Set up a cloud development environment.

Use the Code Editor integrated development environment (IDE) in Amazon SageMaker Studio. Debug Python code and push changes by using GitLab self-managed on Amazon EC2.

In this lab we will be working on **Cloud9, CodeCommit, Sagemaker and S3.**

Practice lab:

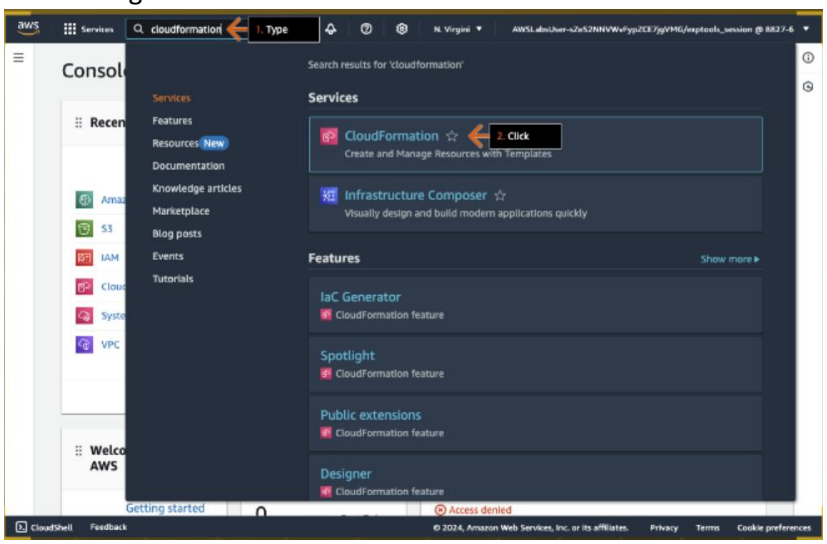
In this practice lab, you will:

- Explore Code Editor in Amazon SageMaker Studio.

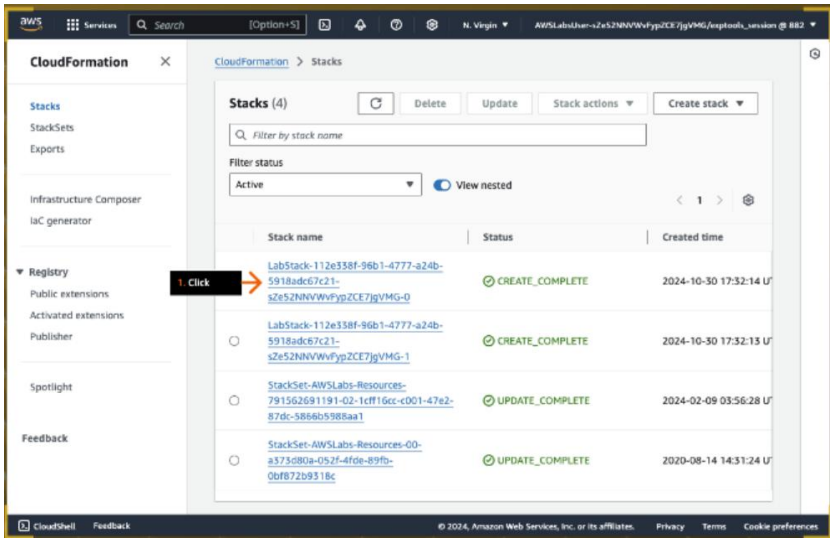
- Debug Python code by using the Code Editor IDE.

- Push changes to GitLab self-managed on Amazon EC2.

1. In the navigation bar search CloudFormation.

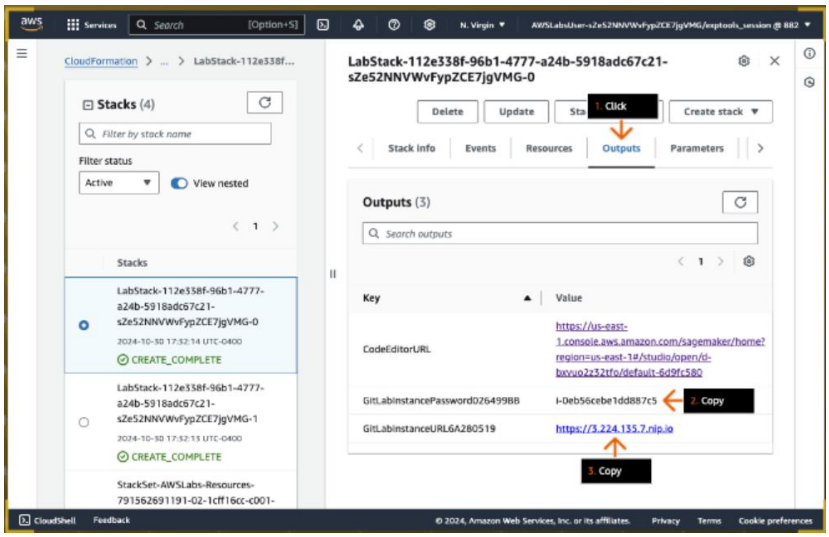


2. In the Stacks section, click the stack name that starts with LabStack and ends with -0.



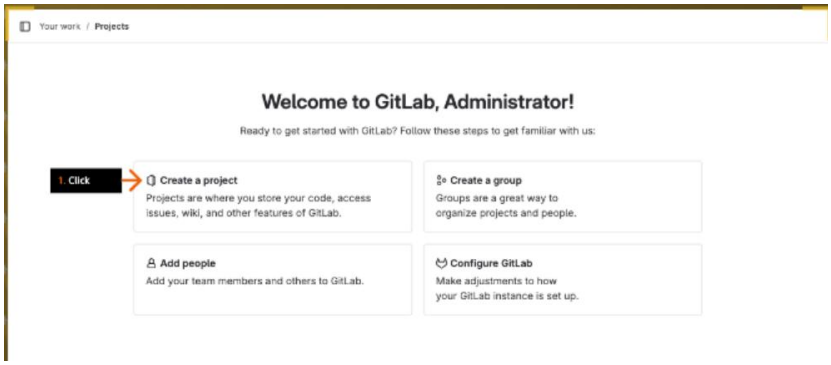
3.

1. Click the Outputs tab.
2. For the GitLabInstancePassword key, under Value, select (highlight) and copy the provided value, and then paste it in the text editor of your choice on your device.
3. Repeat this for the GitLabInstance

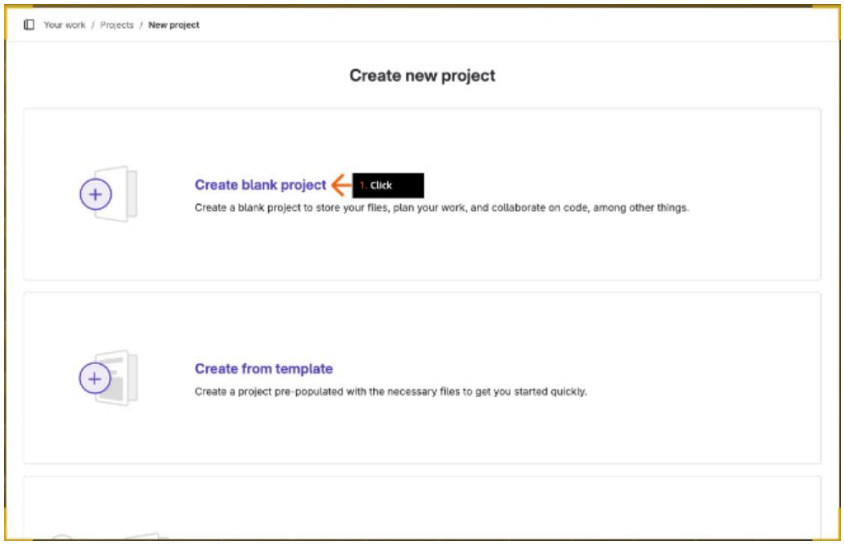


4. In the next steps, you create a new code repository in GitLab self-managed on Amazon EC2.
1. In a new browser tab address bar, paste the GitLabInstanceURL that you copied in an earlier step and press Enter (not shown).
2. For Username or primary email, type:
root
3. For Password, paste the GitLabPassword value that you copied earlier.
4. Click Sign in.

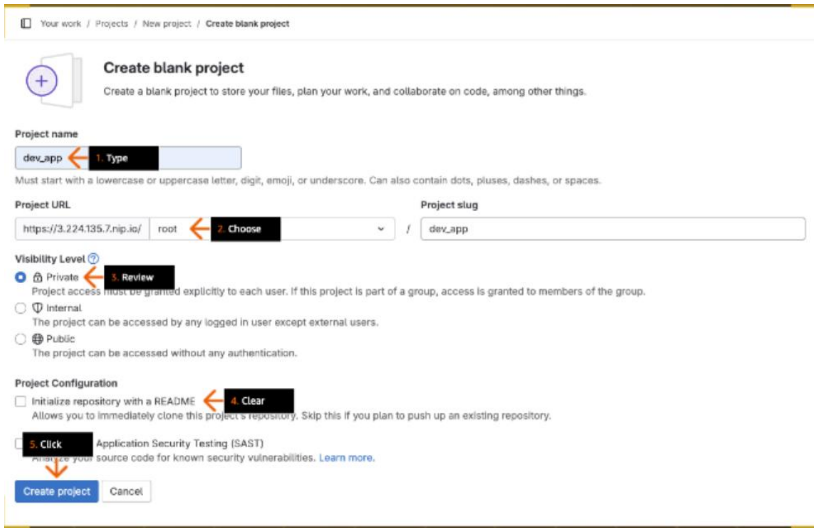
5. Click Create a project.



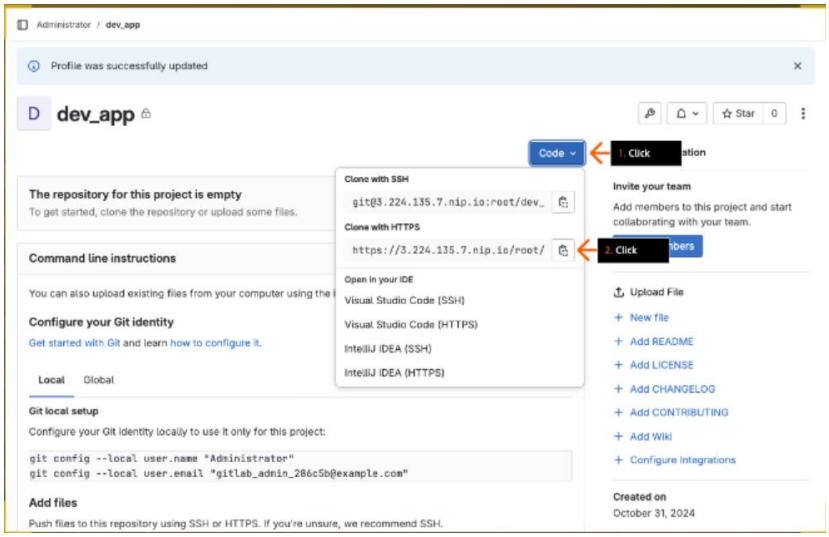
6. Click Create blank project.



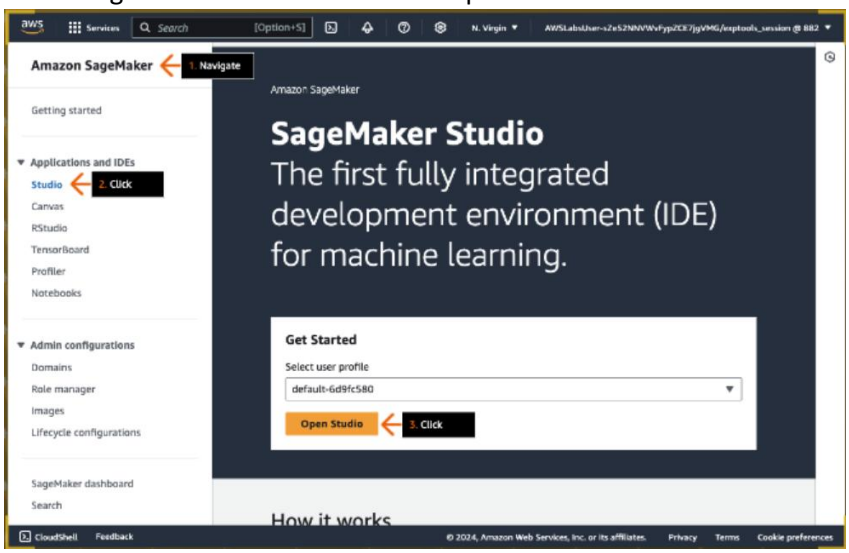
7. 1. For Project name, type:
   dev_app
   2. For Project URL, on the dropdown list, choose root.
   3. For Visibility Level, review to confirm that Private is selected.
   4. For Project Configuration, clear the check box to deselect Initialize repository with a README.
   - Make sure you uncheck this option.
   5. Click Create project.



8. 1. On the project page, click Code to expand the menu.
   2. Under Clone with HTTPS, click the copy icon to copy the URL, and then paste it in your text editor.

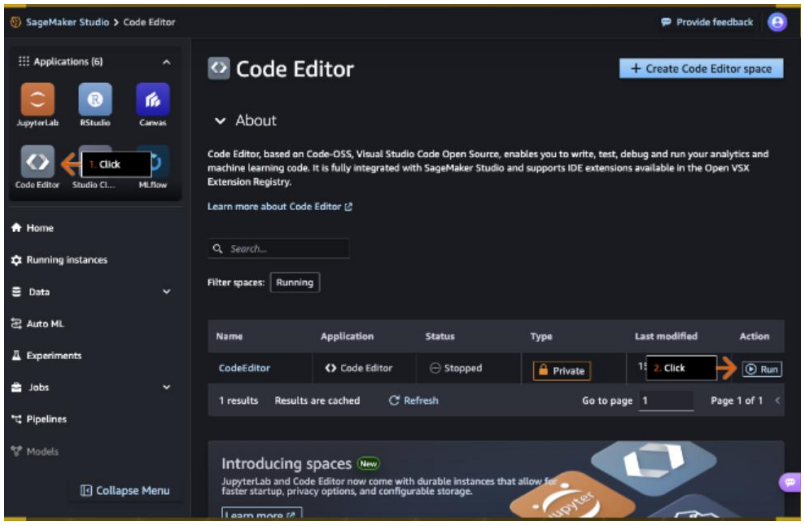


9. 1. In the previous browser tab, navigate to the Amazon SageMaker console.
   - Remember, on the top navigation bar, you can use the Services search box (or click Services) to navigate to a different service console.
   2. In the left navigation pane, under Applications and IDEs, click Studio.
   3. On the Amazon SageMaker Studio home page, click Open Studio.
   - SageMaker Studio opens in a new browser tab (or window). Keep the current browser tab open. You return to the SageMaker console in a later step.

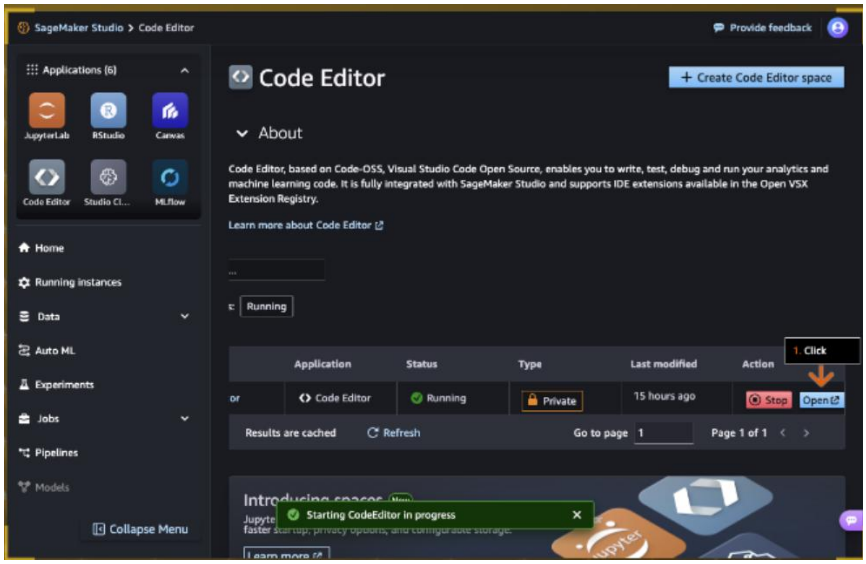


10. If the Welcome pop-up box appears, click Skip Tour for now.
   - You are welcome to take the tour, but it is not covered in this practice lab.

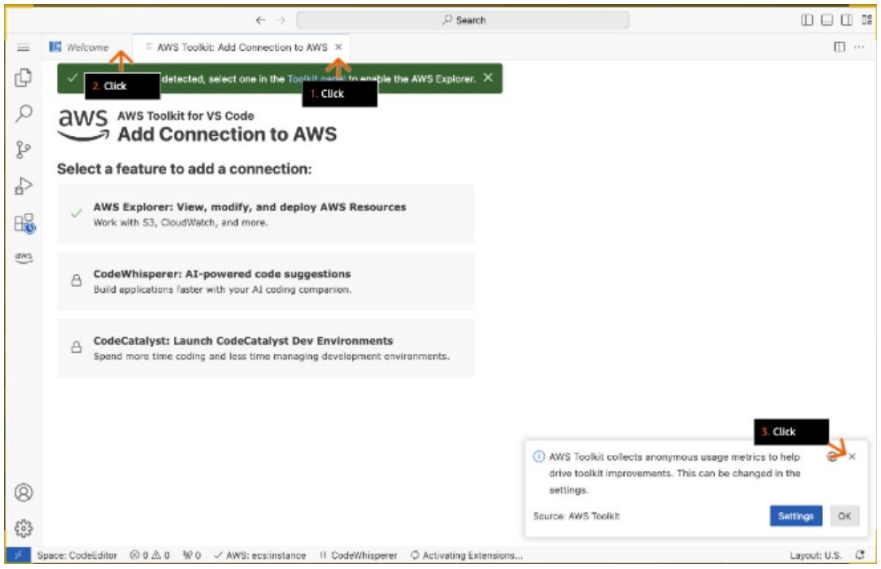
11. 1. In the Applications pane, click Code Editor.
   2. On the Code Editor page, for the Code Editor application, under Action, click Run.
   - The Code Editor application takes few minutes to start.



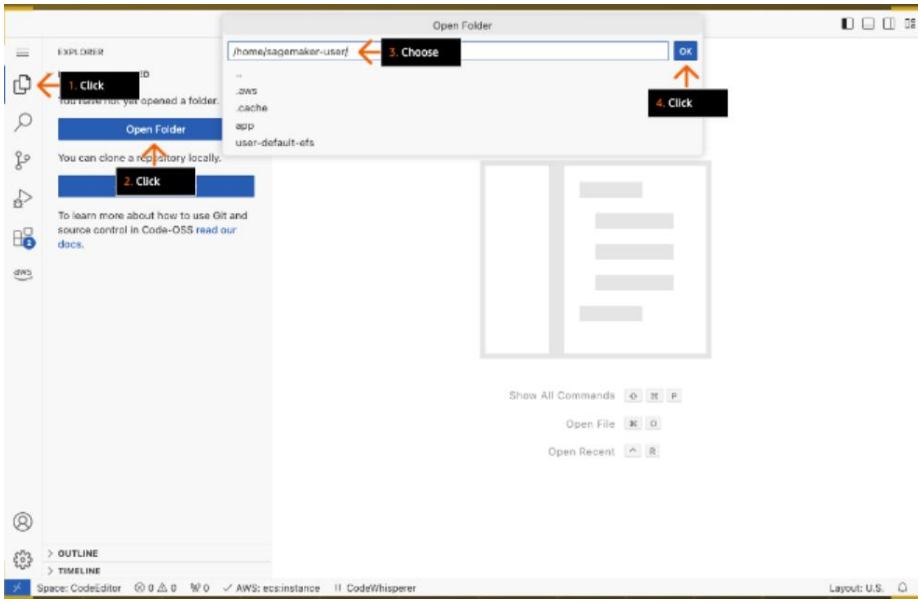
12. Under Action, click Open.
    - The Code Editor application opens in a new browser tab (or window).



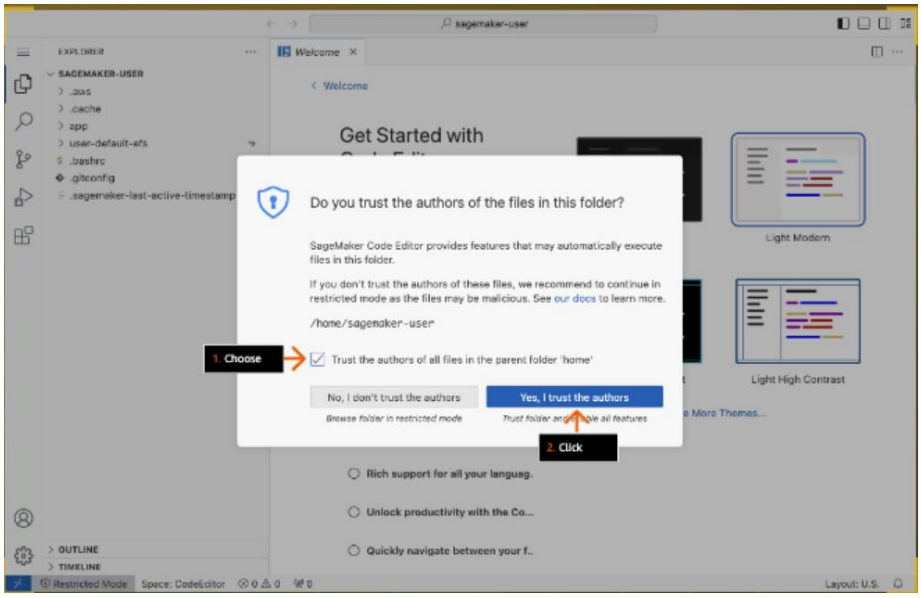
13. - On the new Welcome tab, you have an option to choose a theme, which defines your overall IDE colors (not shown). You are welcome to choose a theme.
    1. To close the AWS Toolkit for VS Code tab, click the X.
    2. On the Welcome tab, review the content, and then click the X to close the tab.
    3. Click the X to close the AWS Toolkit information alert.



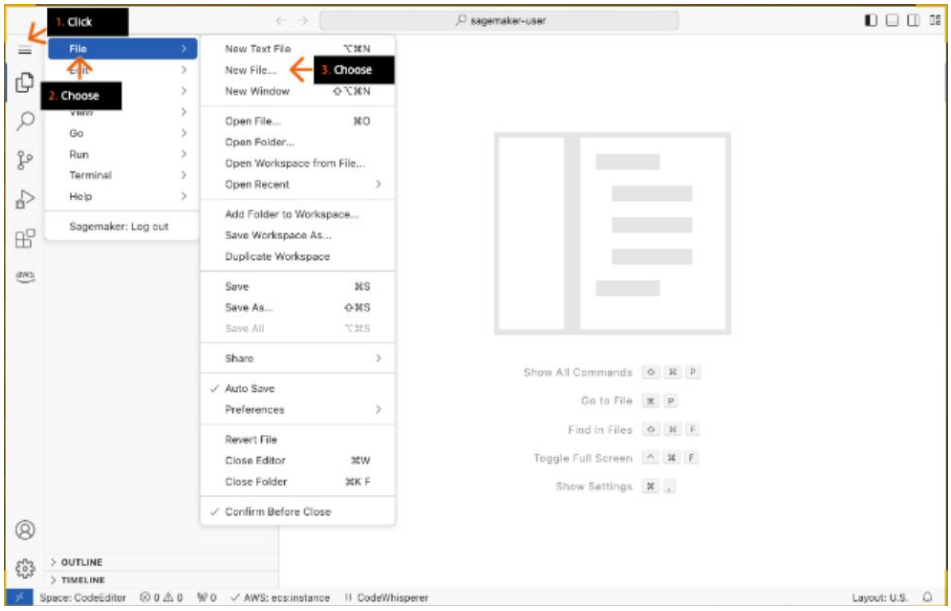
14. 1. To view the Explorer window, in the left sidebar, click the Exploration (two folders) icon.
    2. In the Explorer window, click Open Folder.
    3. In the Open Folder search box, choose /home/sagemaker-user/.
    4. Click OK.



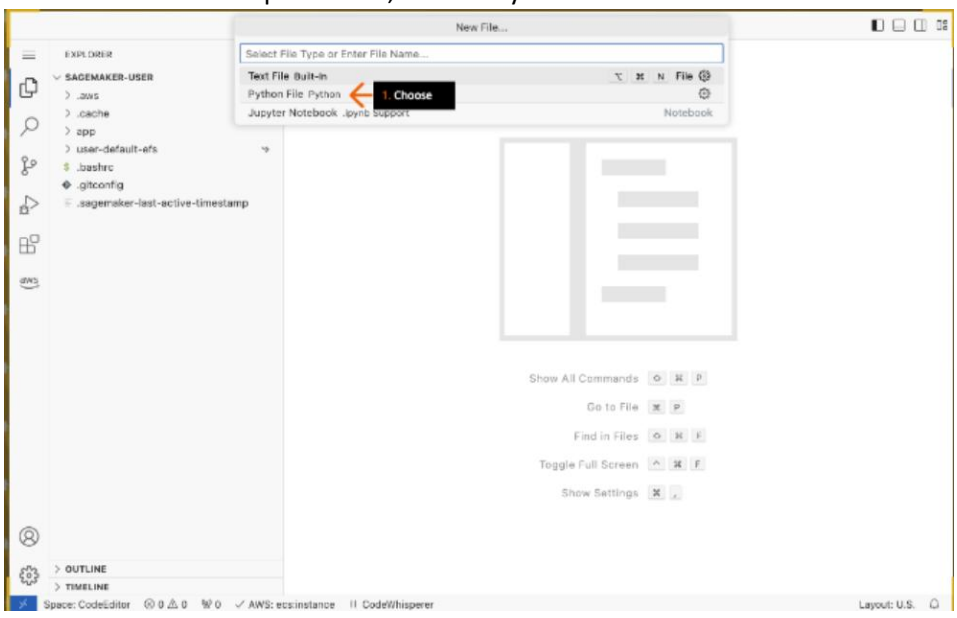
15. 1. In the pop-up box, choose the check box to select Trust the authors of all files....
    2. Click Yes, I trust the authors.



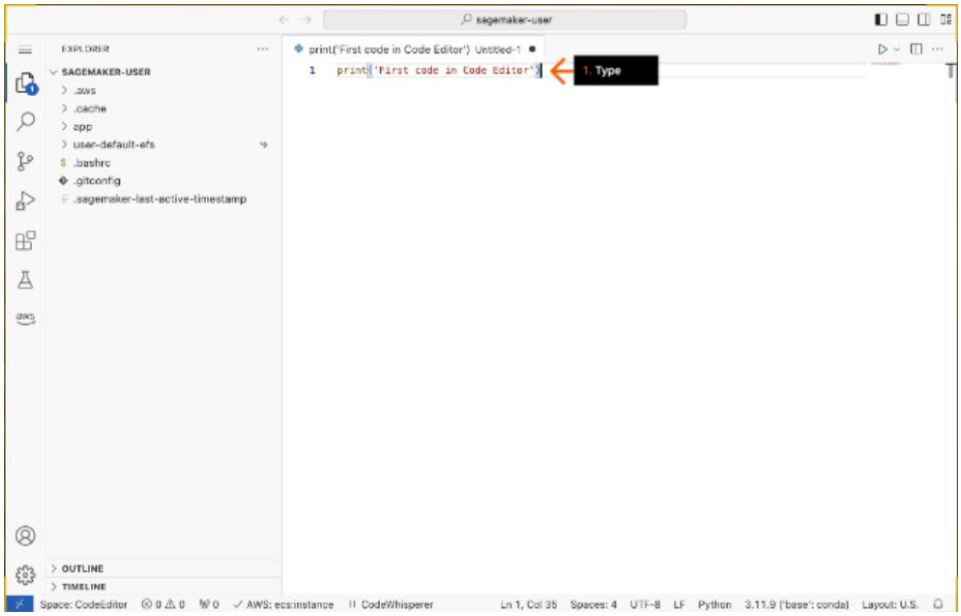
16. 1. In the left sidebar, click the menu icon (three lines) to expand the menu.
    2. Choose File.
    3. Choose New File...



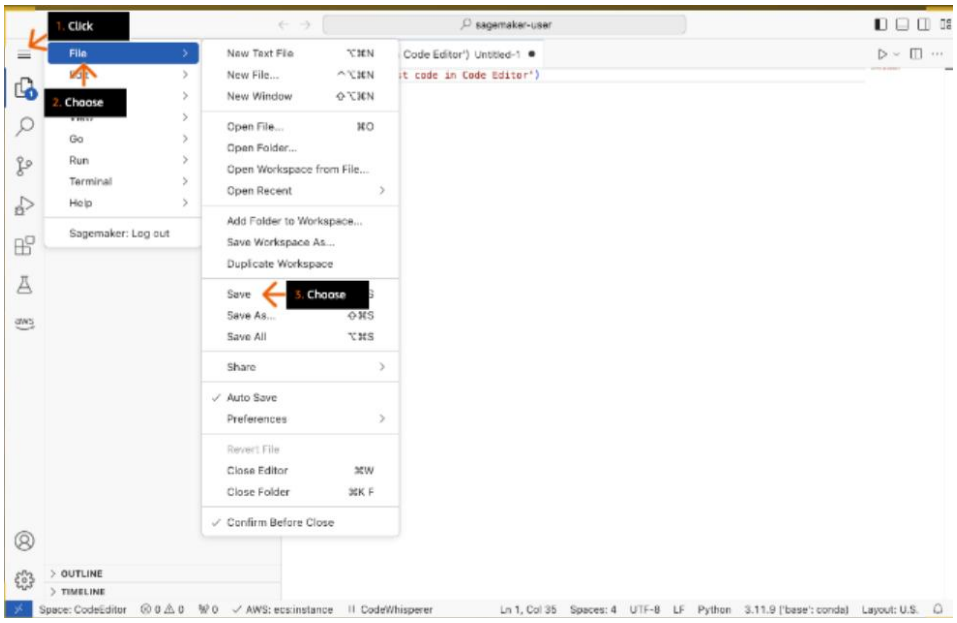
17. On the New File... dropdown list, choose Python File.



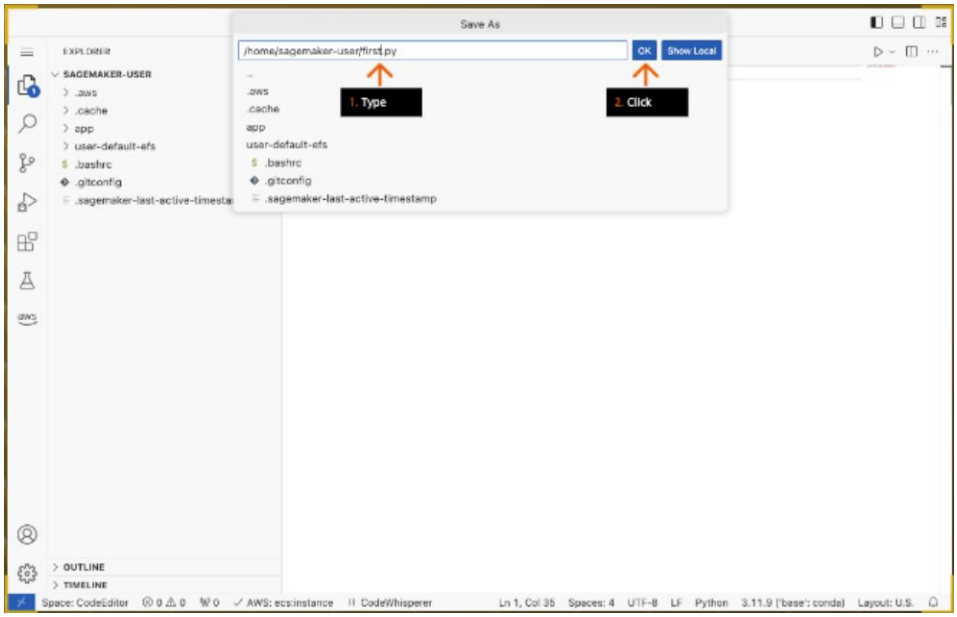
18. In the Python code window, type:
print('First code in Code Editor')



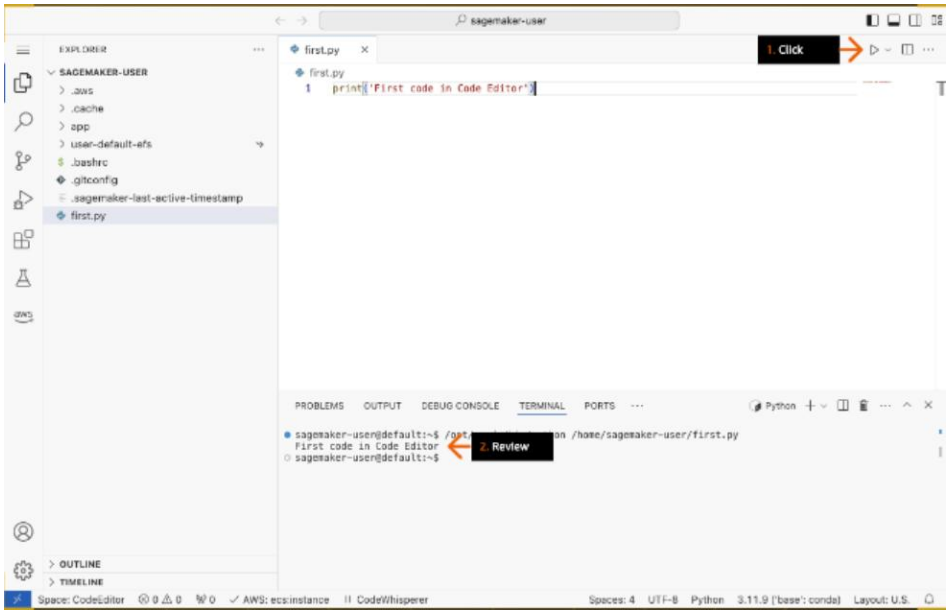
19. 1. In the left sidebar, click the menu icon (three lines) to expand the menu.
    2. Choose File.
    3. Choose Save.



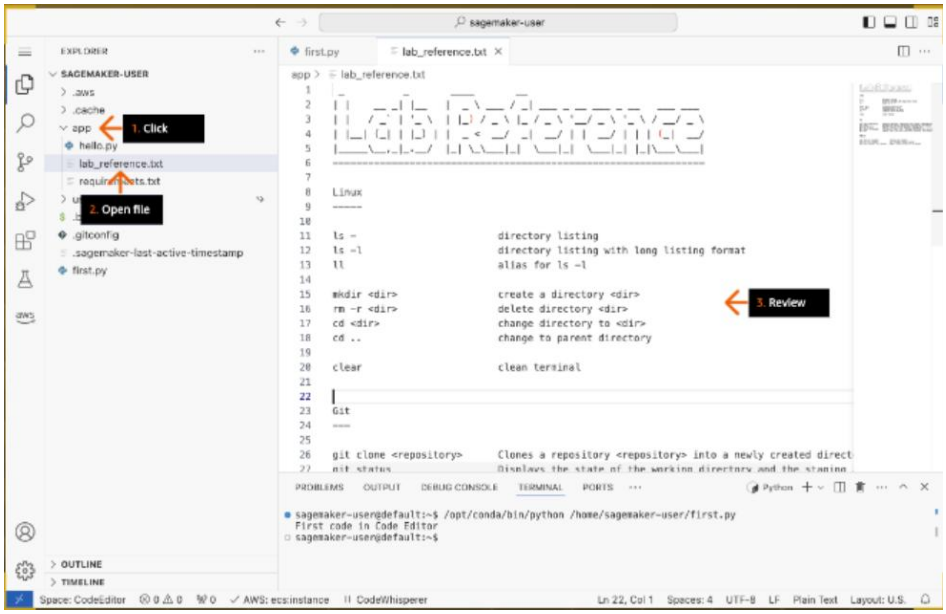
20. 1. In the Save As box, for the file name, type:
    first.py
    2. Click OK.



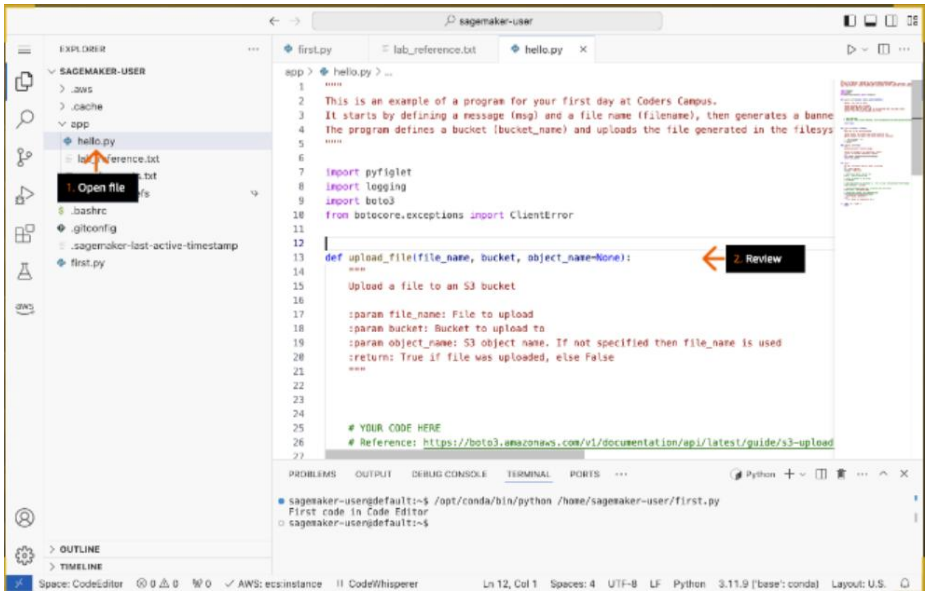
21. 1. On the top navigation bar, click the Run icon.
    2. In the bottom terminal window, review the output.



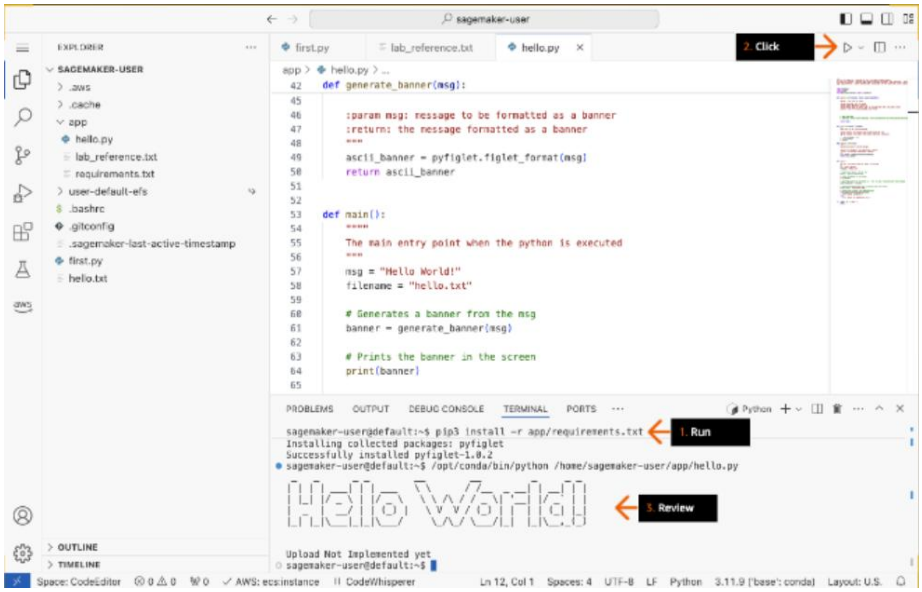
22. 1. In the left Explorer window, click to expand the app folder.
    2. Open (double-click) the lab_reference.txt file.
    3. In the top lab_reference window, review the file contents.
    - Basic "linux", "git" and, "aws cli" commands, that you might need in this lab, are displayed.



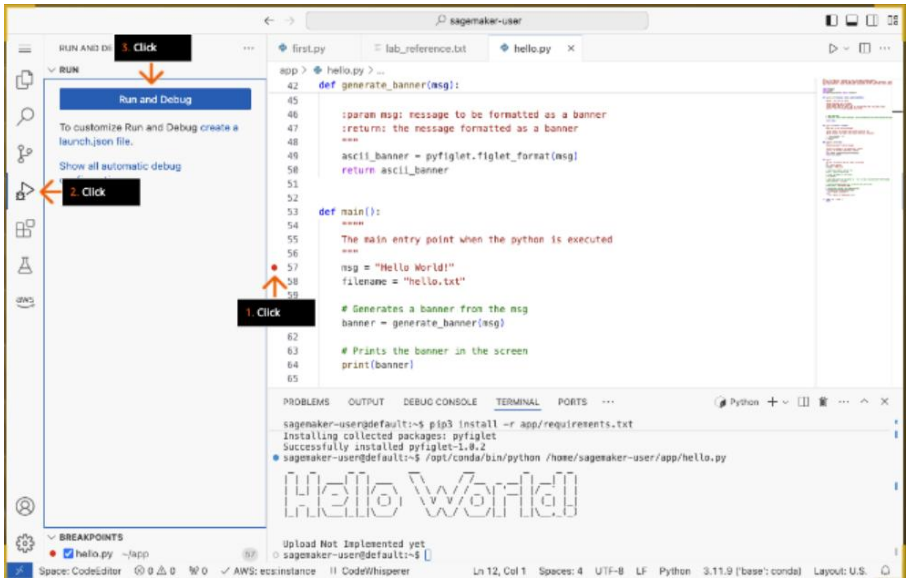
23. 1. In the left Explorer window, open (double-click) the hello.py file.
    - The file opens in a new tab.
    2. Review the code, and try to understand the flow.
    - The main function execution starts on line 53: def main().



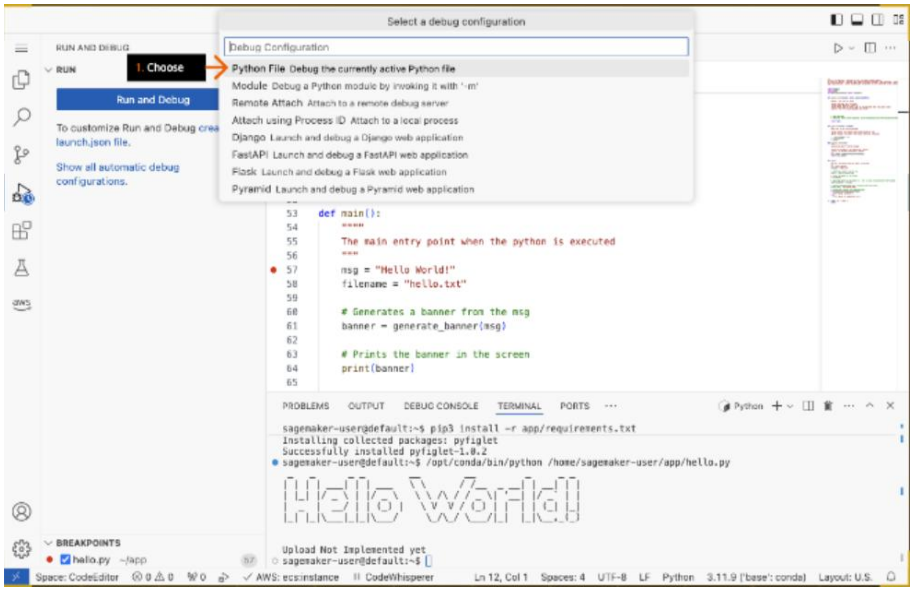
24. 1. To install the required Python modules, in the bottom terminal window, at the command prompt, run (type the command and press Enter):
pip3 install -r app/requirements.txt
- If you copy-paste this command in the terminal, Code Editor will prompt you to share a clipboard. Click Allow to paste the command.
- Python 3 is used to run hello.py in this lab.
2. On the top navigation bar, click the Run icon.
3. In the bottom terminal window, review the output results.



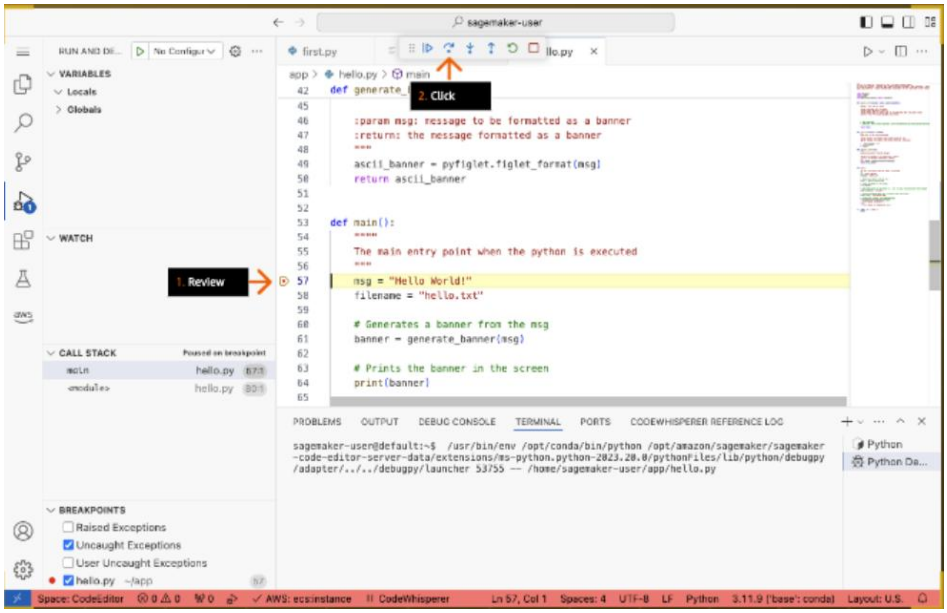
25. 1. In the top hello.py window, to add a breakpoint, click in the left gutter at line 57.
2. In the left sidebar of the IDE, click the Run and Debug icon.
3. Click Run and Debug.



26. On the Select a debug configuration dropdown list, choose Python File.

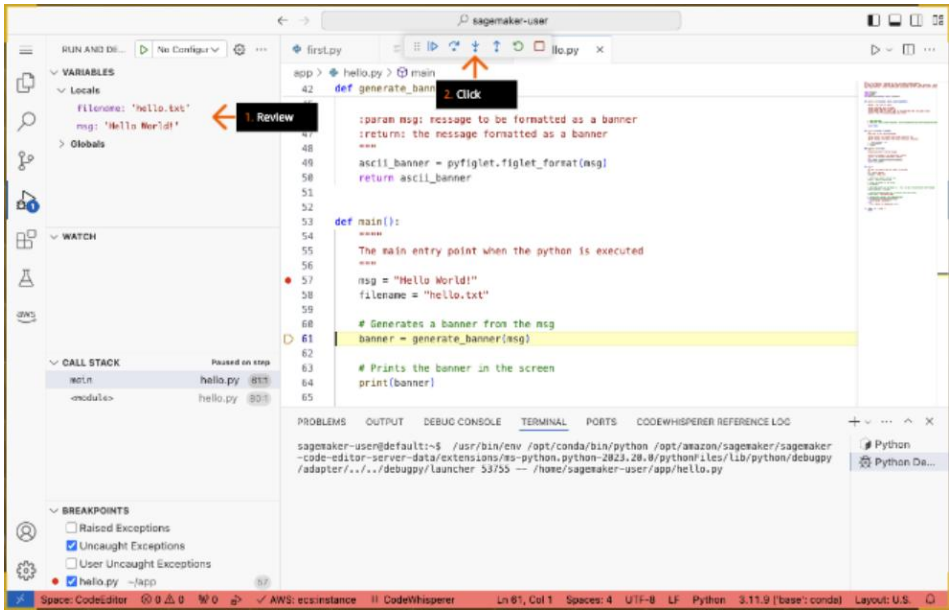


27. 1. Review to see that the breakpoint is now highlighted.
    2. To step through the code until you reach line 61, on the top floating debug toolbar, click the Step Over icon.

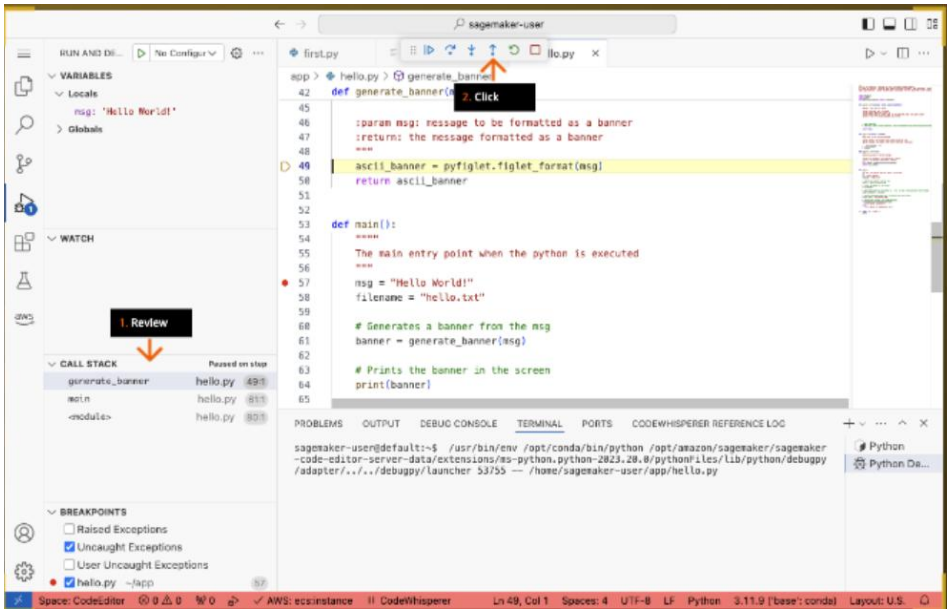


28. 1. In the left RUN AND DEBUG pane, under VARIABLES, review the values.

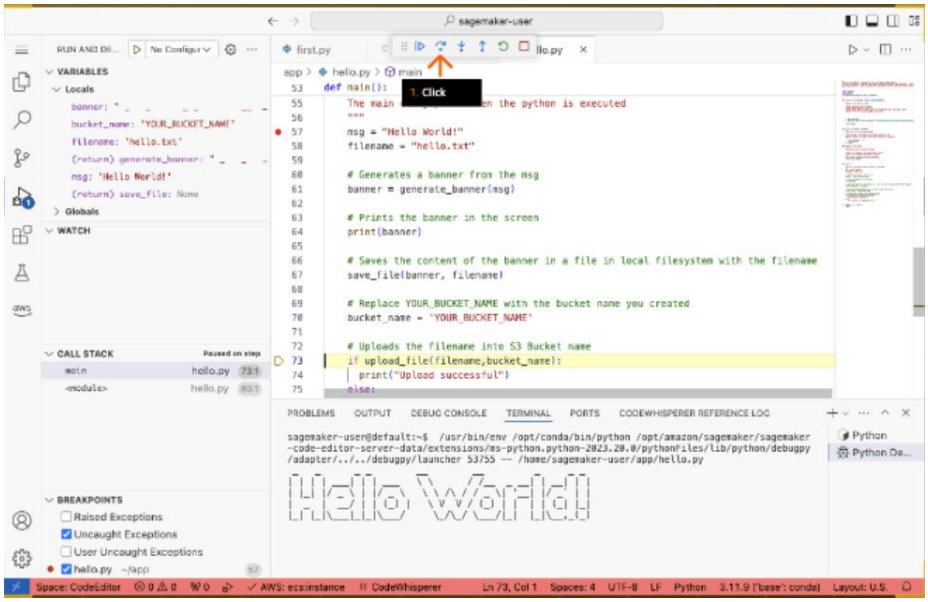
2. At line 61 in the hello.py code window, on the top floating debug toolbar, click the Step Into icon.



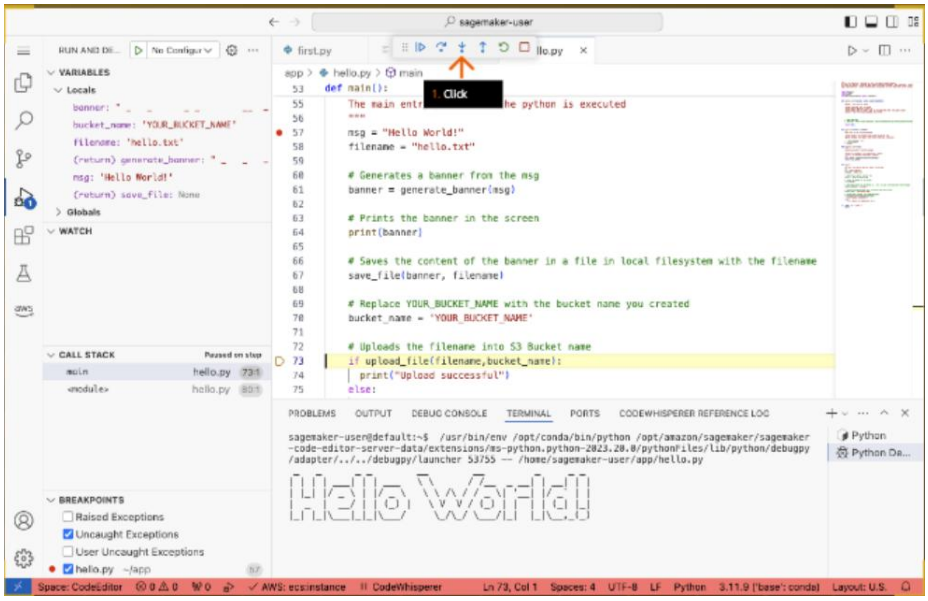
29. 1. Under CALL STACK, review the function details.
- These details show the execution stack. In this lab, the generate_banner() function was called inside the main() function on line 61.
2. On the top floating debug toolbar, click the Step Out icon.



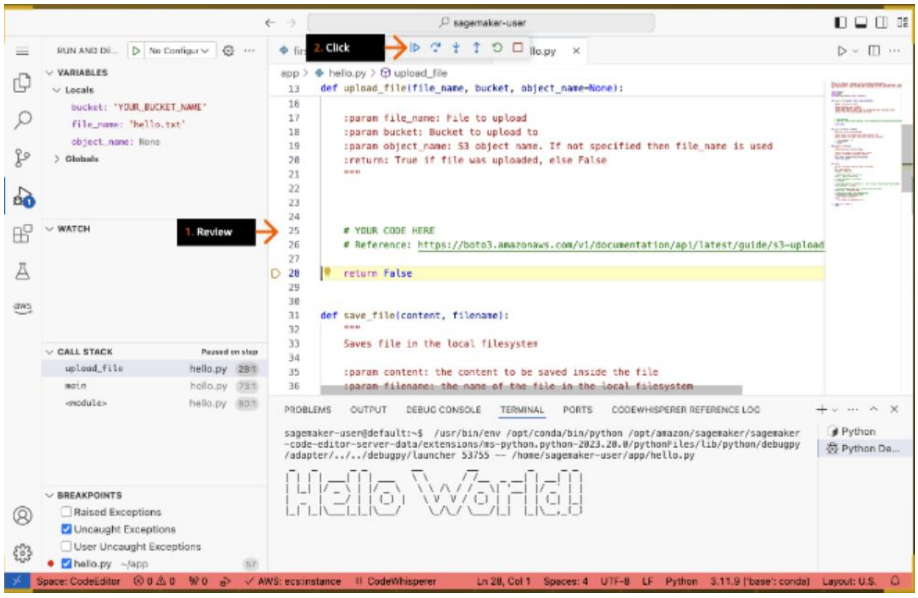
30. On the top floating debug toolbar, click the Step Over icon until line 73.



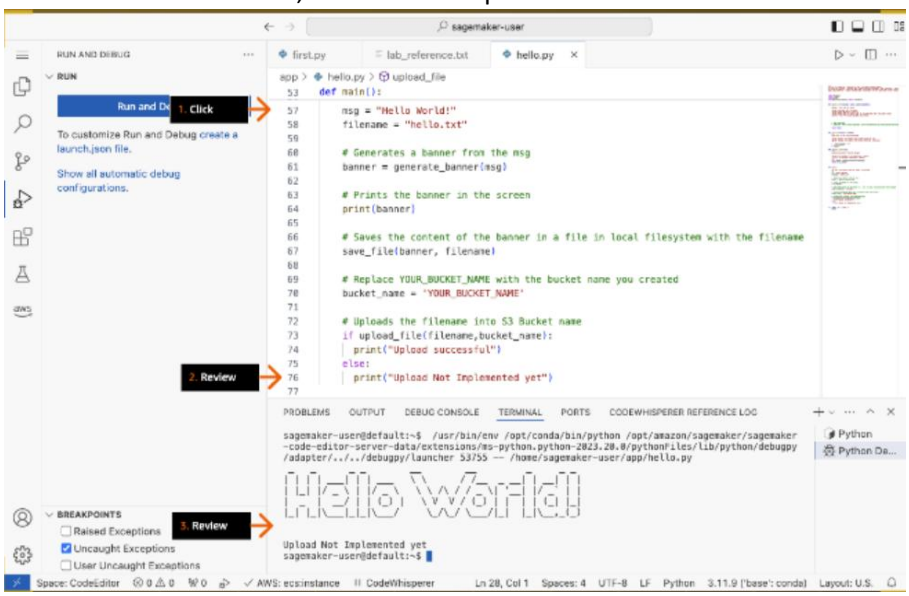
31. At line 73 in the hello.py code window, on the top floating debug toolbar, click the Step Into icon.



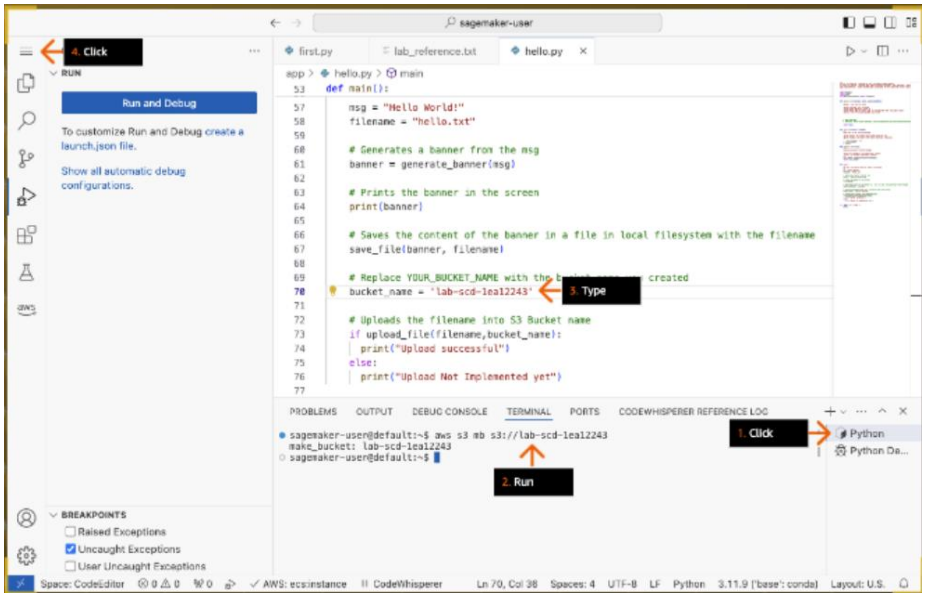
32. 1. On line 25, review the comment, YOUR CODE HERE.
- This method has not been implemented yet. Later, you develop the upload file code yourself. For now, you can improve the code by printing a message about the unimplemented upload function.
2. On the top floating debug toolbar, click the Continue icon.



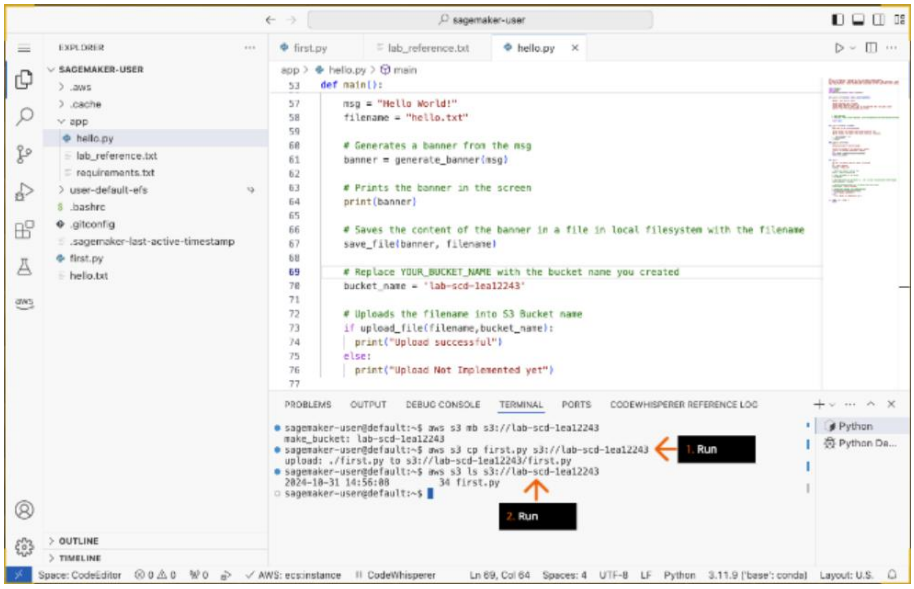
33. 1. At line 57, to delete the set breakpoint, click in the left gutter.
    2. On line 76, review the statement: print("Upload Not Implemented Yet")
    3. In the bottom window, review the output result.



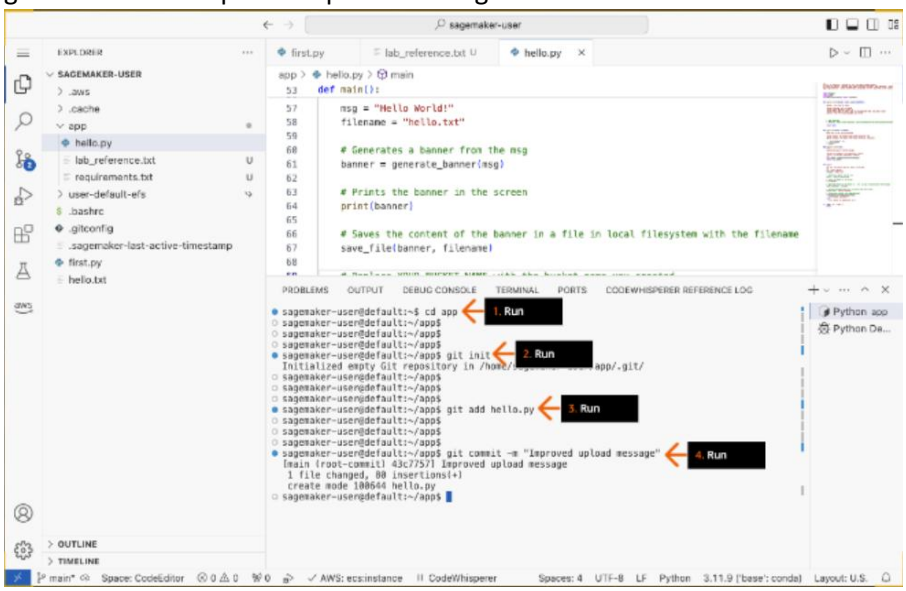
34. 1. To create an S3 bucket with an AWS CLI command, on the right side of the terminal window, click Python.
    2. In the terminal, run:
    aws s3 mb s3://[YourBucketName]
    3. In the top hello.py code window, on line 70, to replace Your_Bucket_Name, type the name of the S3 bucket that you just created.
    4. In the left sidebar of the IDE, click the menu icon (three lines) to expand the menu.
    5. Choose File, and then choose Save (not shown).



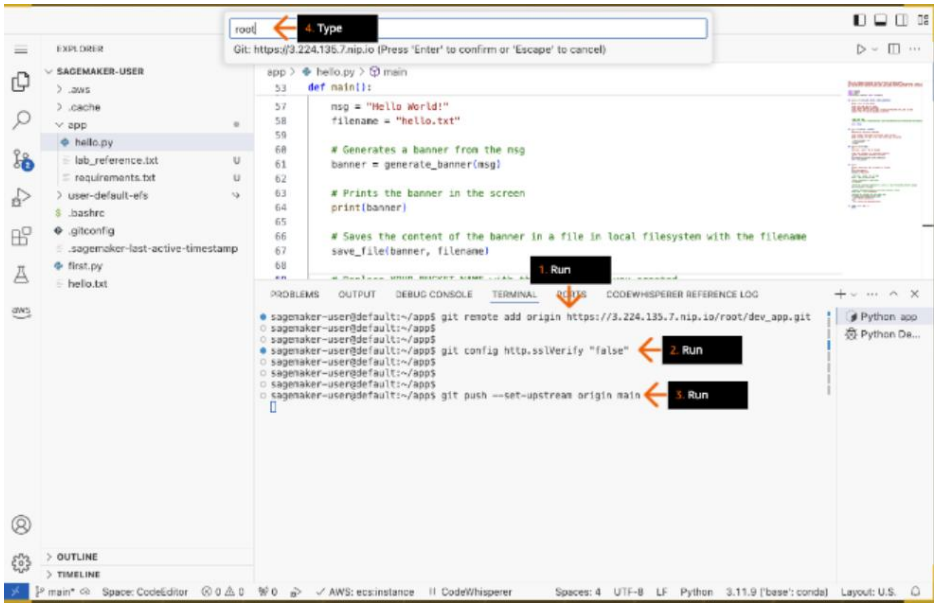
35. 1. To copy the first.py file to your S3 bucket, in the bottom terminal window, replacing the placeholder with your bucket name, run:
    aws s3 cp first.py s3://[your bucket name]
    2. To list the objects in your bucket, replacing the placeholder with your bucket name, run:
    aws s3 ls s3://[your bucket name]



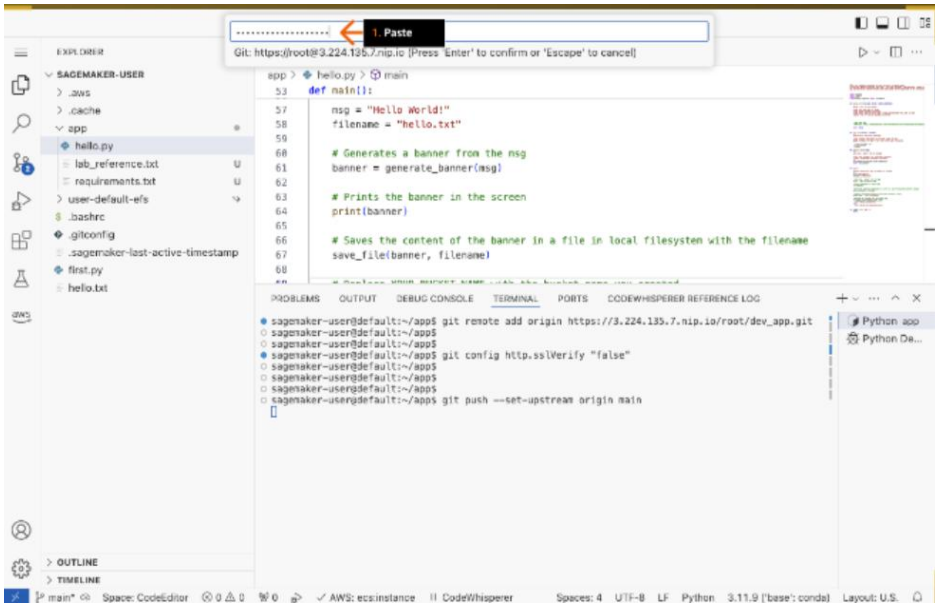
36. 1. To change to the app directory, run:
cd app
2. To initiate a git repository in the app folder, run:
git init
3. To add hello.py to the staging area, run:
git add hello.py
4. To commit the code to the repository with a comment that describes the code change, run:
git commit -m "Improved upload message"



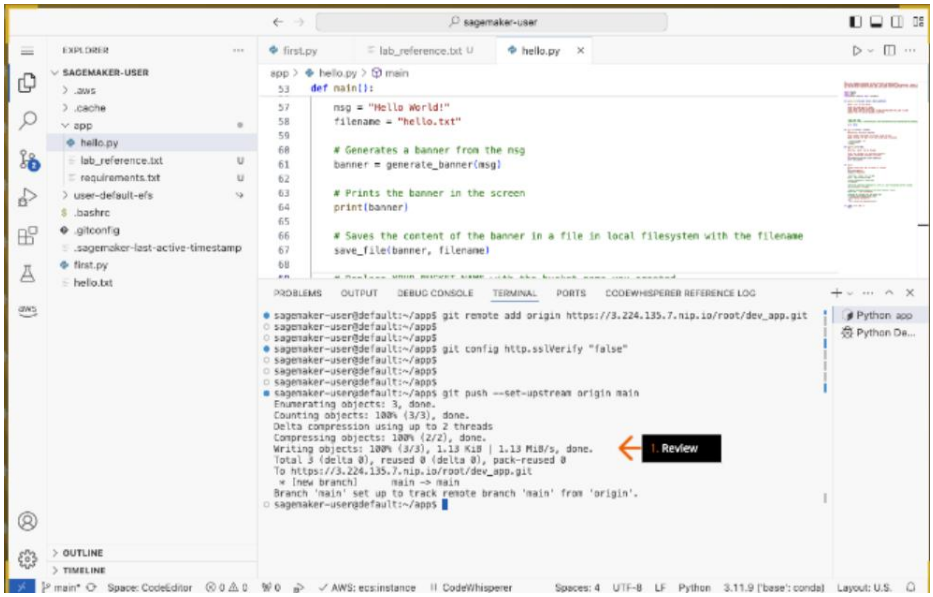
37. 1. To add the remote repository, dev_app, that you created earlier on GitLab, replacing the <GIT_REPO_URL> placeholder with the HTTPS repository URL, run:
git remote add origin <GIT_REPO_URL>
2. To configure Git to not verify the SSL certificate when fetching or pushing over HTTPS, run:
git config http.sslVerify "false"
- In this practice lab environment, the GitLab self-managed is using a self-signed certificate. This configuration should not be used in a production environment.
3. To push the code to the Git repository, run:
git push --set-upstream origin main
4. In the pop-up box at the top of the IDE, to provide credentials to access the Git repository, type:
root
and press Enter.



38. In the pop-up box, paste the GitLabPassword that you copied in earlier step and press Enter.

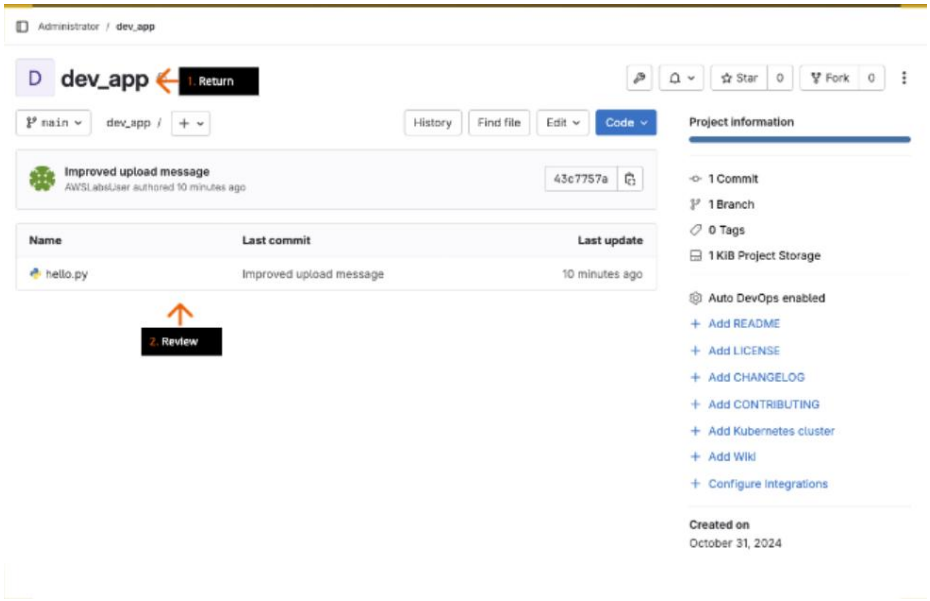


39. Review to confirm that the "git push" command runs successfully.



40. 1. Return to the dev_app repository on the GitLab self-managed browser tab.

2. Review the hello.py file that you just successfully pushed to this repository.
- You might need to refresh your browser to display the file.



DYI:

Use the boto3 s3-uploading-files URL, listed in the upload_file function, to add the missing code to the function. Save and run the hello.py Python file.

Our test service will use the following code to import the hello.py file and call the hello.upload_file function to place a new file in your S3 bucket.

import hello hello.upload_file(file_name, bucket, object_name)

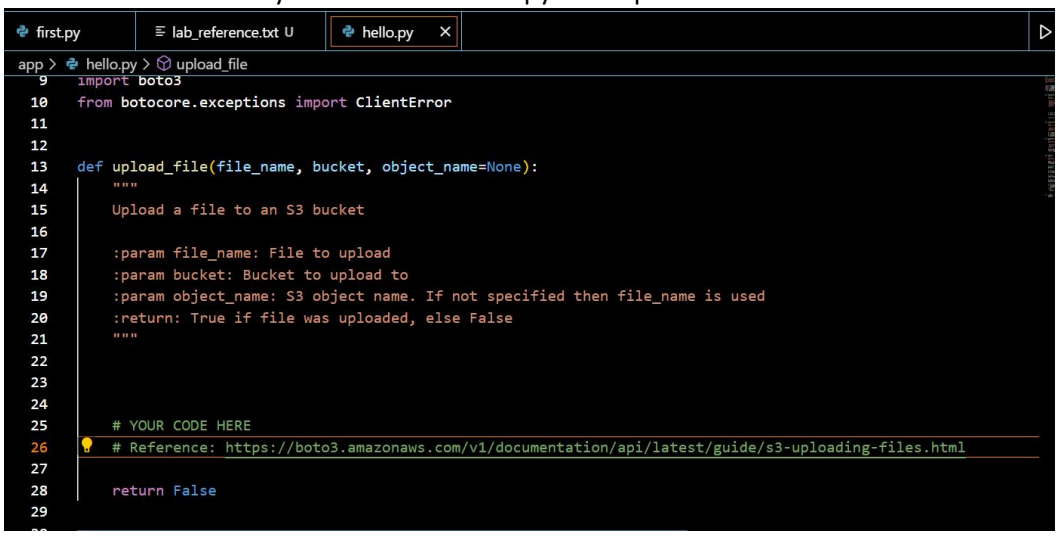
The test service will verify that a file was uploaded to the S3 bucket.

Hints

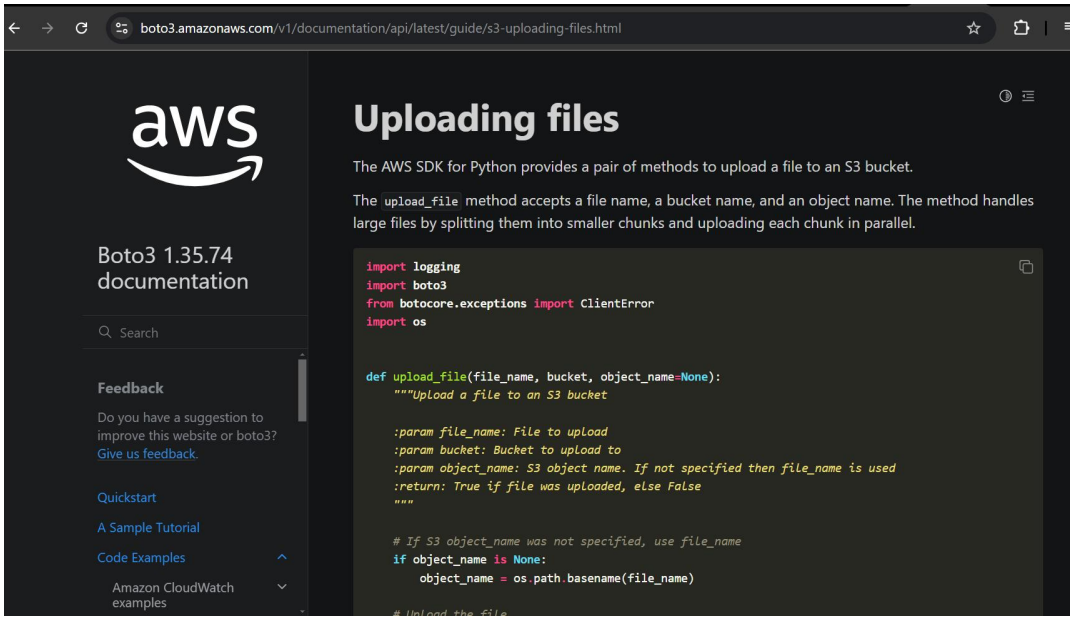
1. A missing library needs to be imported into hello.py. Compare the import statements in hello.py against those in the sample upload_file() function.

Solution:

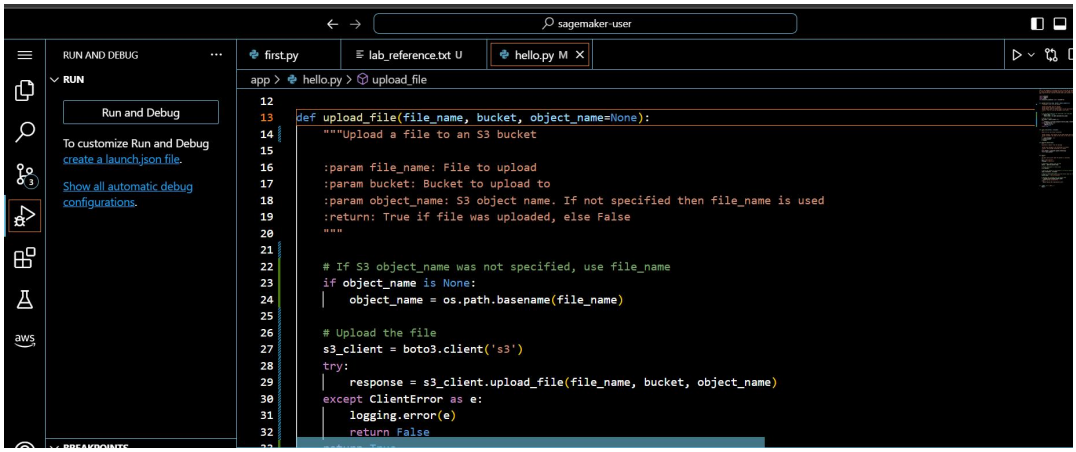
1. On line 26 of the code you can find a url. Copy it and paste it in another browser.



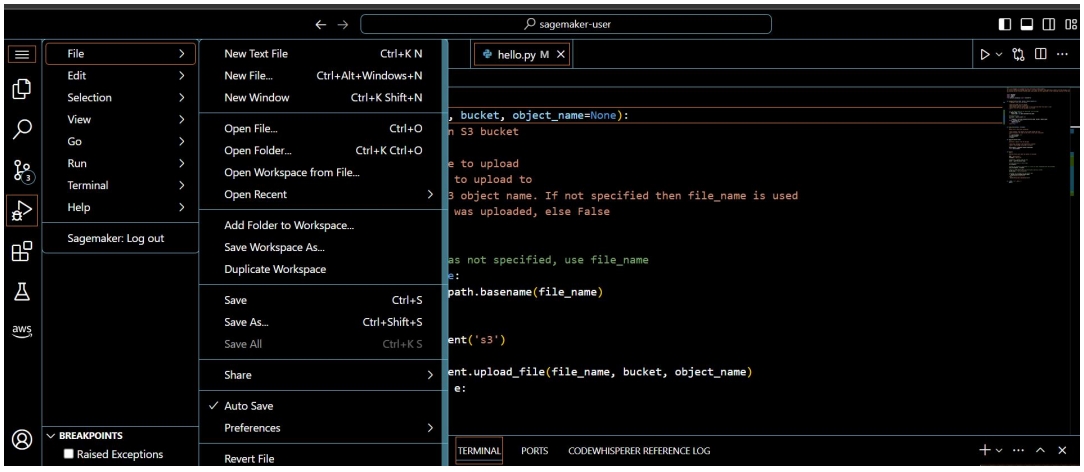
2. You can find a code.



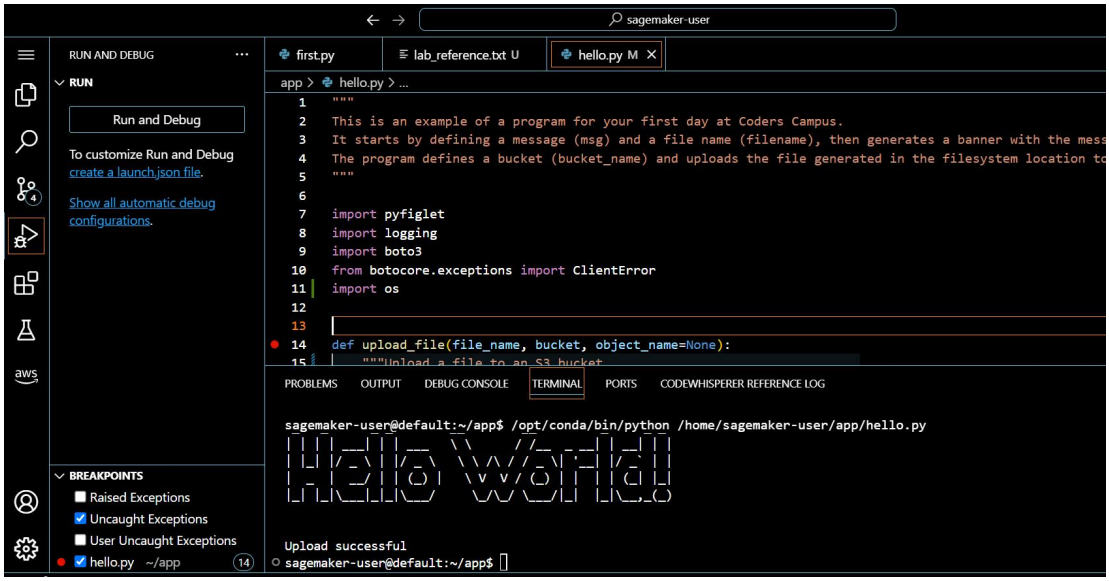
3. Copy the code and replace the upload file function.



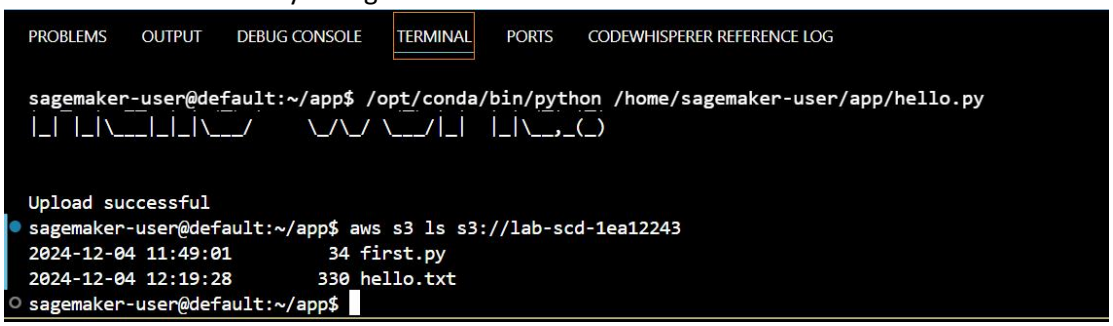
4. Save the code.



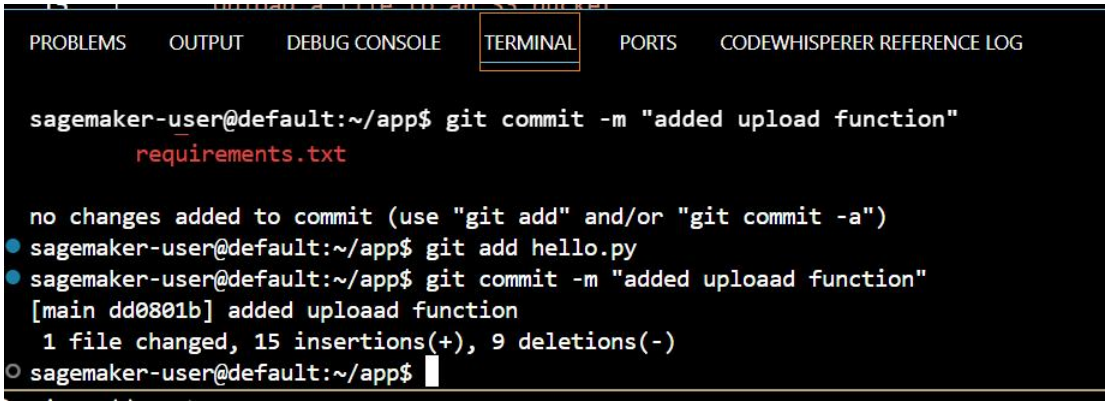
5. Run the code and you should get upload “successful message”.



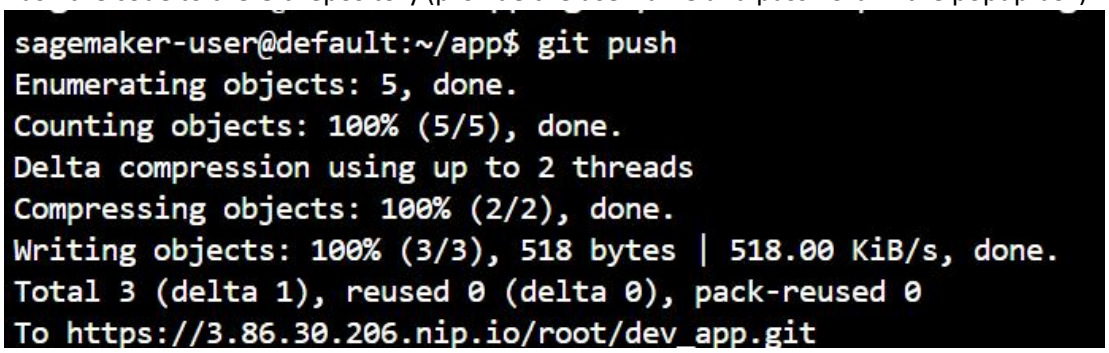
6. This can be confirmed by listing the bucket.



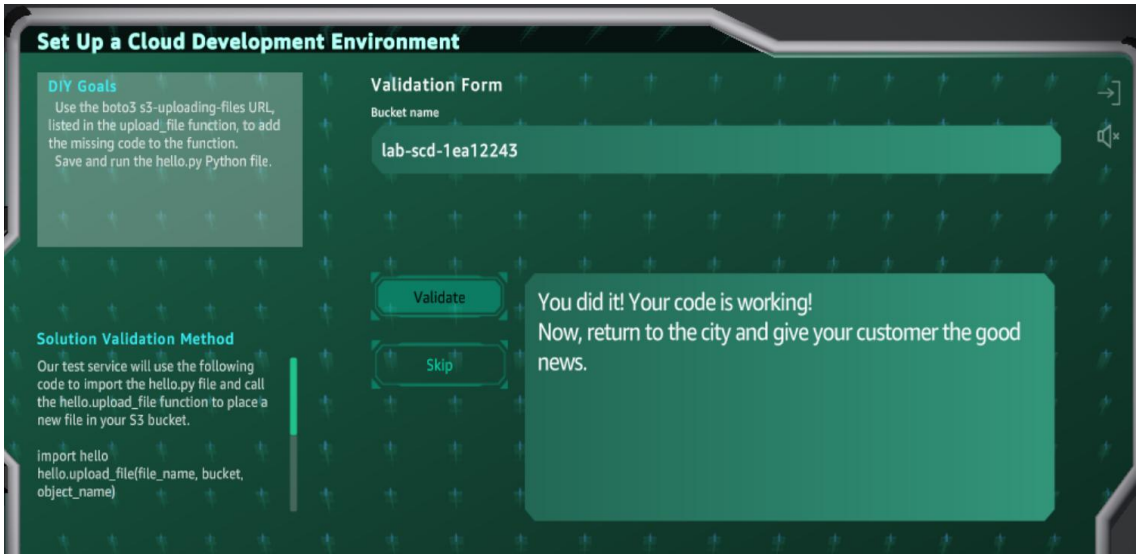
7. Add hello.py and commit the code with the message “added upload function”.



8. Push the code to the Git repository.(provide the username and password in the popup box).



9. Validate the solution by providing the bucket name.



10. The code is working. Thus, the solution is completed.