**Aim** :- Automating application deployment using 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

**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.

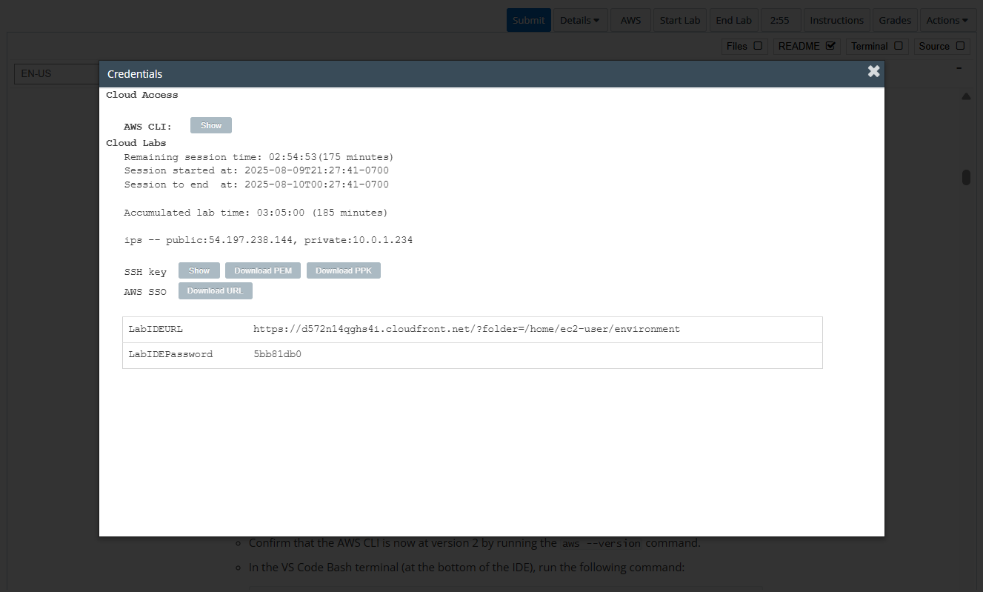
Mateo, a café regular and AWS consultant who specializes in automating repeatable processes, was chatting with Sofía about his work. He mentioned using CodeCommit to collaborate on projects with other developers. Sofía shared that she was researching CodeCommit to centralize the café website's code and 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).

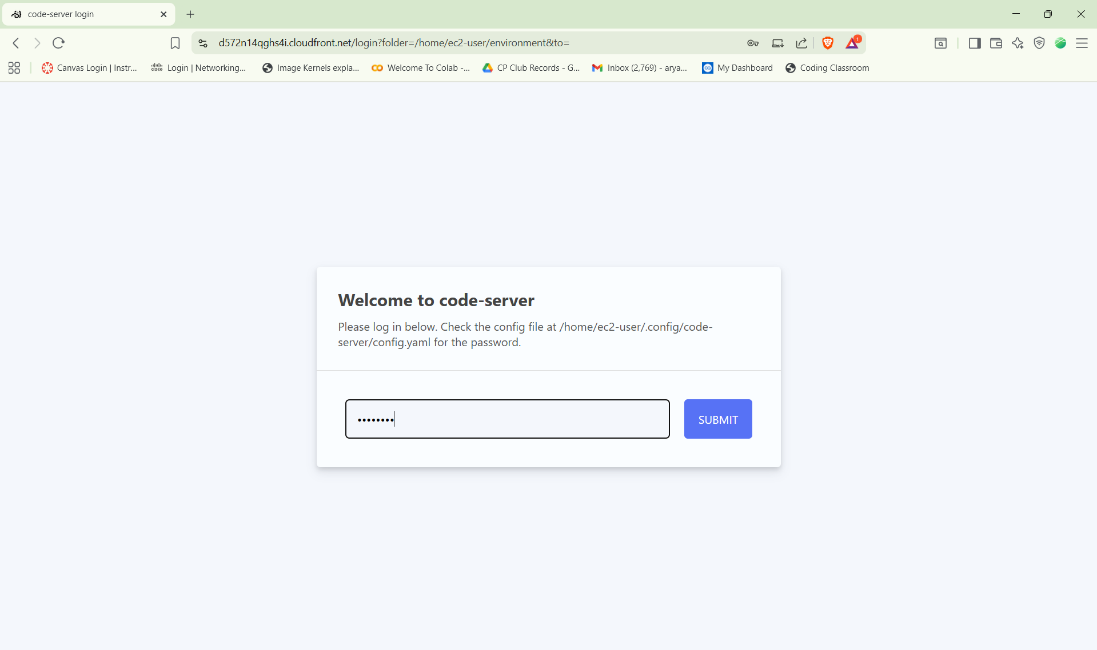
When you *start* the lab, many services are deployed for you. In this lab, you will focus on the services that are represented in the following diagram. As in past labs, the VS Code IDE is used as the development environment. The developer runs commands from VS Code IDE to update the code in the S3 bucket 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.

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.

**Task 1: Preparing the lab**

Connect to the VS Code IDE.

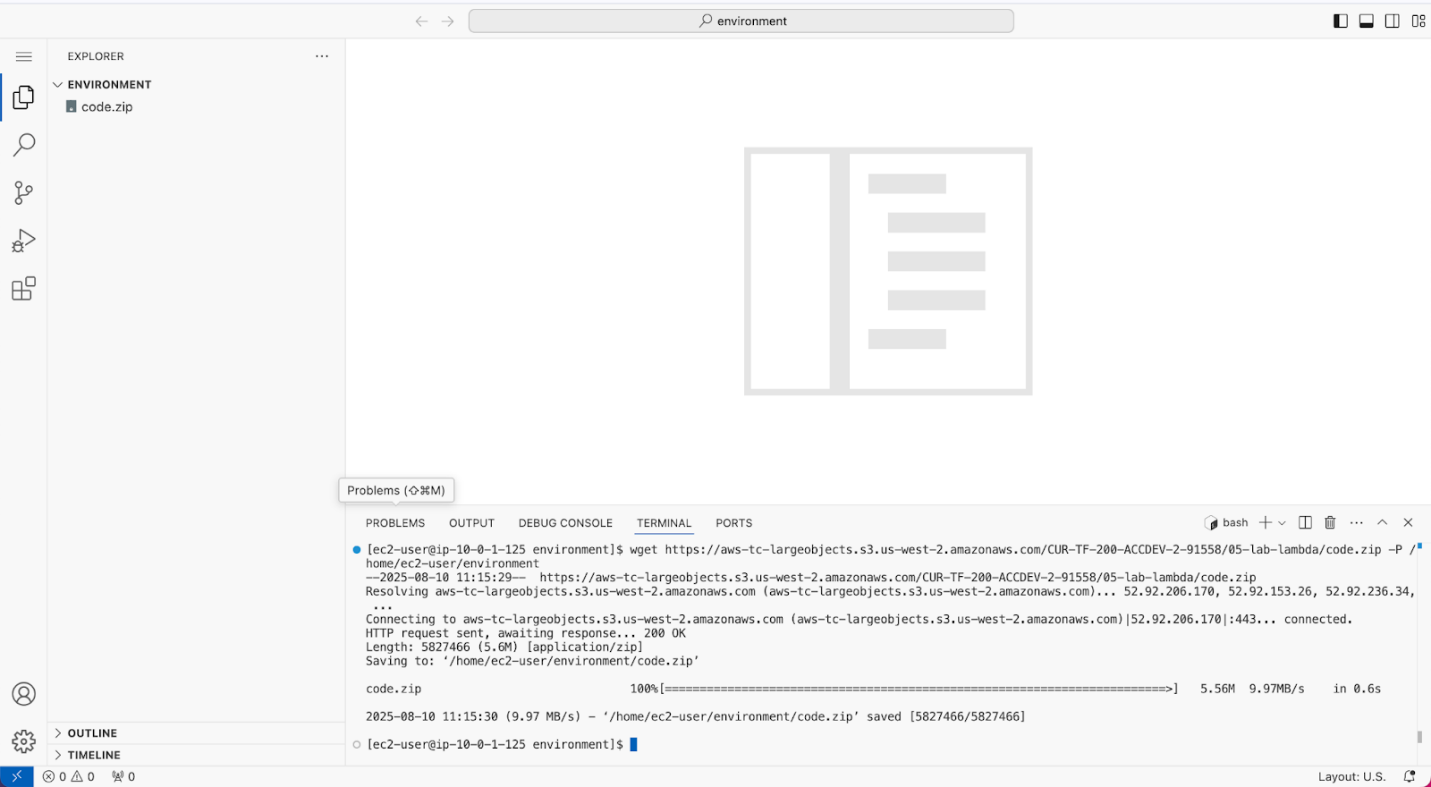
1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
   1. **LabIDEURL**
   2. **LabIDEPassword**
3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.

****

1. Download and extract the files that you need for this lab.

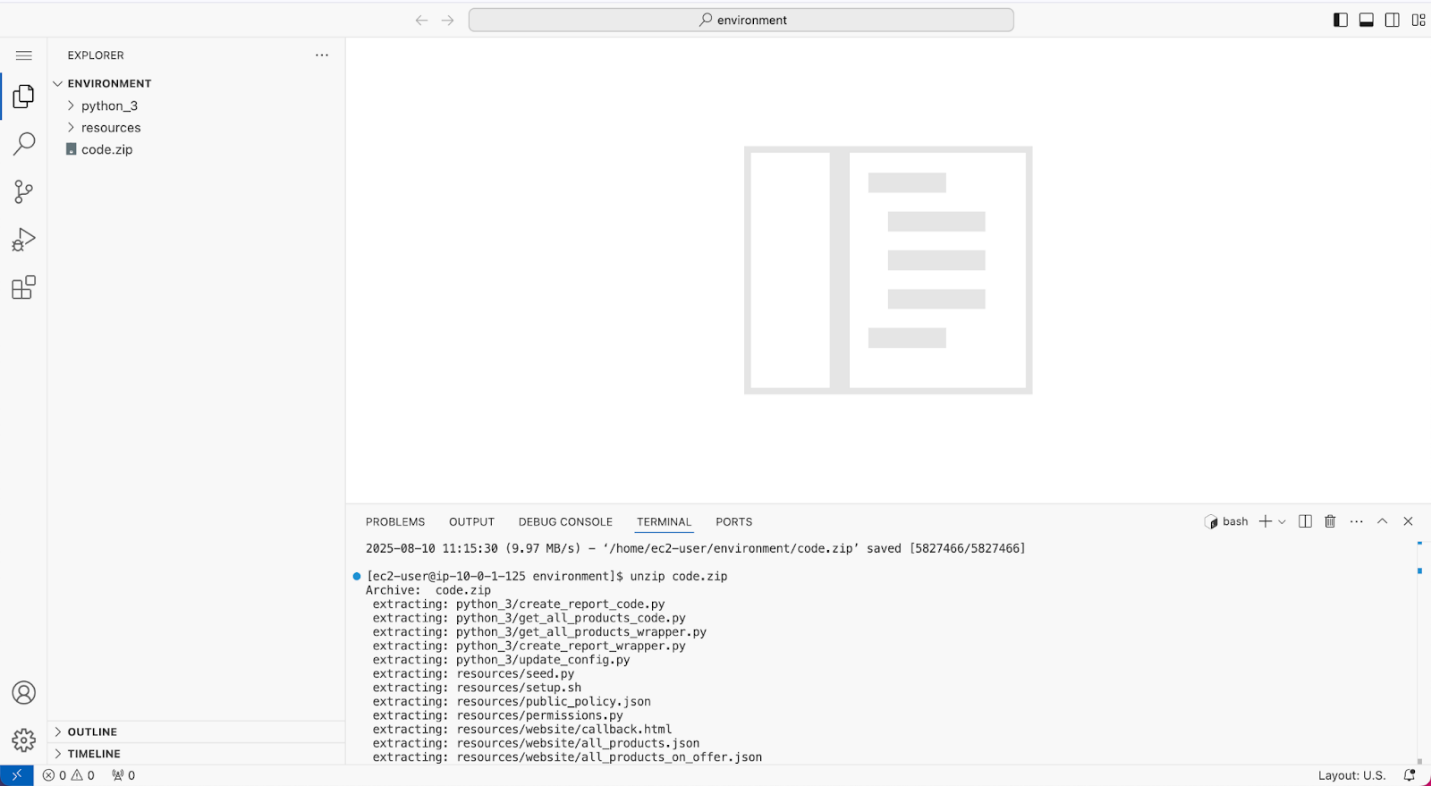
* In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

**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**



1. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.

* Extract the file by running the following command:

unzip code.zip

1. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.

* To set permissions on the script and then run it, run the following commands in the Bash terminal:

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

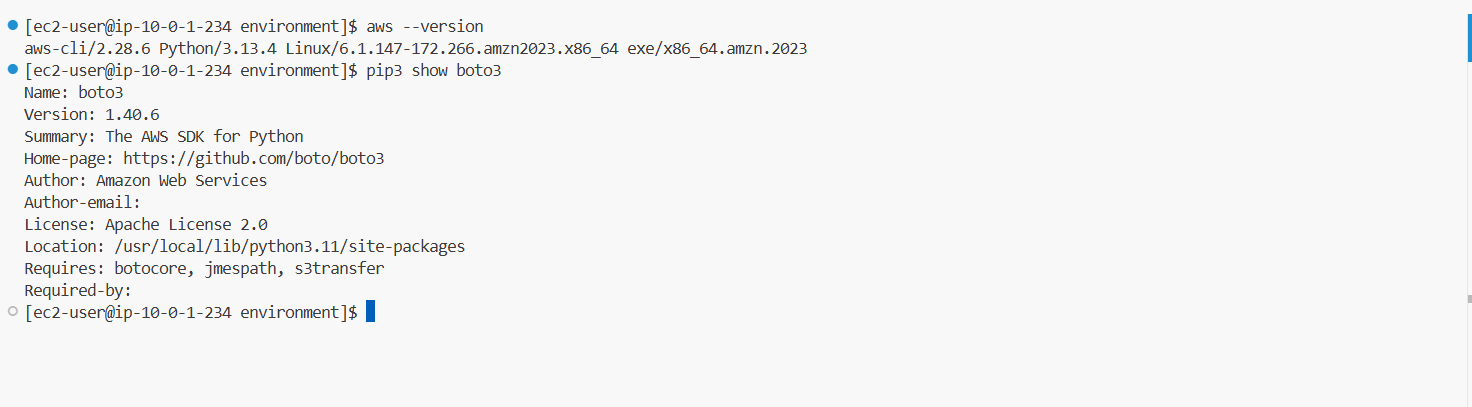
The script will prompt you for the **IP address** by which your computer is known to the internet.

Use [www.whatismyip.com](http://www.whatismyip.com/) to discover this address and then paste the IPv4 address into the command prompt and finish running the script.



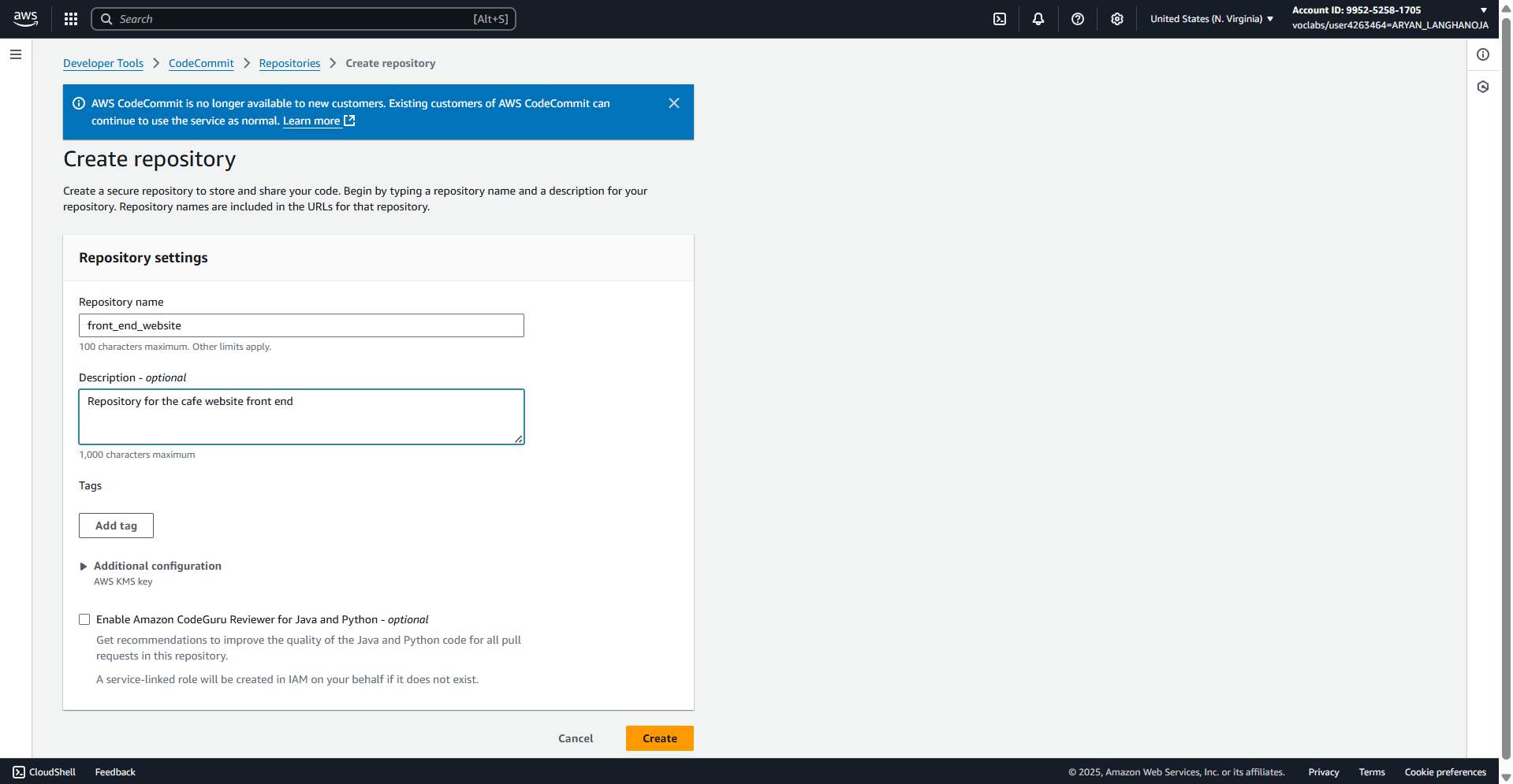
1. Verify the AWS CLI version and also verify that the SDK for Python is installed.

* Confirm that the AWS CLI is now at version 2 by running the **aws --version** command.
* In the VS Code Bash terminal (at the bottom of the IDE), run the following command:

**pip3 show boto3**

**Task 2: Creating a CodeCommit repository**

1. Create a CodeCommit repository to host the codebase.

* In the search bar at the top, search and select CodeCommit to open the AWS CodeCommit service console.
* From the navigation pane, 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**.

1. 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>

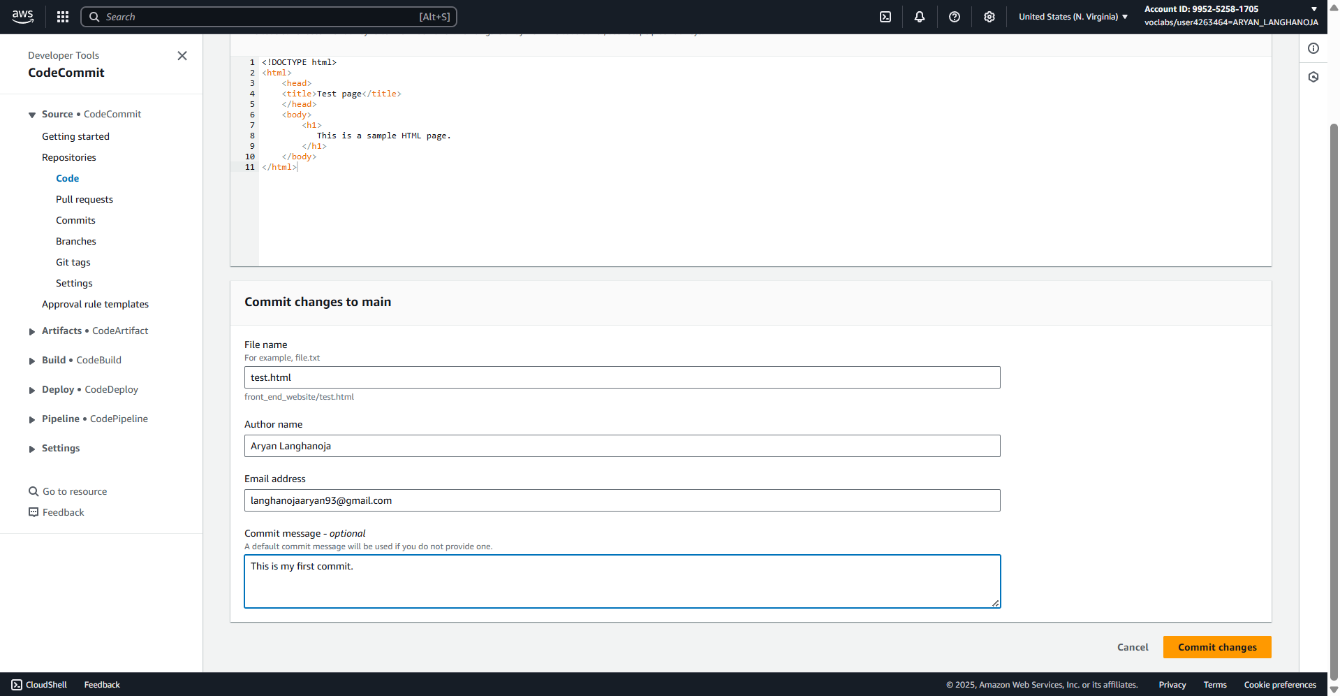
<h1>

This is a sample HTML page.

</h1>

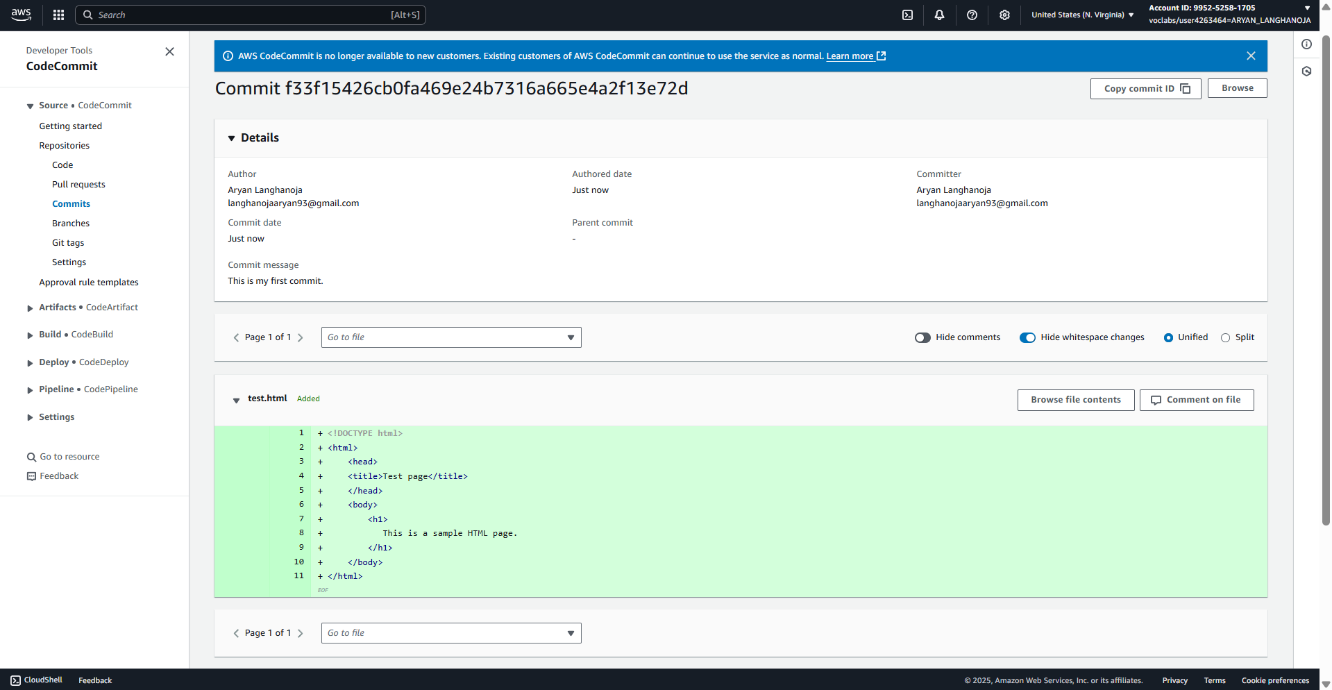
</body>

</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.

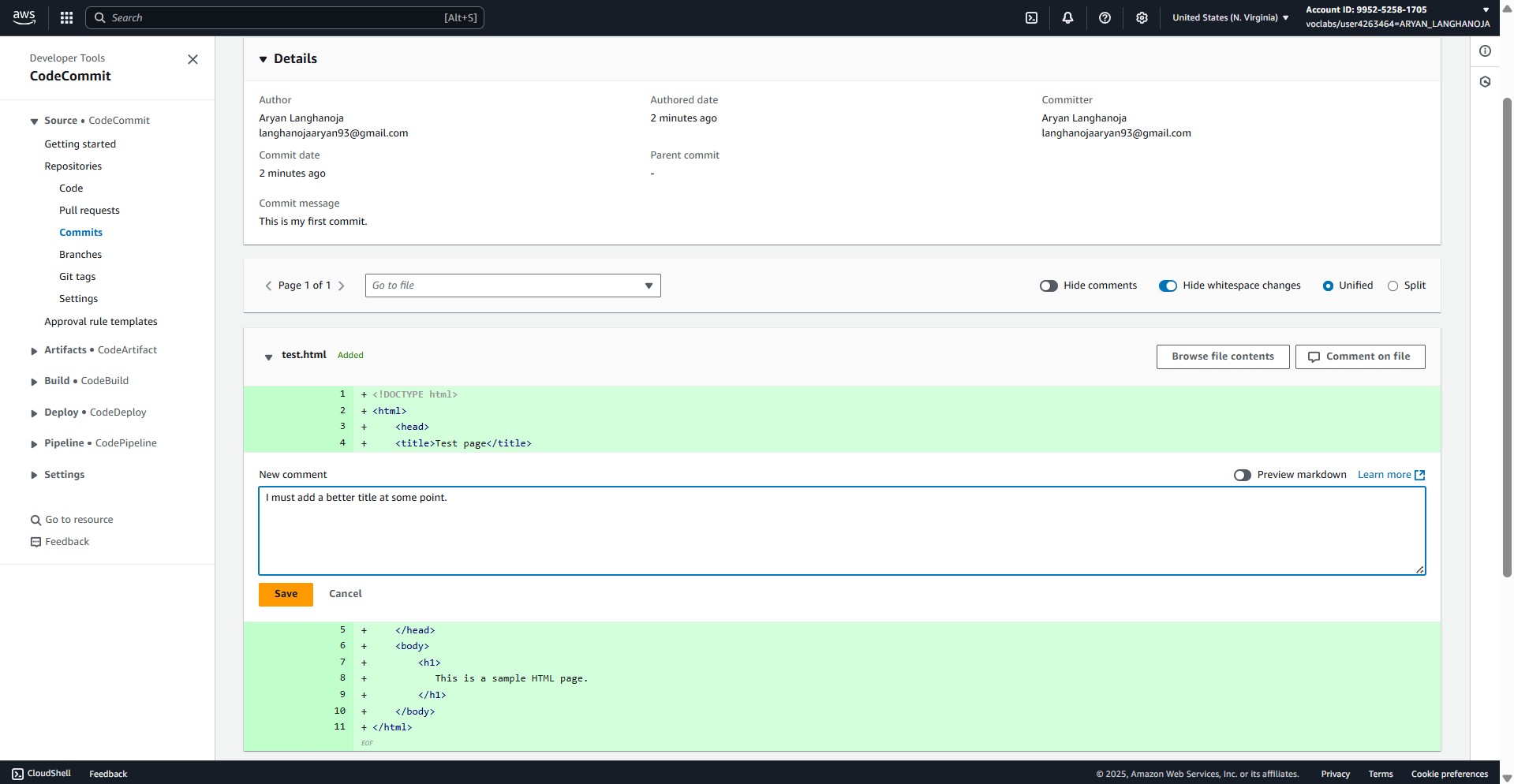
1. Review your commit.

* 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.

****

1. 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.
* Choose **Save**.



**Task 3: Creating a pipeline to automate website updates**

1. Return to the VS Code IDE.

* In the Explorer section, expand *Environment*, which is located in the upper-left corner.
* 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.

1. 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
* ****From the navigation pane, choose menu, then choose **File > Save**.

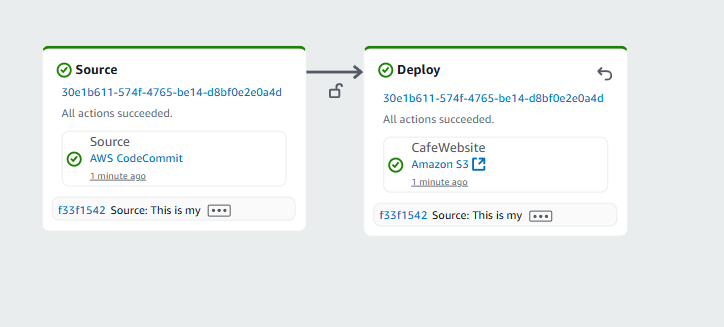
1. To create the pipeline, run the following commands.

**cd ~/environment/resources**

**aws codepipeline create-pipeline --cli-input-json file://cafe\_website\_front\_end\_pipeline.json**



1. Choose the **cafe\_website\_front\_end\_pipeline** hyperlink and review the pipeline status, as shown in the following image.



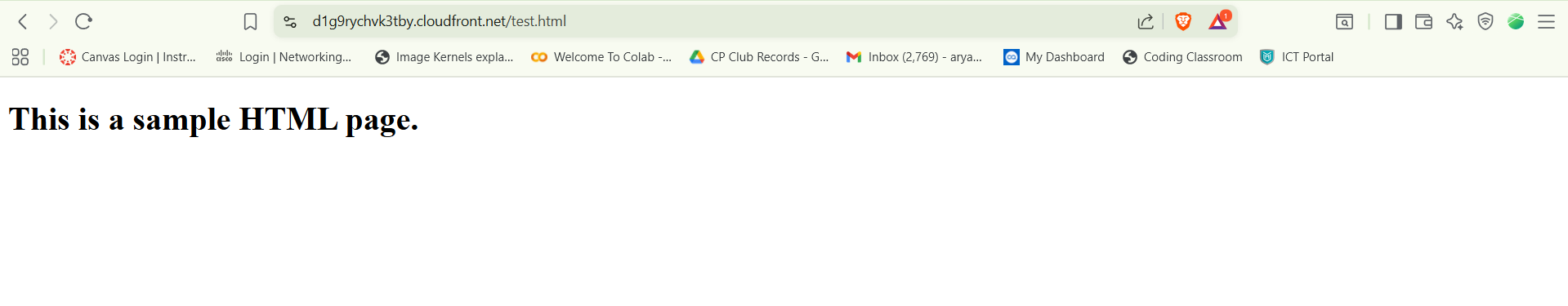
1. Verify the automated deployment.

* Return to the VS Code IDE bash 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://* *d1g9rychvk3tby.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:

****



**Task 4: Cloning a repository in VS Code IDE**

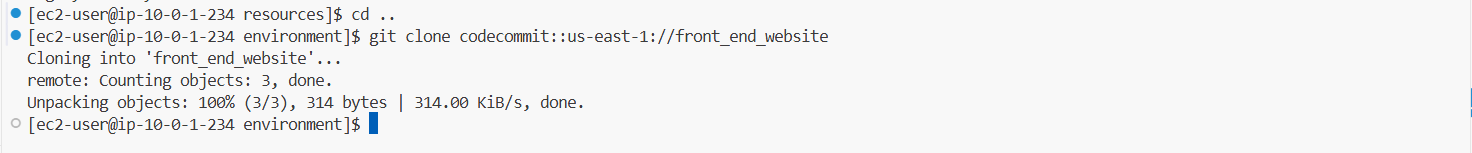
1. Retrieve the SSH clone URL for your repository.

* Navigate to the CodeCommit console.
* In the navigation pane, choose Repositories, then choose the repository created *front\_end\_website*.
* In the Clone URL column, choose Clone HTTPS(GRC) to copy the URL to your clipboard.

1. Clone your repository using the SSH URL.

* Return to the VS Code IDE terminal.
* Run the following command to clone the repository to your VS code IDE, replace the <<Clone URL>> with the value you copied.

cd ..

git clone <<Clone URL>>

**Task 5: Exploring the Git integration with the VS Code IDE**

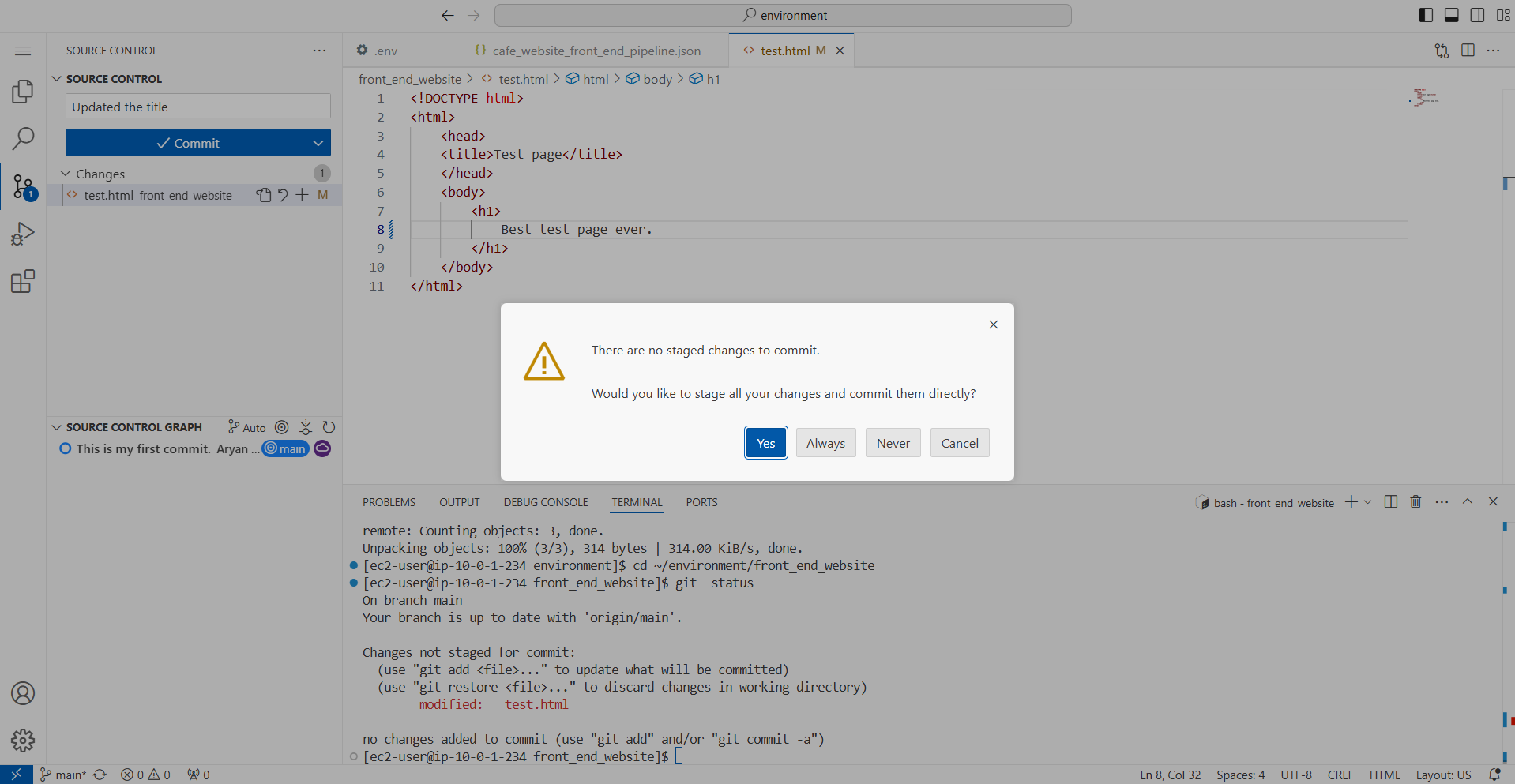
1. Explore repository branch management.

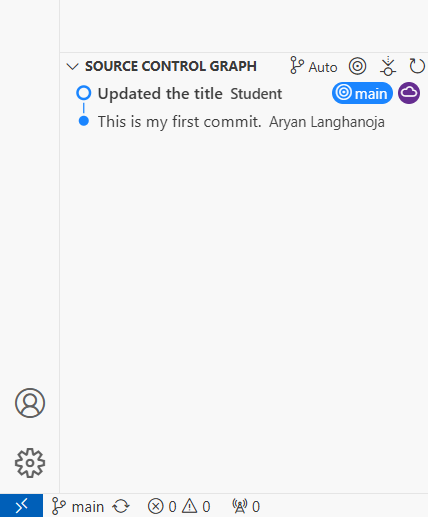
* Locate the branch icon, which is located in the lower-left corner of the IDE.
* Source Control option is opened.
* From the top, choose three dots to the right of *SOURCE CONTROL*.
* From the menu displayed, choose **Source Control Repositories**
* Your repository *front\_end\_website* is displayed.
* The IDE is currently set up to communicate with the **main** branch.
* 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.
* From the explorer menu, expand the **front\_end\_website** folder to reveal the **test.html** file, which is shown in the following image.
* 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.
* Number **1** now appears next to **branch** icon in the left pane. This indicates that changes have been made to the code that need to be committed to the branch.
* Hover over and you will notice a message **Source Control 1 pending change**
* **Note:** This is because you made a change to the file but it is not yet committed to thr repository main branch.

1. Commit your changes to the **main** branch.

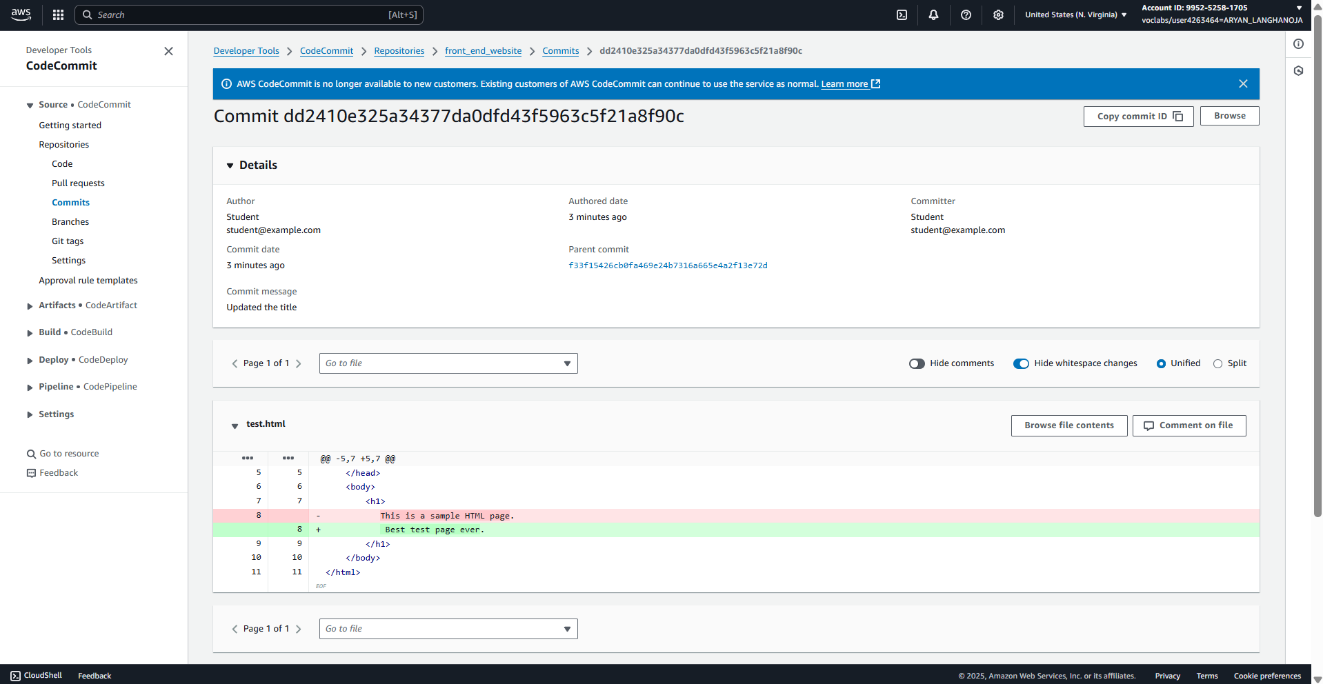
cd ~/environment/front\_end\_website

****git status

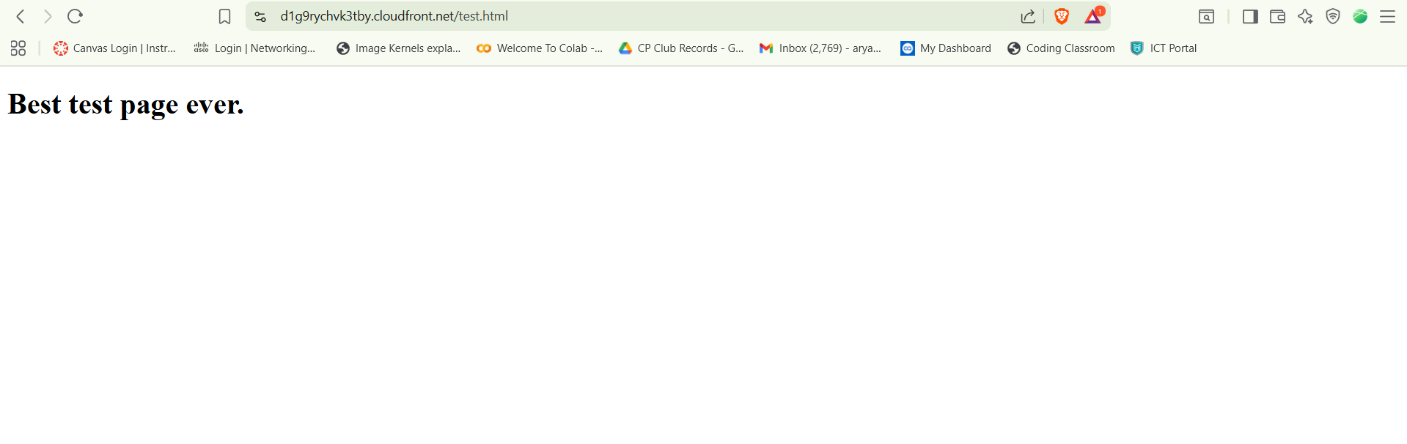
* On the Commit button, choose *more actions* which is located to the right.
* Choose *Commit & Push*, then choose Yes.
* This will push the code and trigger the code pipeline which deploys new version of the file to website.

****

1. 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.

1. 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.

1. Return to the VS Code IDE tab.
2. On the explorer, Under *Environment* folder, expand the front\_end\_website folder and delete the test.html file, as shown in the following image.
3. In the VS Code IDE bash terminal, run the following commands.

The second command will copy all of the contents under the *website* folder to the *front\_end\_website* folder.

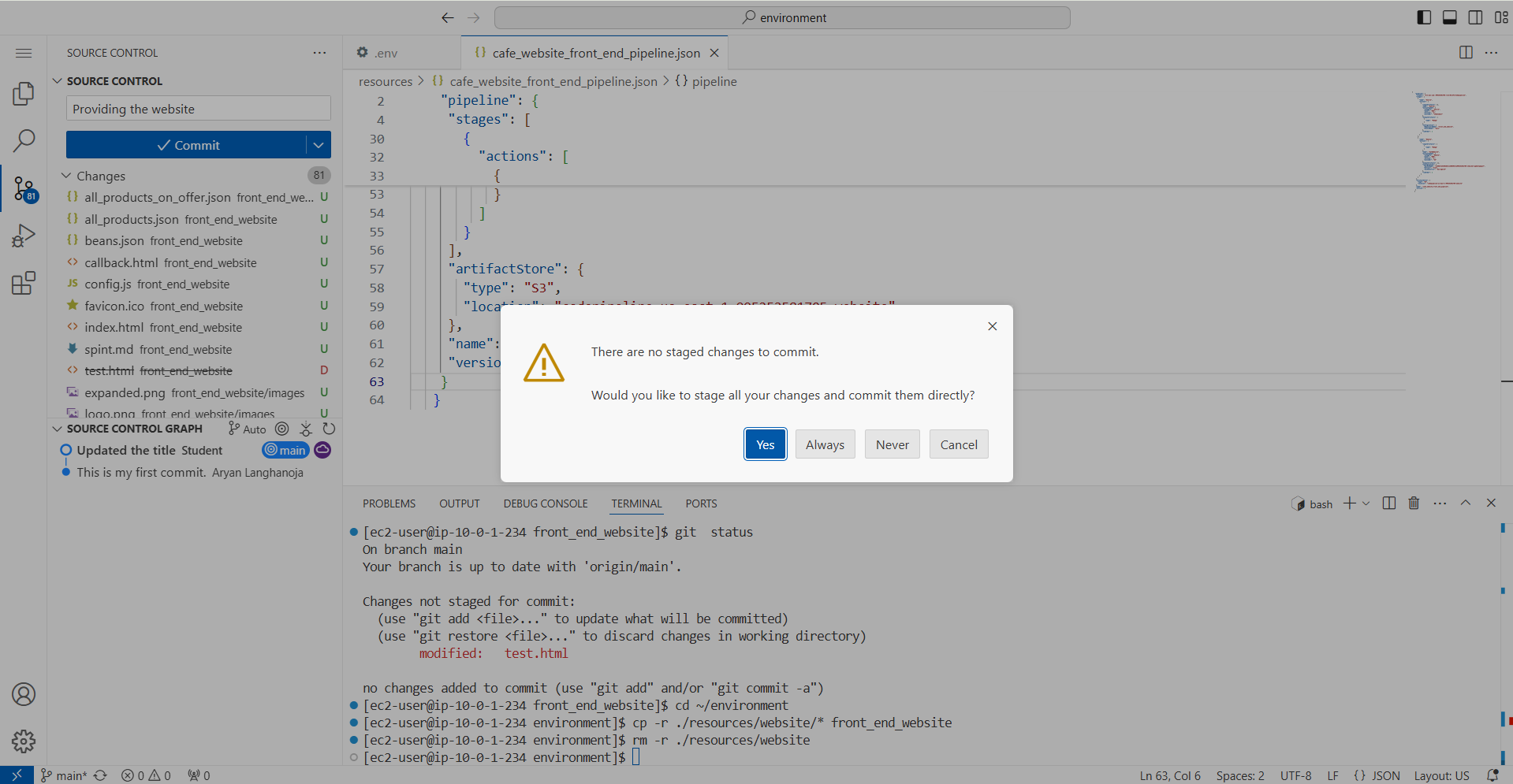
**cd ~/environment**

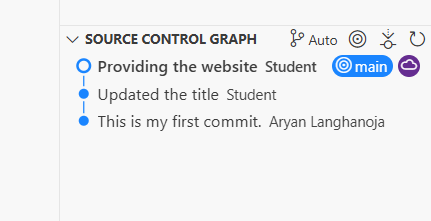
**cp -r ./resources/website/\* front\_end\_website**

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

**rm -r ./resources/website**

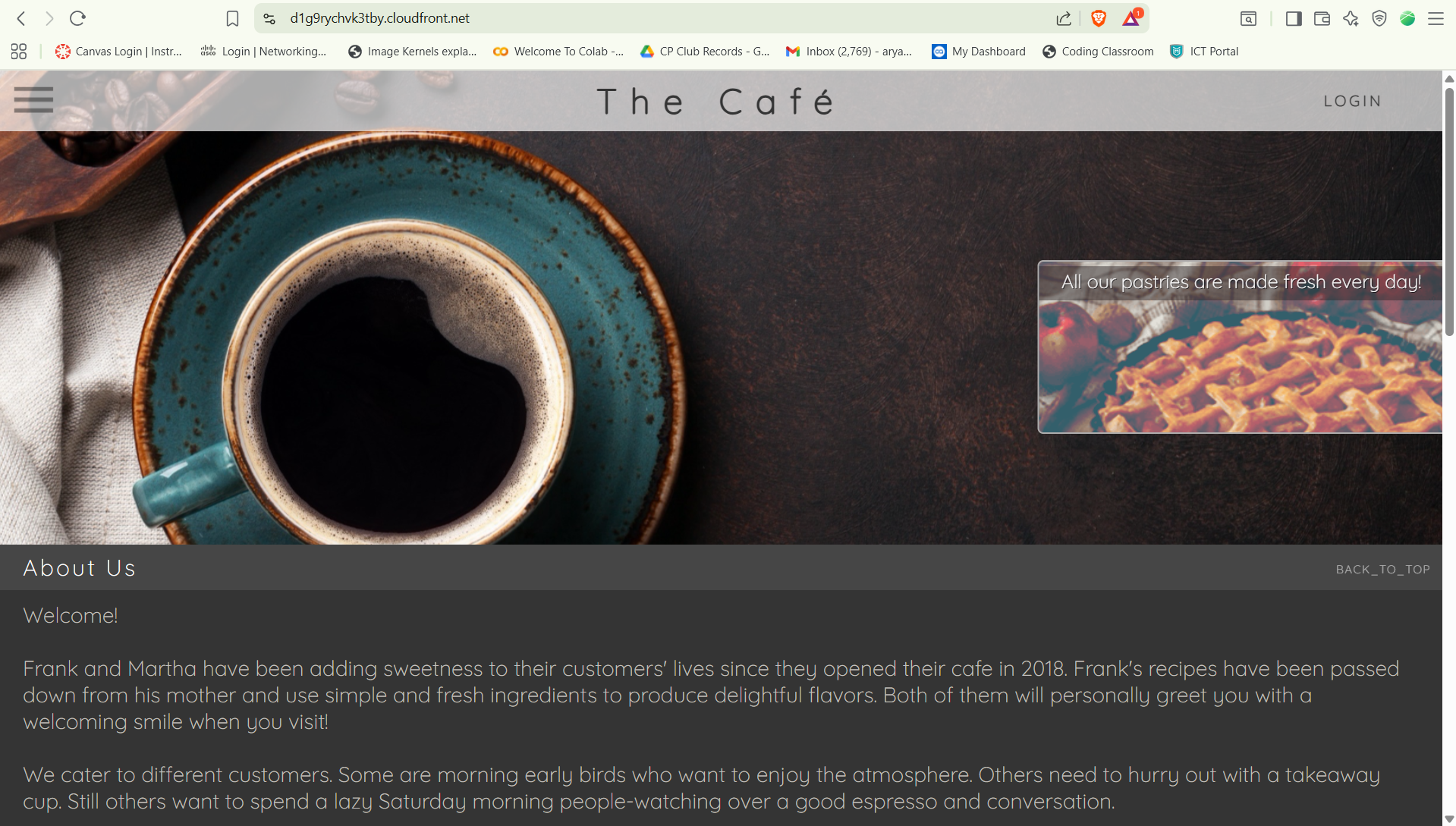
1. Commit your changes.

* Choose the *Branch* icon (now it is showing number of new files to be pushed).
* Enter the following commit message: Providing the website
* On the **Commit** button, choose *more actions* which is located to the right.
* Choose *Commit & Push*, then choose **Yes**.
* To the right of **front\_end\_website**, choose the options icon and choose **Commit**.
* The **Source control Graph** in the lower pane shows history of changes done to the repository.

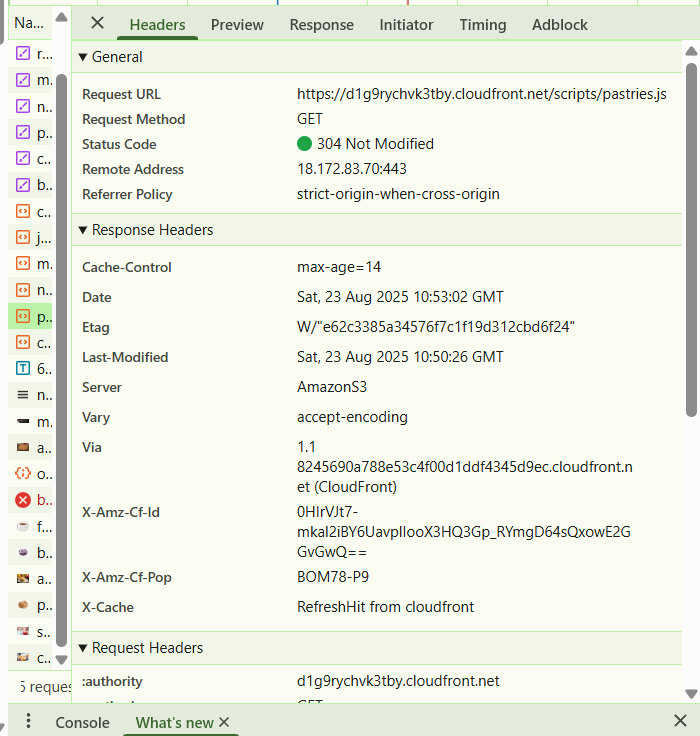
****

1. 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://d1g9rychvk3tby.cloudfront.net**
* The browser now displays the café website. Well done!



1. 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.

**Conclusion:-**

* Connected to VS Code IDE using given LabIDE URL and password.
* Downloaded required lab files using:

**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**

* Extracted files with:

**unzip code.zip**

* Upgraded AWS CLI and configured setup using:

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

* Verified installation with:

**aws --version**

**pip3 show boto3**

* Created a CodeCommit repository named front\_end\_website and committed a test file (test.html).
* Added and reviewed comments in the commit history.
* Updated the pipeline configuration file (cafe\_website\_front\_end\_pipeline.json) with AWS Account ID and S3 bucket.
* Created the pipeline with:

**cd ~/environment/resources**

**aws codepipeline create-pipeline --cli-input-json file://cafe\_website\_front\_end\_pipeline.json**

* Verified automated deployment by retrieving CloudFront domain:

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

* Cloned the repository using:

**git clone <<Clone URL>>**

* Edited and committed changes (test.html) using Git integration in VS Code IDE.
* Verified updates reflected on the website via CloudFront URL.
* Replaced old website files with updated ones using:

**cd ~/environment**

**cp -r ./resources/website/\* front\_end\_website**

**rm -r ./resources/website**

* Committed and pushed full website content with message “Providing the website”.
* Verified final deployment by accessing the CloudFront URL (without test.html).
* Checked cache-control settings using browser developer tools under Network → Headers.

**Result :-**

