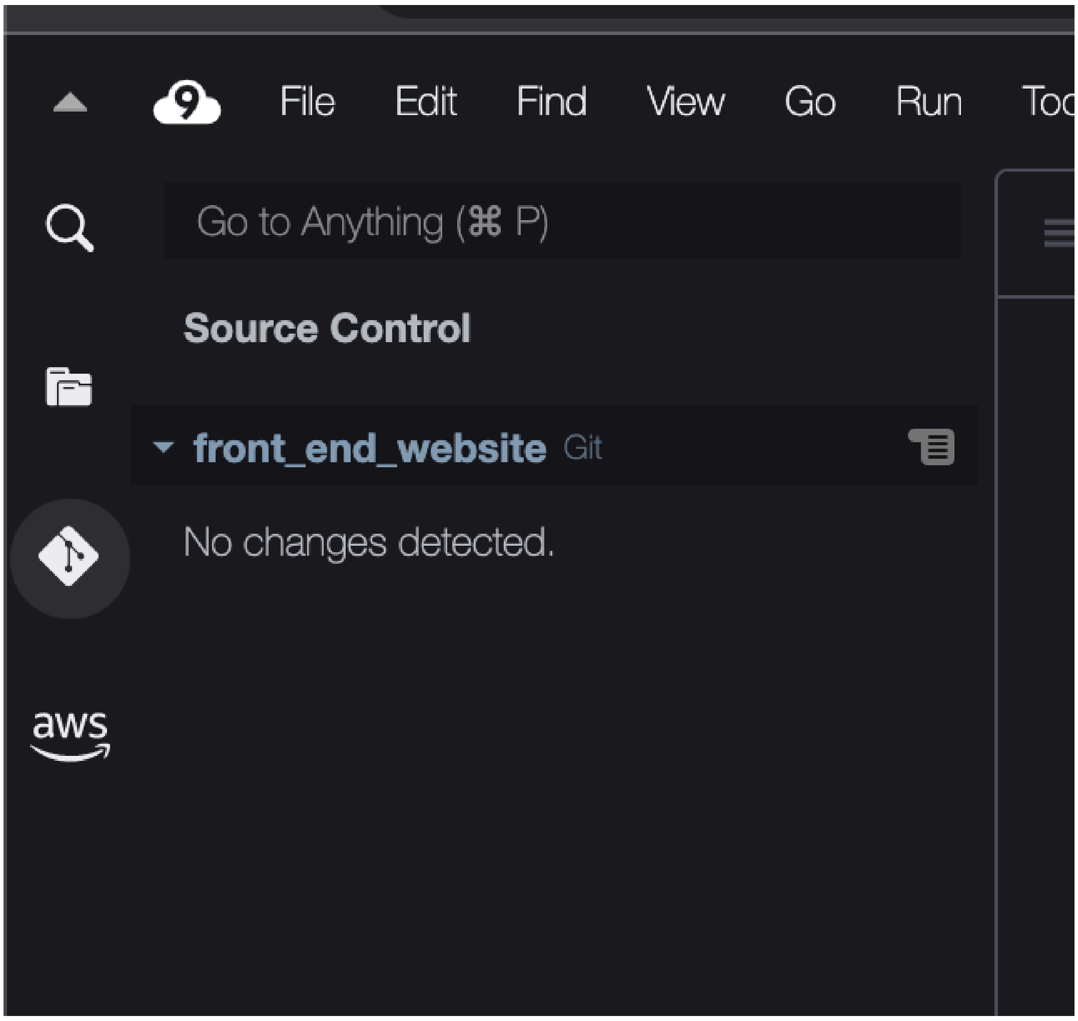
26. Clone your repository by using the HTTP (GRC) URL.
    - Return to the AWS Cloud9 terminal.

- Choose the Git source control icon.
- Choose **Clone Repository**, and paste the HTTPS (GRC) URL, which looks like the following:

```
codecommit::us-east-1://front_end_website
```

- Press Enter.
- If prompted to enter a folder location, choose the default setting. This will place the repository files under environment/front_end_website.

  After the *front_end_website* repository is cloned, you can see it listed under **Source Control**, as shown in the following image.



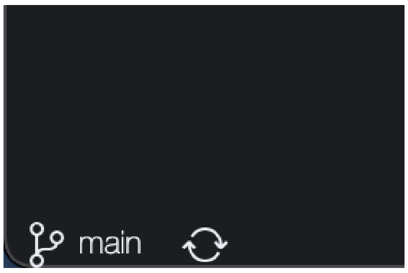
You now have a local clone of your repository that can be edited using your IDE. Next, you will lean how to manage your local copy of the repository and synchronize changes with CodeCommit.

Task 5: Exploring the Git integration with the AWS Cloud9 IDE

You can interact with the repository through the command line, but AWS Cloud9 provides Git integration in the IDE to make it easier to manage your repository. In this task, you will perform simple repository management operations by using this integration.

27. Explore repository branch management.
    - Locate the branch icon, which is located in the lower-left corner of the IDE.

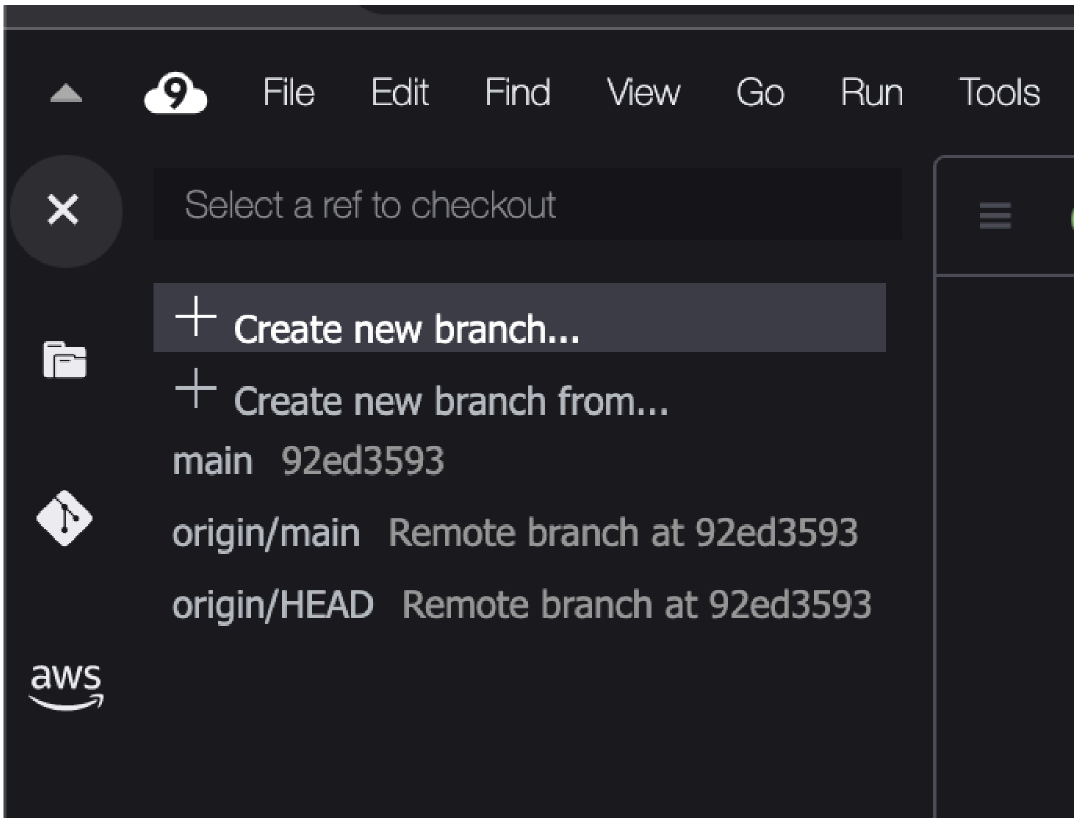
      It's the icon that is furthest to the left in the following image.

      The IDE is currently set up to communicate with the **main** branch.

    - Choose the branch icon to reveal the branch management options in the IDE.

      The options look similar to the following image.

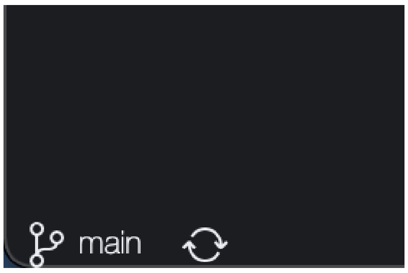
      main 92ed3593
      origin/main Remote branch at 92ed3593
      origin/HEAD Remote branch at 92ed3593

      This is where you can locally create multiple repository branches and switch between branches.

      Because you will use the **main** branch for this update, you don't need to create additional branches.

28. Explore repository synchronization.
    - Choose the synchronize icon, which is located in the lower-left corner of the IDE.

      It's the icon that is furthest to the right in the following image.



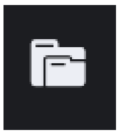
- When prompted with the warning message, choose **OK, Don't Show Again**.
- When asked if you would like to periodically run 'git fetch', choose **No**.

29. Explore and edit the repository files.

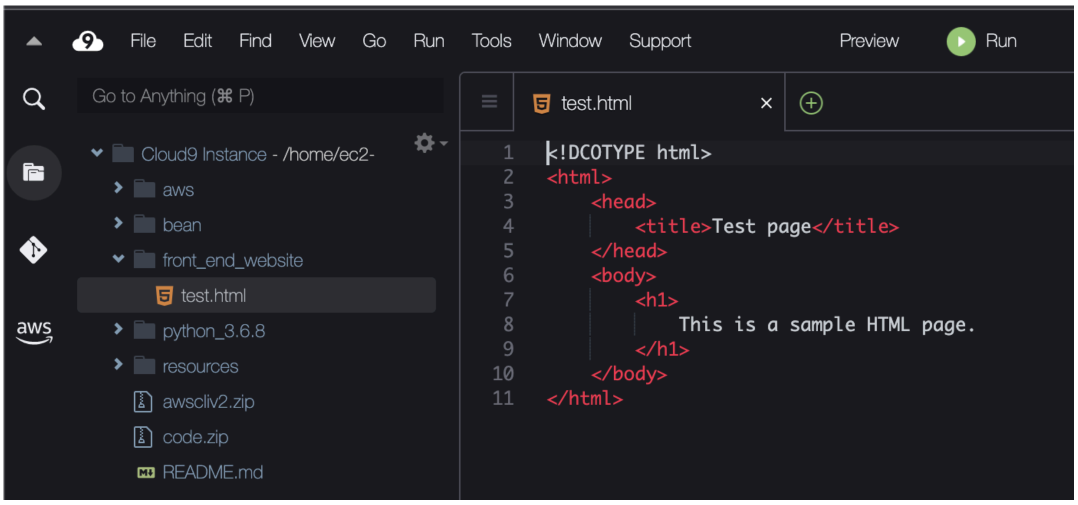
    Remember that the repository was cloned to the local folder environment/front_end_website. You can open repository files in your IDE to view and edit them.

    - Choose the files icon, which is located in the upper-left corner.

      The icon looks like the following image:

    - Expand the **front_end_website** folder to reveal the **test.html** file, which is shown in the following image.

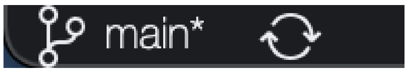
```
<!DCOTYPE html>
<html>
    <head>
        <title>Test page</title>
    </head>
    <body>
        <h1>
            This is a sample HTML page.
        </h1>
    </body>
</html>
```

    - Open the **test.html** file and edit the page title on line 4. Replace the current title with the following text:

      ```
      Best test page ever.
      ```

    - Save your changes.

An asterisk now appears next to **main** in the lower-left corner. This indicates that changes have been made to the code that need to be committed to the branch.

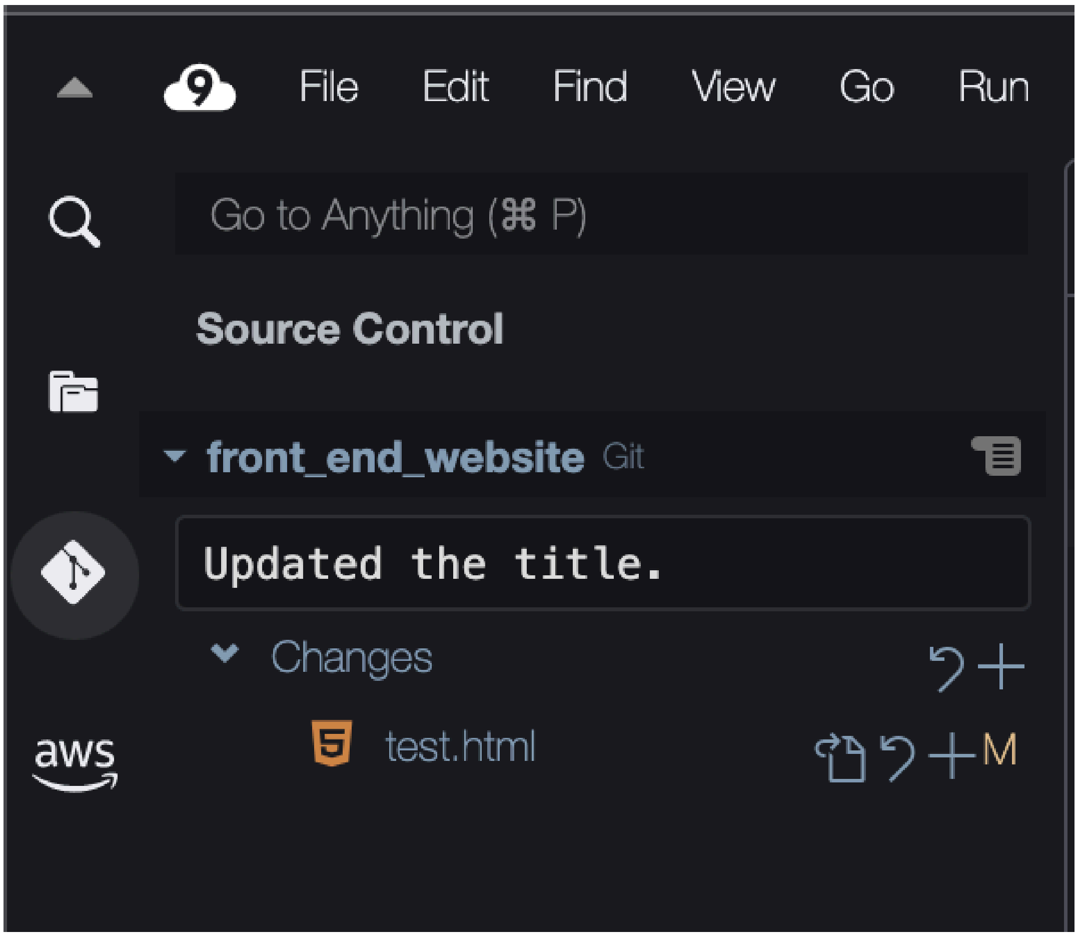


30. Commit your changes to the **main** branch.
    - Choose the Git source control icon.

      Note: You don't have to use the IDE integration. You could also use Git from the command line. For example, if you enter the following command, you should find the *test.html* file listed in the output.

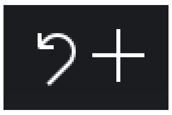
      ```
      cd ~/environment/front_end_website
      git status
      ```

    - In the **Message** text box, enter `Updated the title` as shown in the following image.

      Note: Notice that one file, *test.html*, is listed under **Changes**. This is the same file that was returned by the *git status* command.

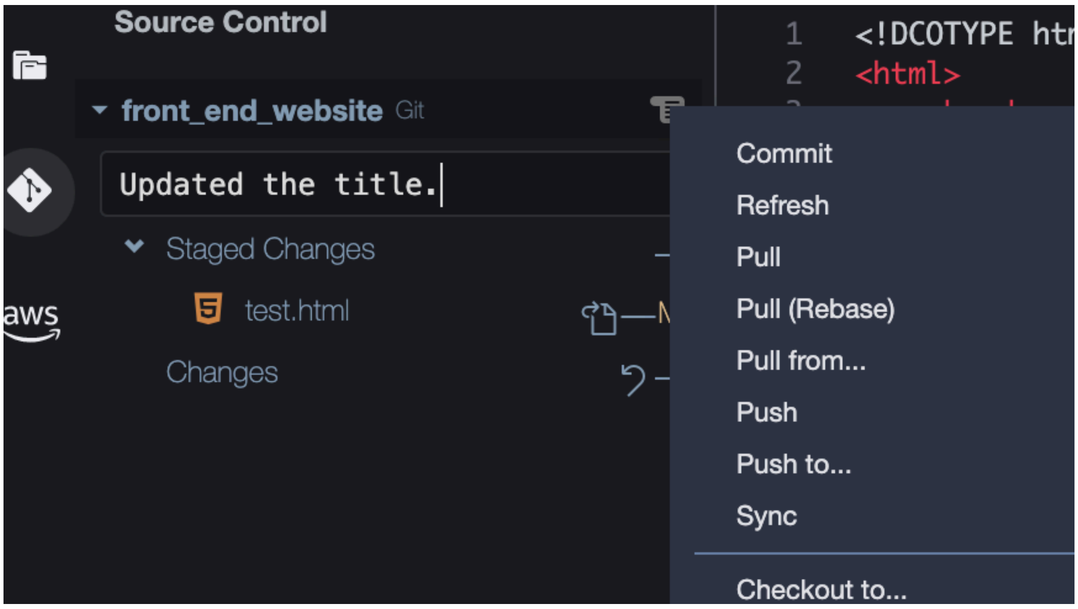
    - Choose the **Stage Changes** icon, which is located to the right of **Changes** and is shown in the following image:

- In the AWS Cloud9 terminal, enter the following commands to configure your user name and email address. Replace the *<User_Name>* and *<User_Email>* placeholders with your name and email address.

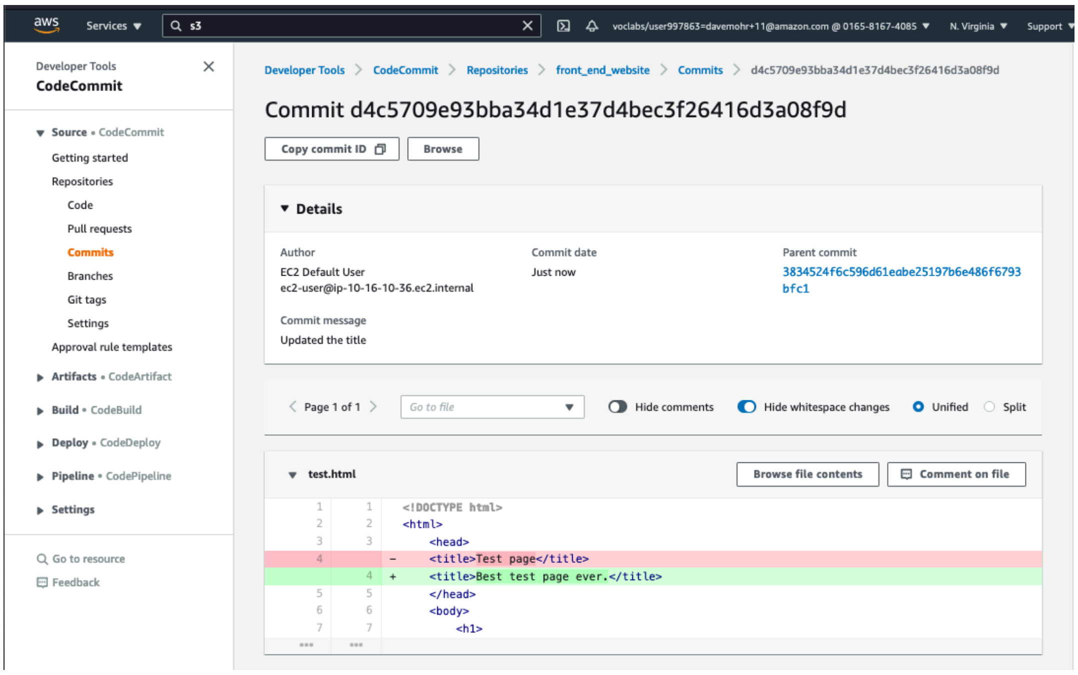
```
git config --global user.name "<User_Name>"
git config --global user.email <User_Email>
```

- Choose the menu icon to the right of **front_end_website**, and choose **Commit**, as shown in the following image:



- Open the menu again, and choose **Push**.

31. Review the changes in CodeCommit.
    - Navigate to the CodeCommit console.
    - Choose the **front_end_website** repository link.
    - In the navigation pane, under **Repositories**, choose **Commits**.
    - Choose the link for the commit ID with the most recent commit date.
    - Go to the **test.html** section, and notice the highlighted lines, which are shown in the following image.



Commit d4c5709e93bba34d1e37d4bec3f26416d3a08f9d

Details

Author: EC2 Default User ec2-user@ip-10-16-10-36.ec2.internal

Commit date: Just now

Parent commit: 3834524f6c596d61eabe25197b6e486f6793bfc1

Commit message: Updated the title

Hide comments: off · Hide whitespace changes: on · Unified [✓] · Split

test.html

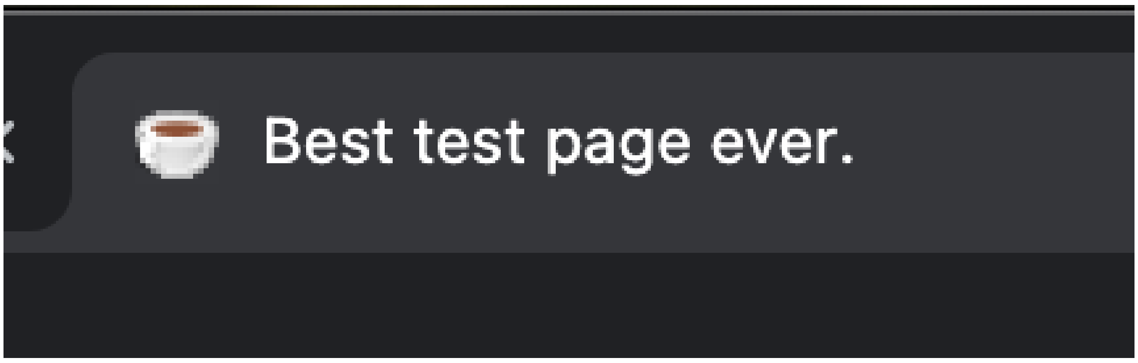
```
     <!DOCTYPE html>
     <html>
         <head>
-        <title>Test page</title>
+        <title>Best test page ever.</title>
         </head>
         <body>
             <h1>
```

The first highlighted line is preceded by a minus sign (-) and highlighted in red. This shows the contents of the line before the change was made. The second highlighted line is preceded by a plus sign (+) and highlighted in green. This line shows the current content of the line.

Great work! You have confirmed that AWS Cloud 9 is able to push changes to your CodeCommit repository.

32. Now, return to the tab that contains the café website and refresh the page.

    Notice that the browser tab title has changed, as shown in the following image. This proves that the pipeline deployed the changes that you committed from your local repository.



Note: You might need to refresh the page a few times before you see the new tab title.

Now that you have learned to manage your local repository and synchronize it with CodeCommit, it's time to update the repository with the actual café website code.

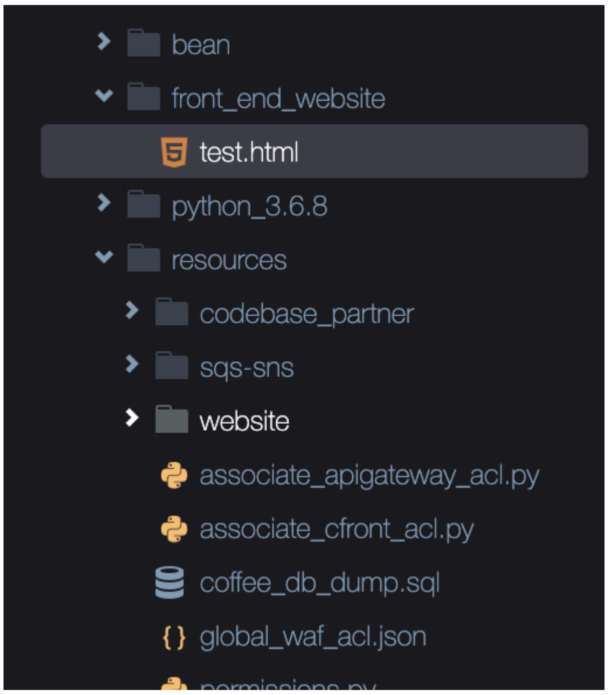
Task 6: Pushing the café website code to CodeCommit

In this task, you will first remove the test.html file. Then, you will update the local repository with the café website code. Finally, you will verify that your pipeline built the café website on S3. You will also verify that *max-age* is set to 14 to confirm that the latest changes are being applied and the caching was updated.

33. Return to the AWS Cloud9 tab.

34. Choose the files icon.

35. Expand the **front_end_website** folder and delete the **test.html** file, as shown in the following image.



36. In the AWS Cloud9 terminal, run the following commands.

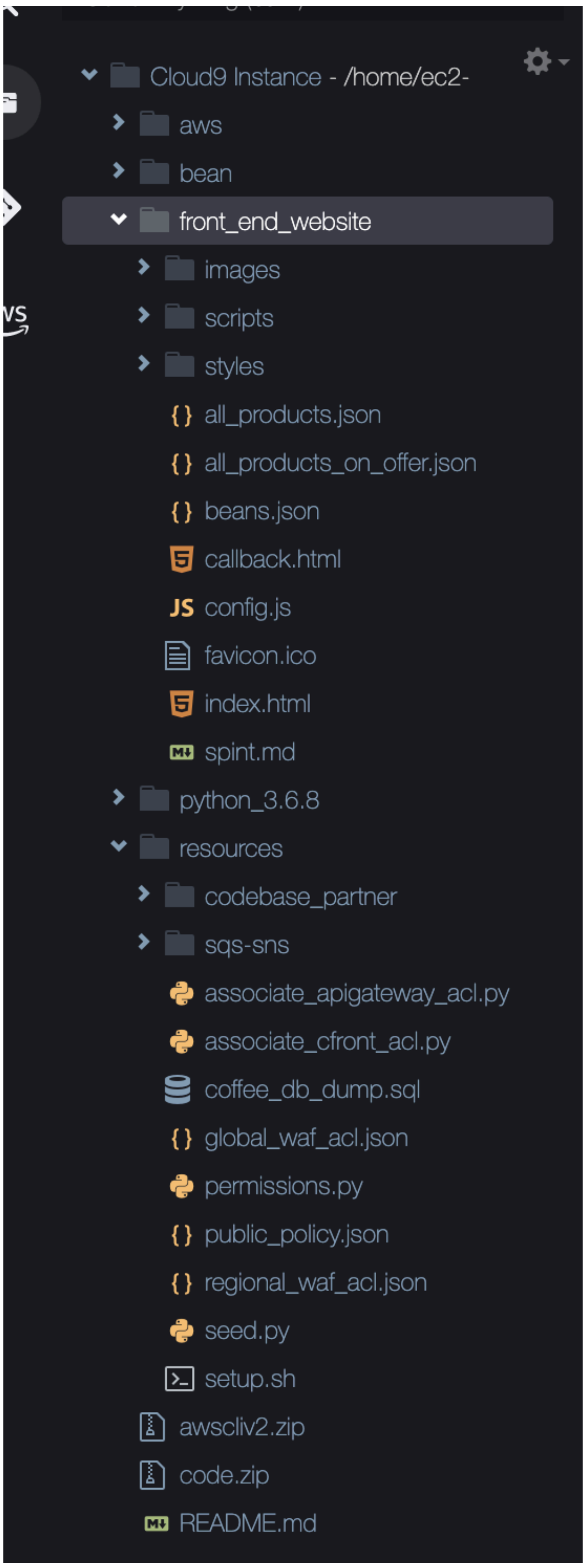
    The second command will copy all of the contents under the *website* folder to the *front_end_website* folder.

```
cd ~/environment
cp -r ./resources/website/* front_end_website
```

37. To delete the *website* folder, so that there is one source of truth, run the following command.

```
rm -r ./resources/website
```

    Now, your local repository has the folder structure shown in the following image:



38. Commit your changes.
    - Choose the Git source control icon.
    - Choose the Stage Changes icon.
    - Enter the following commit message: `Providing the website`
    - To the right of **front_end_website**, choose the options icon and choose **Commit**.
    - Open the options menu again and choose **Push**.
    - The status in the lower-left corner of the window briefly shows the message *Updating Git*.

39. Return to the browser tab that shows the *test.html* page.

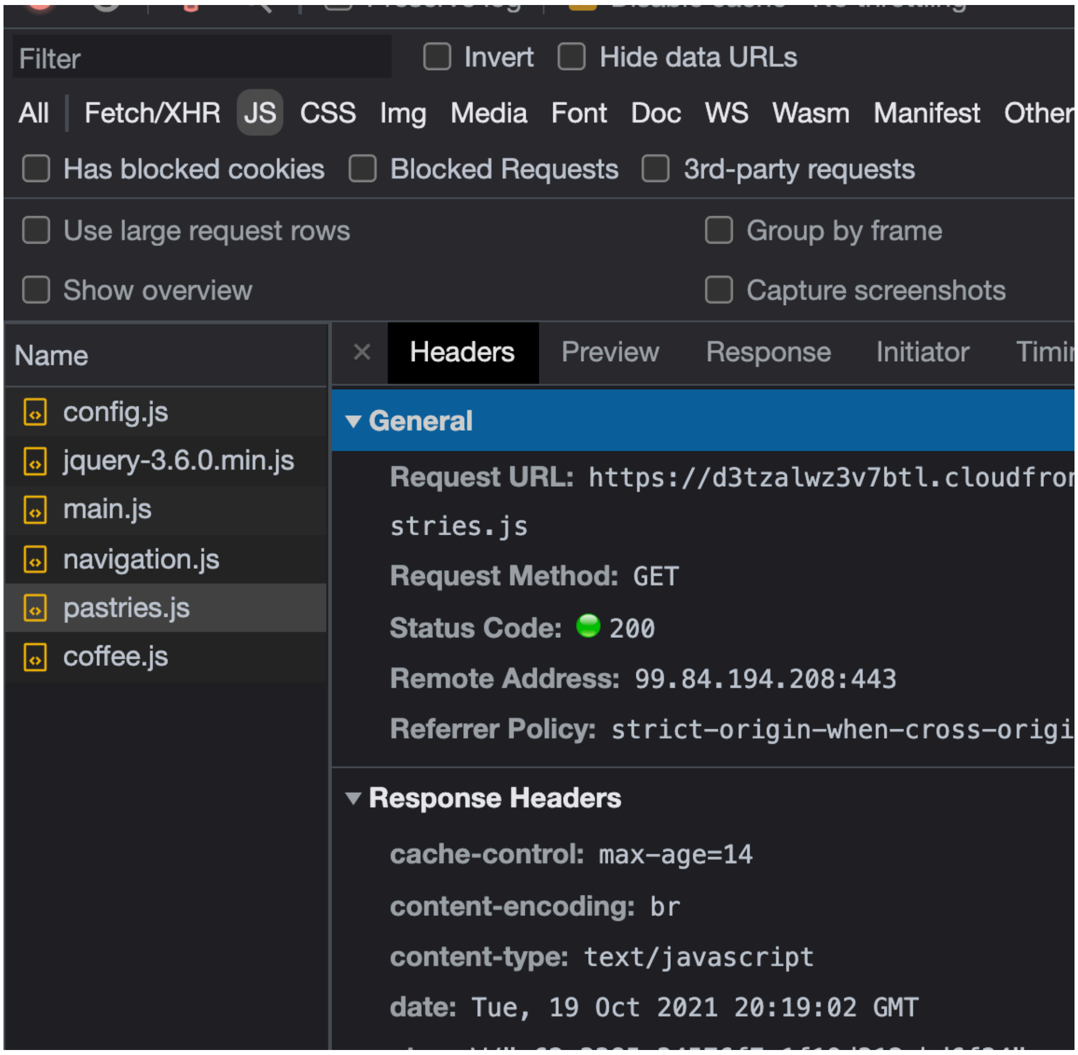
40. Update the URL by removing **test.html** from the address, and then submit the updated URL.

    The updated address is similar to the following example:
    `https://aaabbb111222.cloudfront.net`

    The browser now displays the café website. Well done!

41. Verify that your pipeline applied the cache-control setting, which was configured in Task 3.
    - Remain on the café website page, and open your browser's developer tools.

      Note: To open the developer tools, open your browser's context menu (right-click) and choose **Inspect**.
    - Choose the **Network** tab, and then refresh the webpage.
    - Choose **pastries.js**.
    - Choose the **Headers** tab, and locate the **Response Headers** section, as shown in the following image.



config.js
jquery-3.6.0.min.js
main.js
navigation.js
pastries.js
coffee.js

General

Request URL: https://d3tzalwz3v7btl.cloudfro
stries.js
Request Method: GET
Status Code: 200
Remote Address: 99.84.194.208:443
Referrer Policy: strict-origin-when-cross-origi

Response Headers

cache-control: max-age=14
content-encoding: br
content-type: text/javascript
date: Tue, 19 Oct 2021 20:19:02 GMT

Notice that the **cache-control** value is set to *max-age=14*, which indicates that pipeline updated the cache settings. This means that the website is being built from the most recent repository update.

Note: If **cache-control** is set to *max-age=0*, the pipeline might still be applying the update to the S3 bucket. Wait a few seconds, refresh the page, and choose **pastries.js** again.

Congratulations! You have moved the codebase to a secure and scalable managed service. You have also ensured that the website will stay up to date with the latest improvements.

Update from the café

This is a big win for the café! Now that the codebase is centralized, the café can bring in more developers to collaborate and enhance the site as the business grows. Sofía doesn't have to remember to update the website, because the CI/CD pipeline will automatically deploy changes to the website. She can track versioning and implement a code approval process. If any issues arise from updates, she can use CodeCommit to review commit logs to trace changes and resolve code bugs.

Submitting your work

42. At the top of these instructions, choose **Submit** to record your progress and when prompted, choose **Yes**.

    Tip: If you previously hid the terminal in the browser panel, expose it again by selecting the **Terminal** ☐ checkbox. This action will ensure that the lab instructions remain visible after you choose **Submit**.

43. If the results don't display after a couple of minutes, return to the top of these instructions and choose Grades

Tip: You can submit your work multiple times. After you change your work, choose **Submit** again. Your last submission is what will be recorded for this lab.

44. To find detailed feedback on your work, choose Details followed by ▸ **View Submission Report**.

Lab complete 🎓

🏁 Congratulations! You have completed the lab.

45. Choose End Lab at the top of this page, and then select Yes to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

46. Select the **X** in the top-right corner to close the panel.