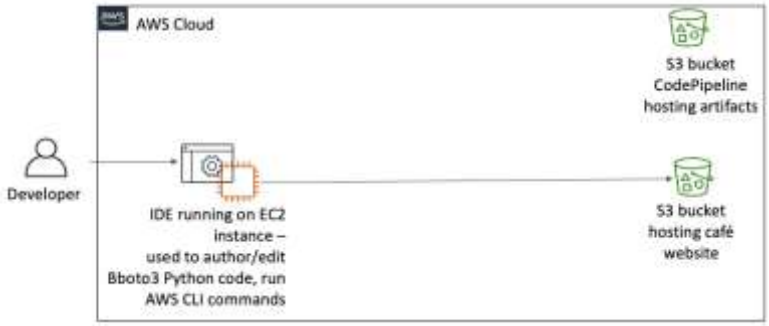
# Experiment – 7

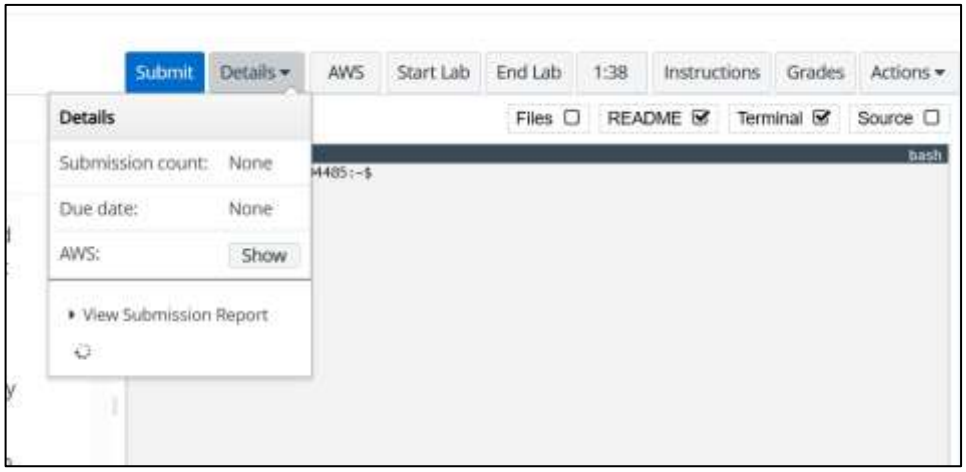
❖ **Aim:** Implement an automated DevOps pipeline on a cloud platform (AWS) to Provision infrastructure using Infrastructure as Code (IaC).




❖ **Task 1: Preparing the development environment**

Before you can start working on this lab, you must import some files and install some packages in the Visual Studio Code Integrated Development Environment (VS Code IDE) that was prepared for you.

1. Connect to the VS Code IDE.
    - At the top of these instructions, choose Details followed by **AWS: Show**




    - Copy values from the table for the following and paste it into an editor of your choice for use later.
        - **LabIDEURL**
        - **LabIDEPassword**
        - In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
        - 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.

2. 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**.
3. Download and extract the files that you need for this lab.
   In the VS Code IDE bash 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**

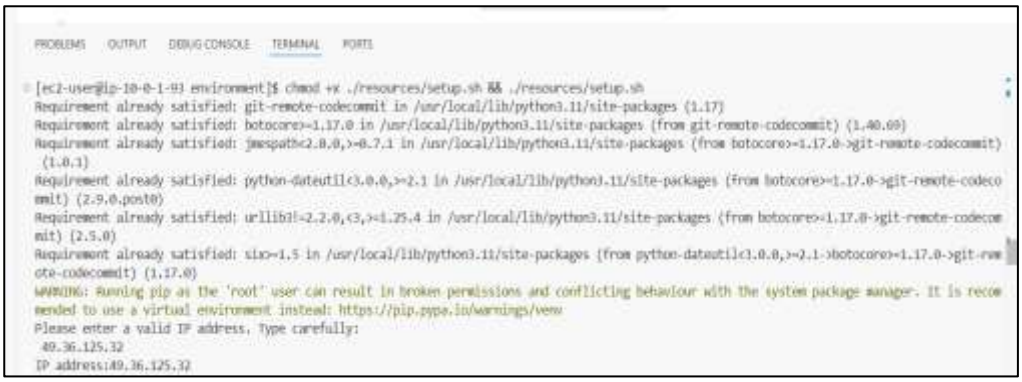
   - Notice that a code.zip file was downloaded to the VS Code IDE. The file is listed in the Environment window.

   - Extract the file:
     **unzip code.zip**

4. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE. 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.
- Verify the version of AWS CLI installed.
- In the VS Code IDE Bash terminal, run the following command:

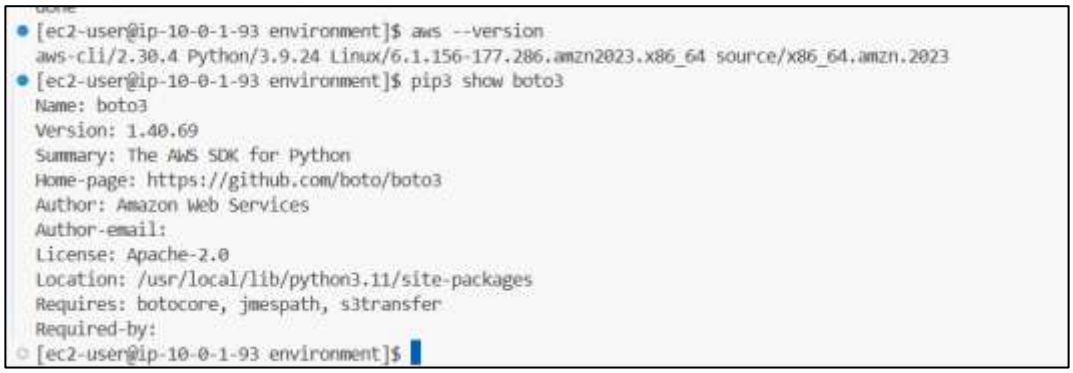
5. Verify the version of AWS CLI installed.

   In the VS Code IDE Bash terminal, run the following command:

   **aws –version**

   The output should indicate that version 2 is installed.

6. Verify that the SDK for Python is installed.
   Run the following command:
   **pip3 show boto3**
   **Note**: If you see a message about not using the latest version of pip, ignore the message.



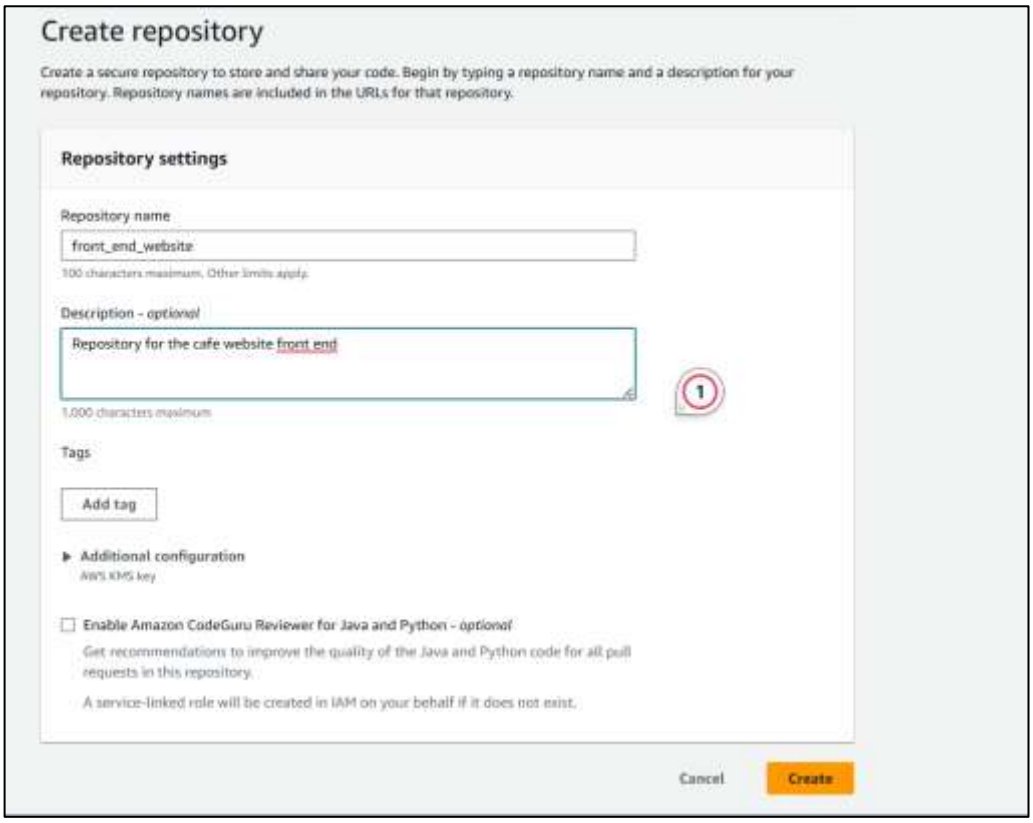
❖ **Task 2: Creating a CodeCommit repository**

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.

7. Create a CodeCommit repository to host the codebase.
    a. In the search bar at the top, search and select CodeCommit to open the AWS CodeCommit service console.
    b. From the navigation pane, choose **Create repository**.
    c. Configure the following settings:
        i. **Repository name:** Enter front_end_website
        ii. **Description:** Enter Repository for the cafe website front end
        iii. Keep the rest of the default settings.
    d. Choose **Create**.




8. Create a test file.
    a. In the **front_end_website** section, choose **Create file**.
    b. 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:

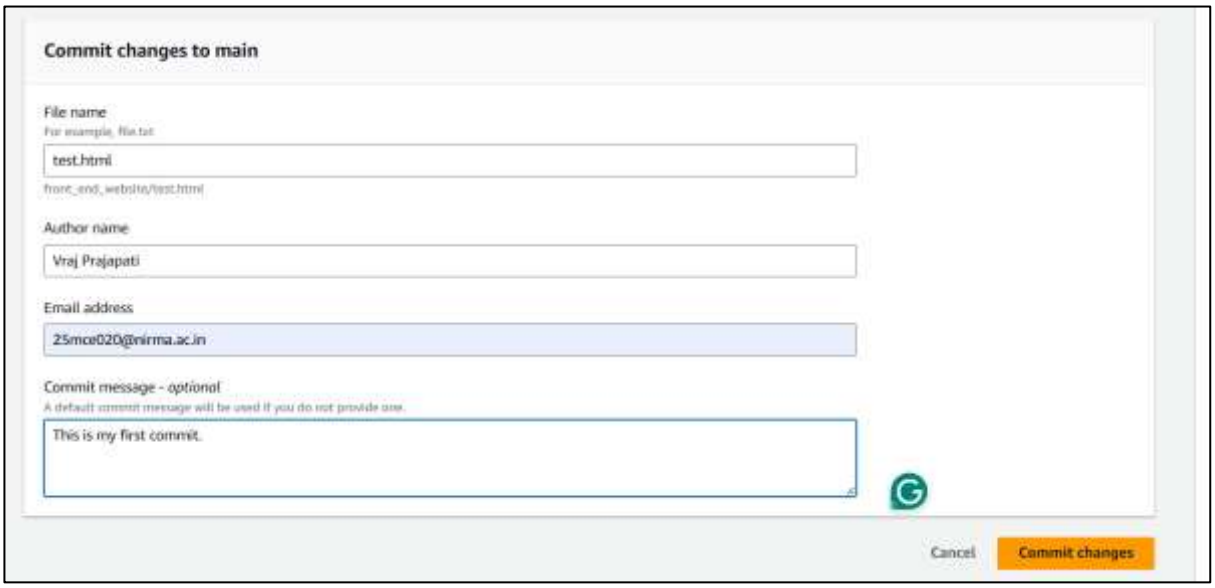
  **File name:** Enter test.html

  **Author name:** Enter your name.

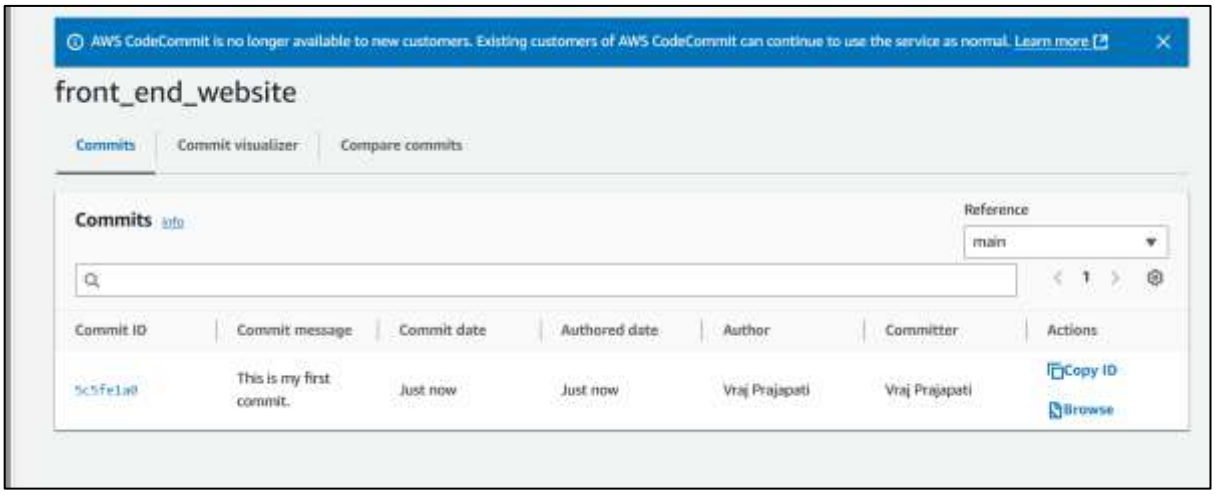
  **Email address:** Enter the same email address that you used in Task 1.

  **Commit message:** Enter This is my first commit.

  Choose **Commit changes**.

9. 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.




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.

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 VS Code IDE 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.

10. Return to the VS Code IDE.
11. In the Explorer section, expand *Environment*, which is located in the upper-left corner.
12. 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
}
```

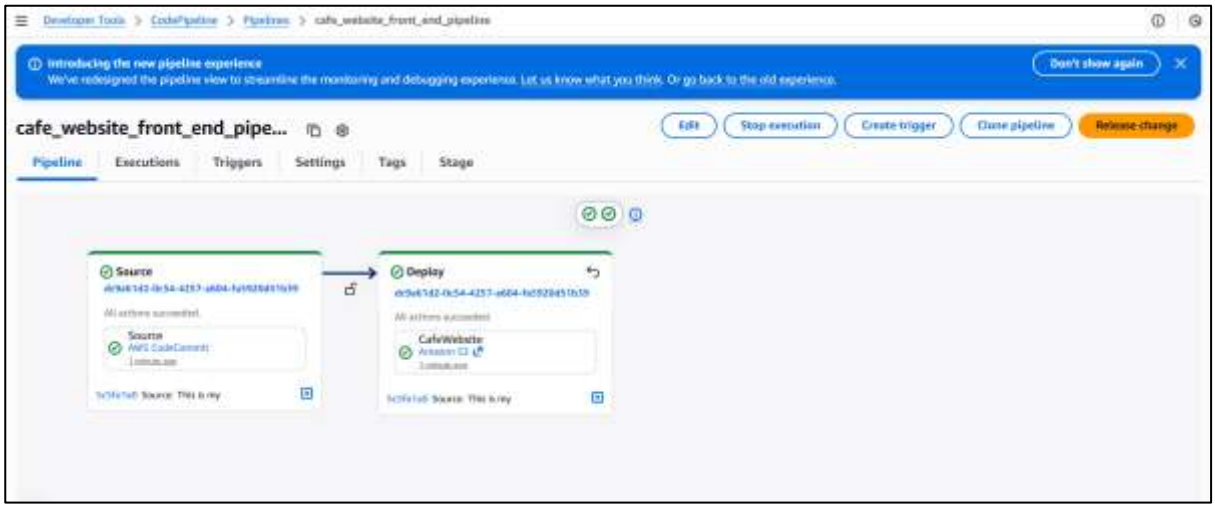

13. Update the cafe_website_front_end_pipeline.json file:
    - Replace the two *<FMI_1>* placeholders with the your AWS account ID.
    - Replace the *<FMI_2>* placeholder with the name of the bucket that has *s3bucket* in the name.
    - From the navigation pane, choose menu, then choose **File > Save**.
14. 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**

15. Navigate to the CodePipeline console.

    The **Pipelines** section lists the **cafe_website_front_end_pipeline** Pipeline.

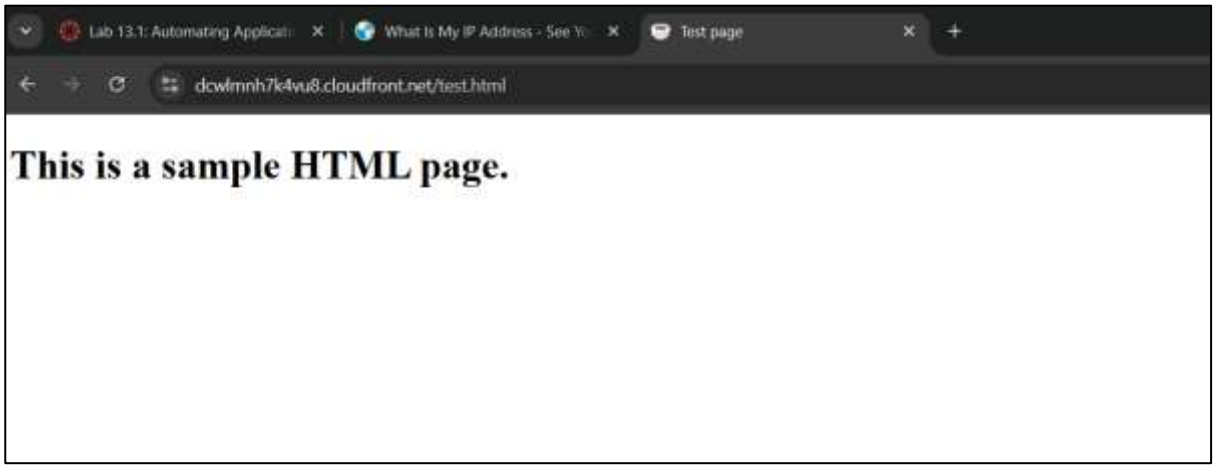

16. Choose the **cafe_website_front_end_pipeline** hyperlink and review the pipeline status, as shown in the following image.




    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.

17. 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://aaabbb111222.cloudfront.net/test.html.
- Open a new browser tab, and enter the URL that you just created.

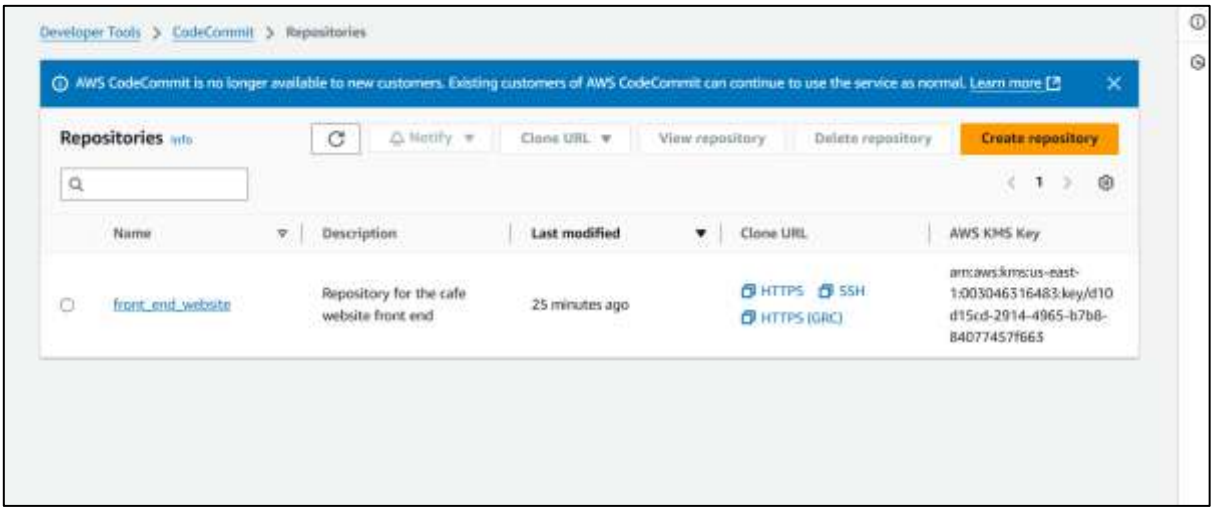



### ❖ Task 4: Cloning a repository in VS Code IDE

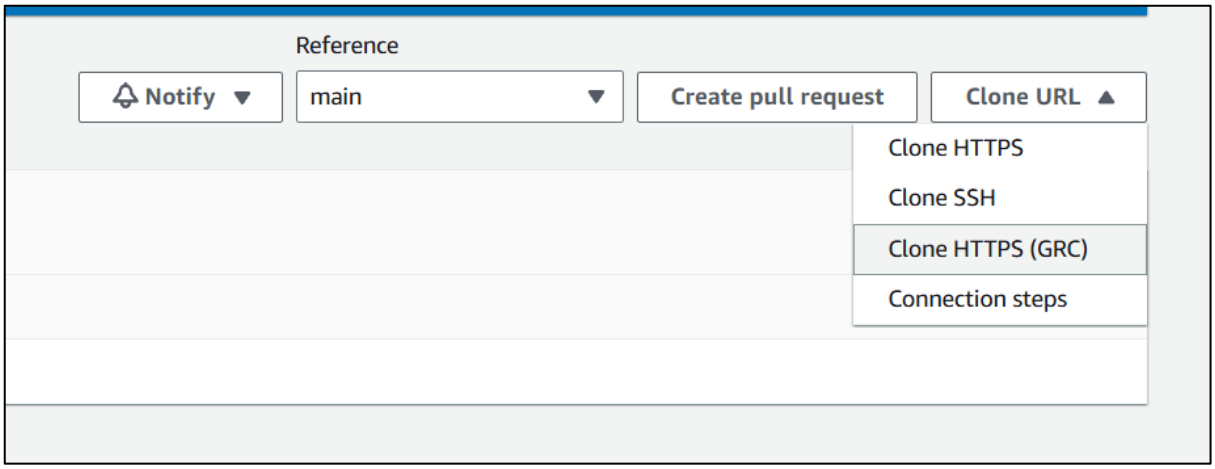
It's more efficient to edit code in an IDE than it is to use the CodeCommit console. In this task, you will clone a local copy of the repository in your VS Code IDE work environment.

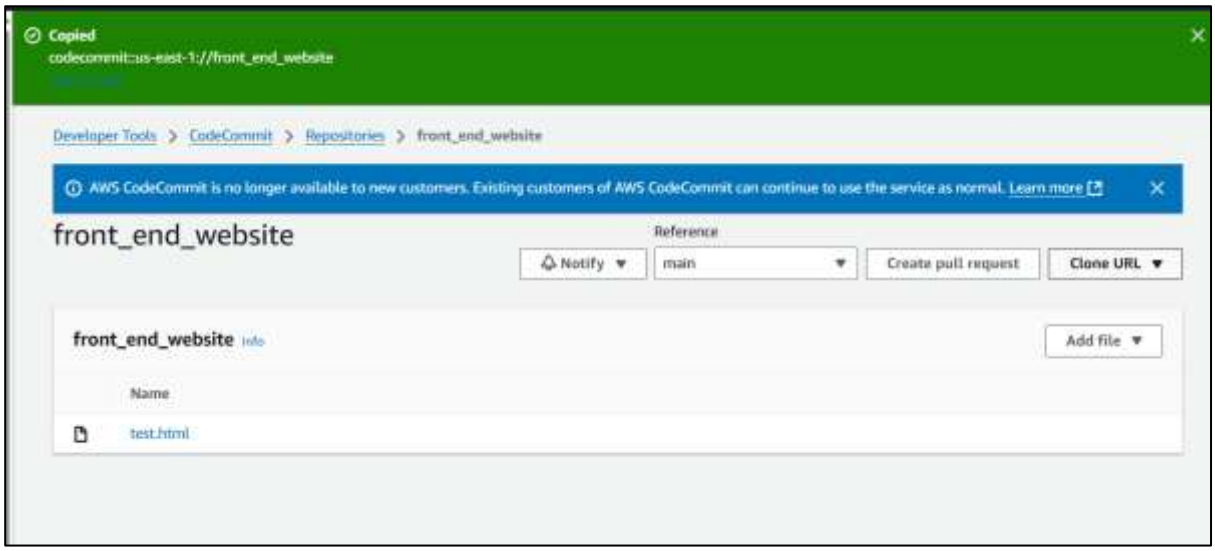
**18.** Retrieve the SSH clone URL for your repository**.**

- 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.

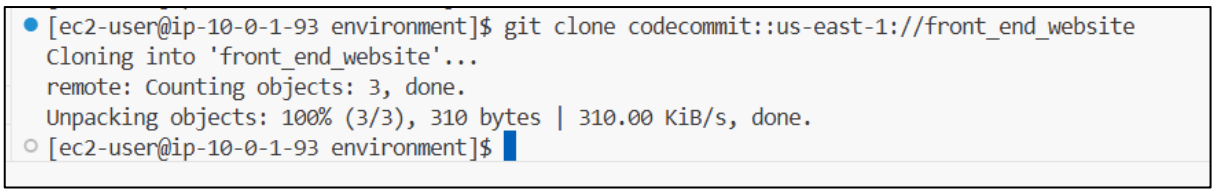



19. Clone your repository using the SSH URL.
    - ❖ Return to the VS Code IDE terminal.
    - ❖ Run the following command to clone the repository to your VS code IDE, replace the <<Clone URL>> with the value you copied.

      **cd ..**

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

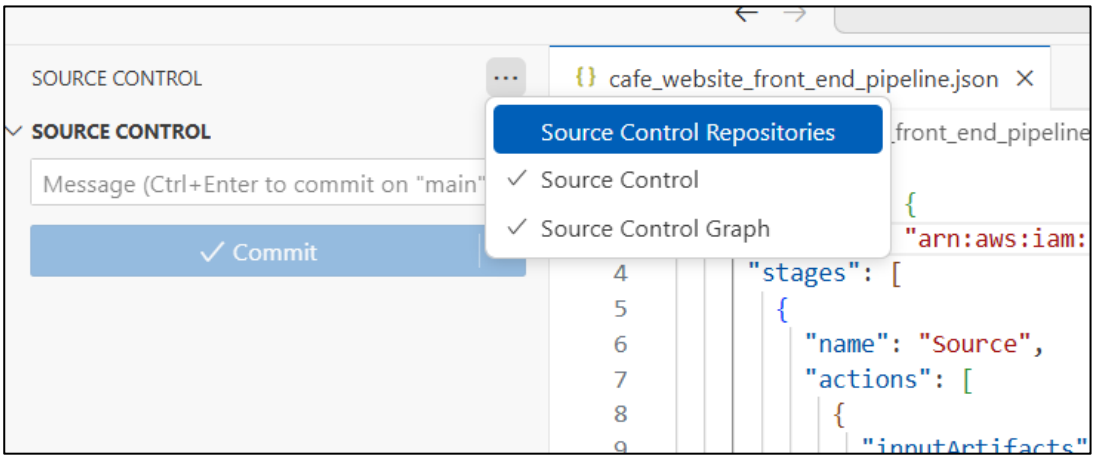


❖ **Task 5: Exploring the Git integration with the VS Code IDE**

You can interact with the repository through the command line, but VS Code 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.

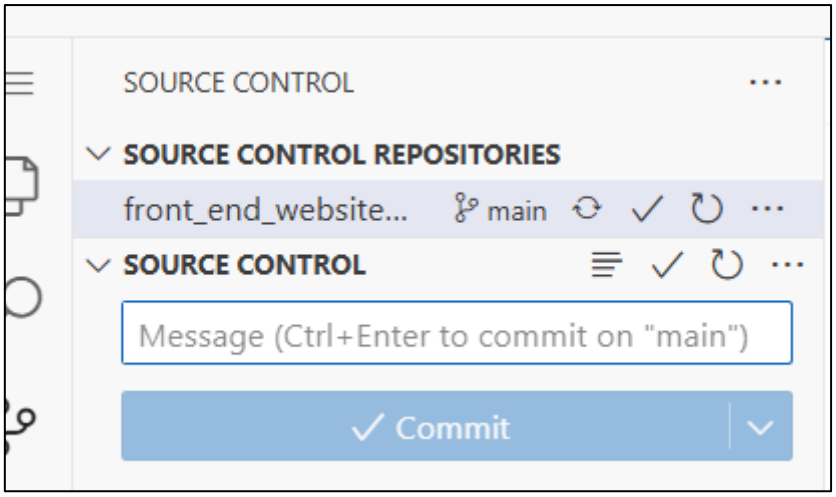
20. Explore repository branch management.

- 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.



21. Explore and edit the repository files.

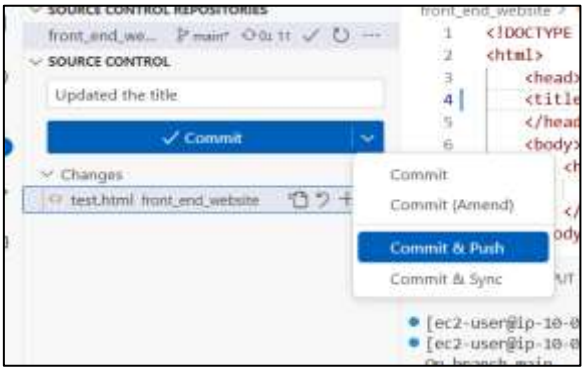
    Remember that the repository was cloned to the local folder environment/front_end_website. You can open repository files in your IDE to view and edit them.

    - 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:


    - 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.**

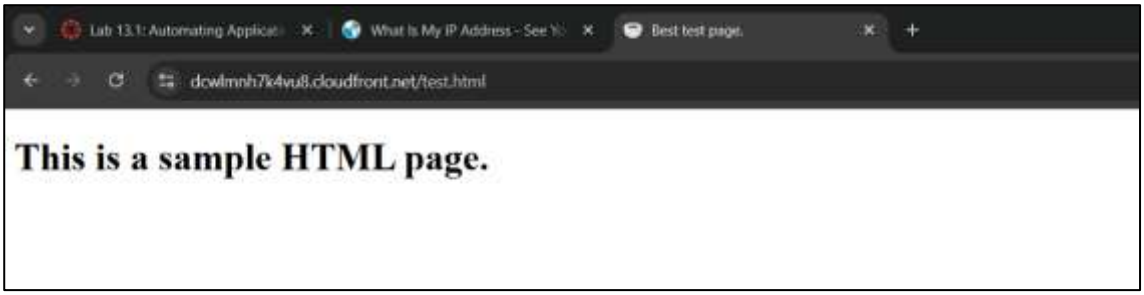

22. Commit your changes to the **main** branch.
    - Choose the  Branch icon to open the Source Control.
      **cd ~/environment/front_end_website**
      **git status**
    - In the Message text box, enter Updated the title as shown in the following image.
    - On the Commit button, choose  more 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.




23. 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.


24. 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.




❖ **Task 6: Pushing the café website code to CodeCommit**

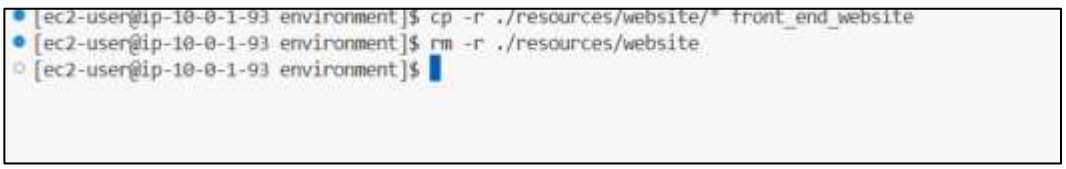
In this task, you will first remove the test.html file. Then, you will update the local repository with the café website code. Finally, you will verify that your pipeline built the café website on S3. You will also verify that *max-age* is set to 14 to confirm that the latest changes are being applied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

25. Return to the VS Code IDE tab.
26. On the explorer, Under *Environment* folder, expand the **front_end_website** folder and delete the **test.html** file, as shown in the following image.
27. 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.

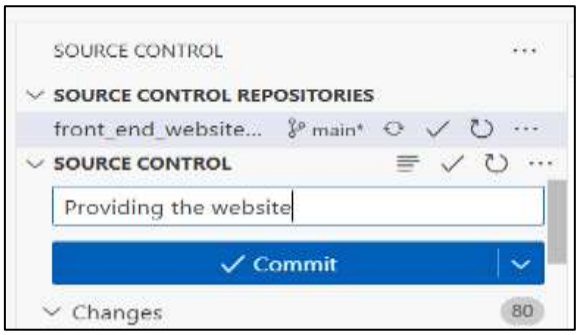
    **cd ~/environment**

    **cp -r ./resources/website/* front_end_website**
28. To delete the website folder, so that there is one source of truth, run the following command.
    **rm -r ./resources/website**




29. 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.

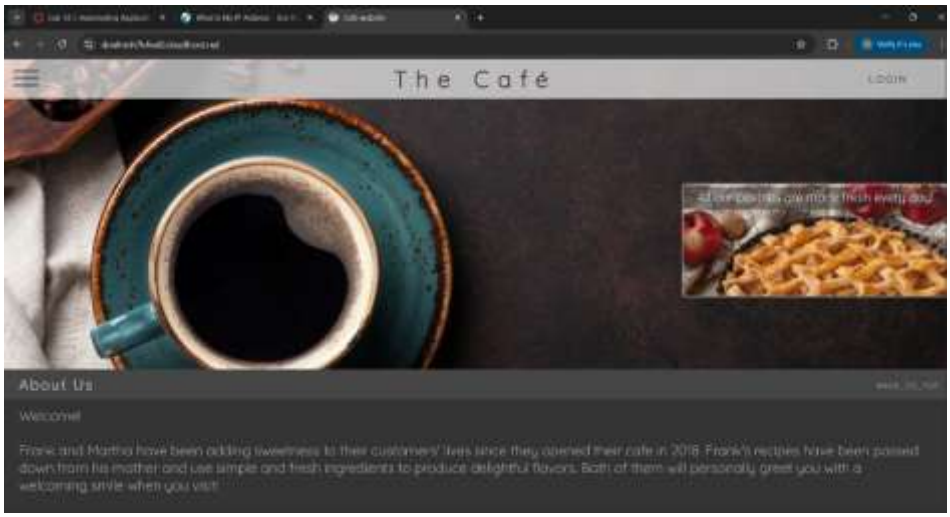
- To the right of front_end_website, choose the options icon and choose Commit.



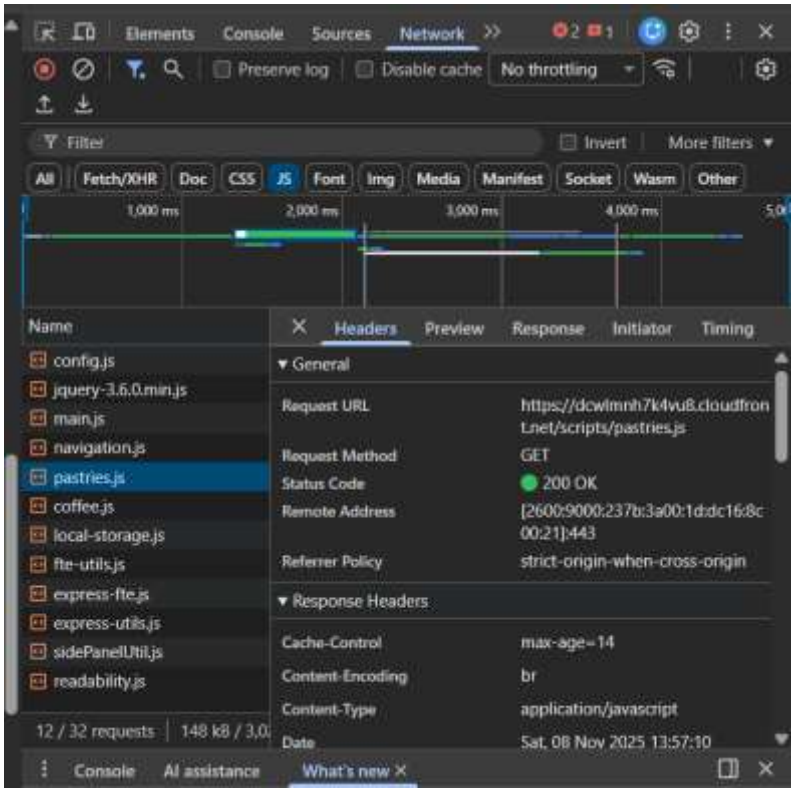

- The Source control Graph in the lower pane shows history of changes done to the repository.


30. Return to the browser tab that shows the test.html page.
31. 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



32. 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.
    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.
    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.




❖ **Conclusion:**

This experiment successfully demonstrated the implementation of an automated DevOps pipeline on AWS using Infrastructure as Code (IaC). By integrating AWS services such as CodeCommit, CodePipeline, and S3, infrastructure provisioning and continuous deployment were automated, ensuring efficient, version-controlled, and repeatable updates to the café website.