

## Templating a Serverless Application.

### Solution Request:

Create an application template, using AWS SAM, that is composed of an Amazon API Gateway endpoint, an AWS Lambda function, and an Amazon DynamoDB table, which are the baseline for many serverless applications.

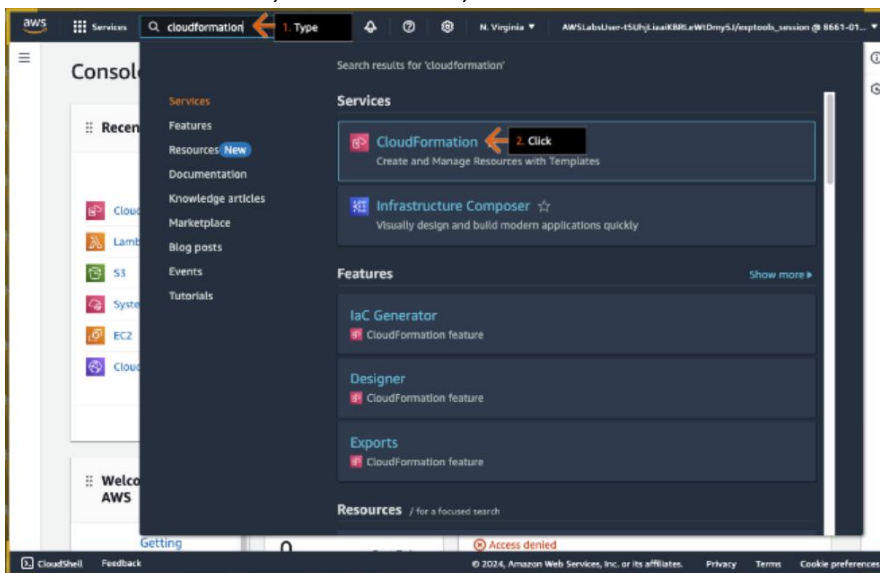
In this lab we will be working on **Cloud9, SAM, CloudFormation.**

### Practice lab:

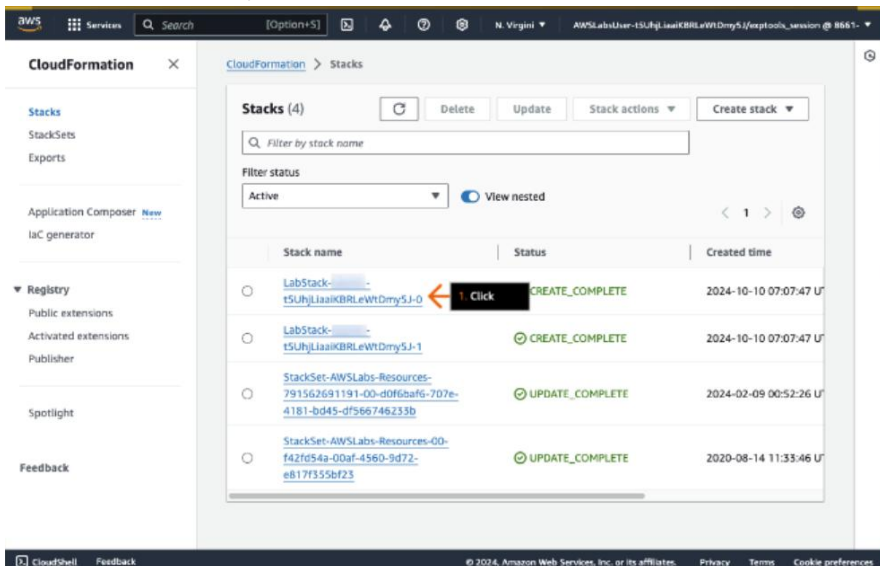
In this practice lab, you will:

- Use AWS SAM to create an Amazon API Gateway, AWS Lambda, and Amazon DynamoDB template.
- Run and debug the API Gateway endpoint and Lambda function locally. Deploy an AWS SAM application and run the application remotely.
- Create a repository on GitLab self- managed to store the application.

1. In the top navigation bar search box, type: CloudFormation
2. In the search results, under Services, click CloudFormation.

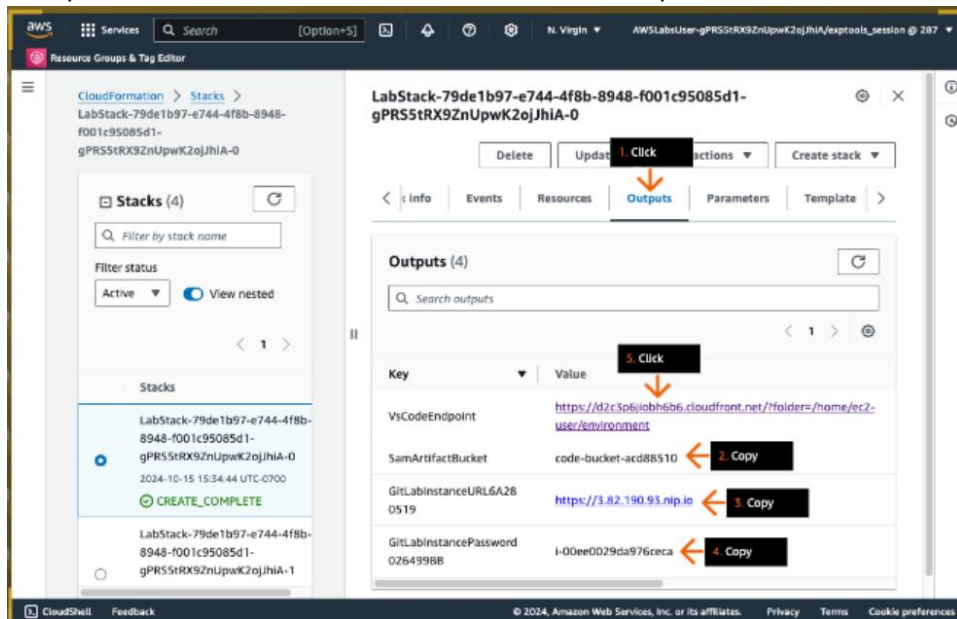


2. In the Stacks section, click the stack name that starts with Lab Stack and ends with -0.

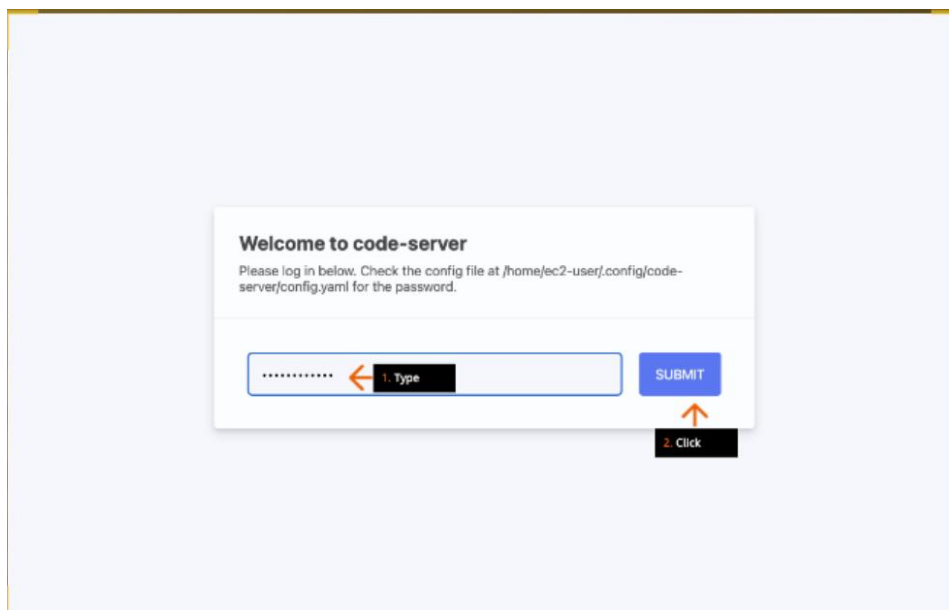


3. 1. Click the Outputs tab.

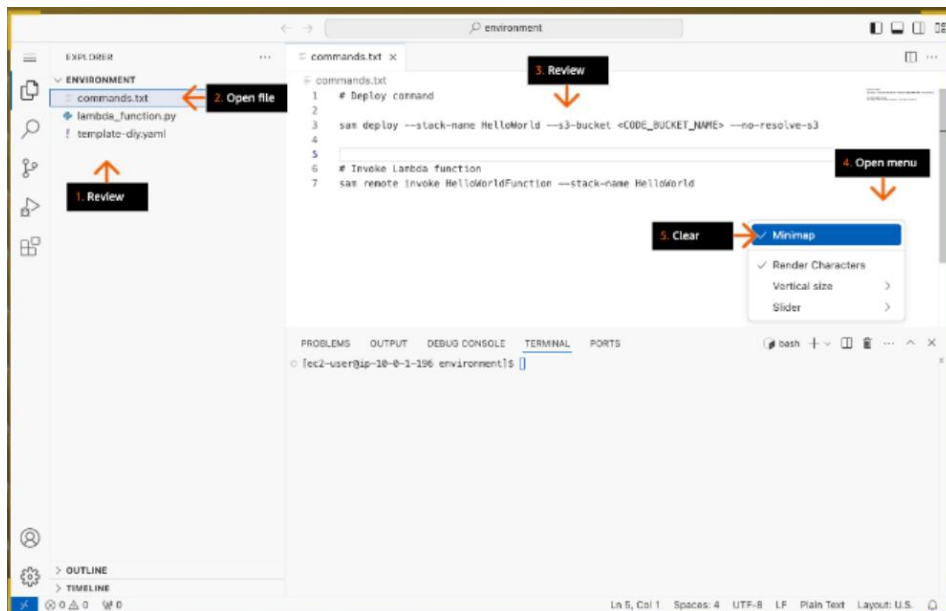
2. For the SamArtifactBucket key, under Value, select (highlight) and copy the provided value, and then paste it in the text editor of your choice on your device.
3. Repeat this for the GitLabInstanceURL key.
4. Repeat this for the GitLabInstancePassword key.
5. For the VsCodeEndpoint key, under Value, click the value link.
- The VS Code IDE (integrated development environment) opens in a new browser tab (or window).
- Keep the AWS CloudFormation console browser tab open. You return to it.



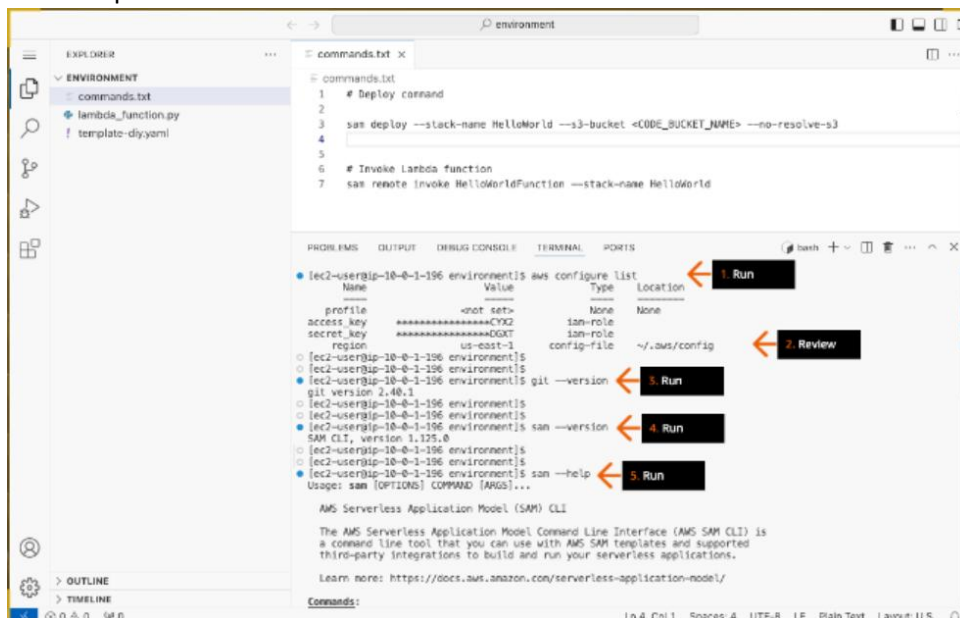
4. 1. In the text box, type: LabUserCQ987!
2. Click SUBMIT.



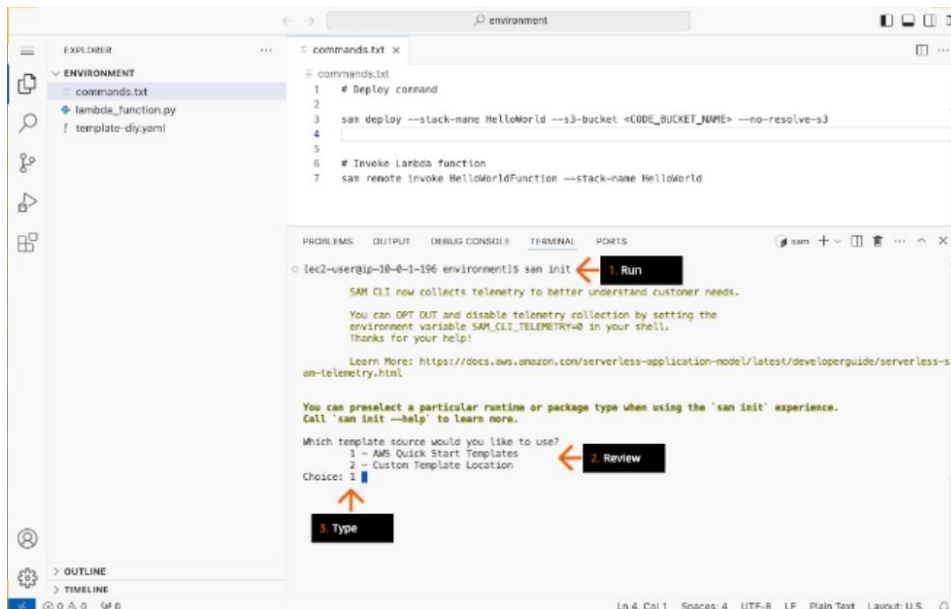
5. 1. In the left Explorer window, review available files.
  - The lambda\_function.py file is used in a later step.
  - The template-diy.yaml file can be used as a hint in the later DIY section of this solution.
2. Open (double click) the commands.txt file.
3. In the top commands.txt window, review the file.
  - This file contains AWS SAM CLI commands and AWS CLI commands that you use in later steps.
4. In the window, open the context (right-click) menu.
5. Clear the check to deselect Minimap.



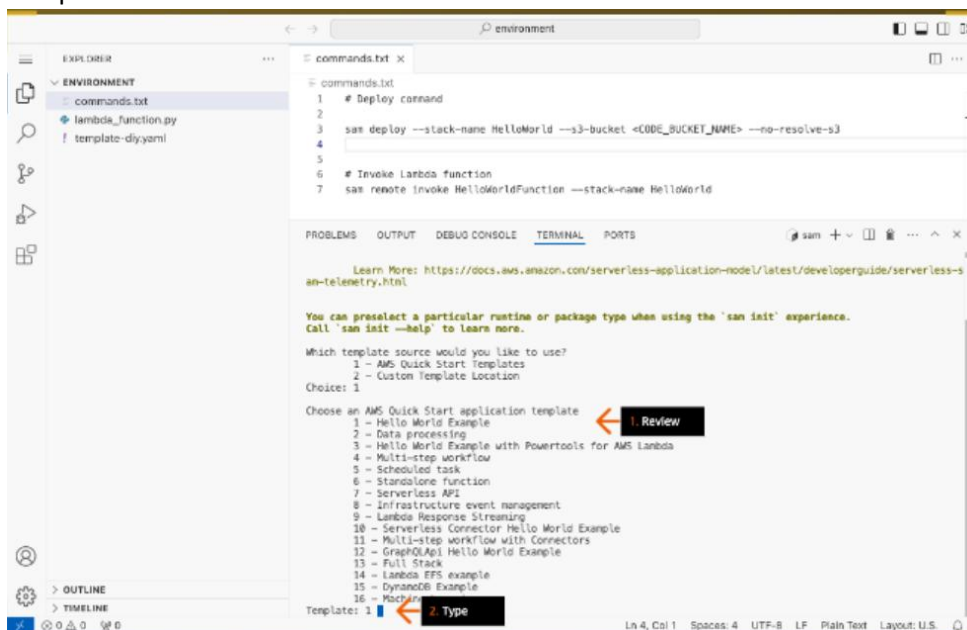
6. 1. In the bottom terminal window, at the command prompt, run (type the command and press Enter):  
aws configure list  
- You can also copy-paste this text. If you receive an undefined value when you paste this, try again.
2. Review the preconfigured default AWS profile that uses the AWS Region, us-east-1.
3. To check the git version, run:  
git --version
4. To check the SAM CLI version, run:  
sam --version
5. To view the complete list of SAM commands, run:  
sam --help



7. 1. To initialize a new serverless application, run:  
sam init
2. For "Which template source would you like to use?", review the available options.
3. For Choice, type:  
1  
and press Enter.

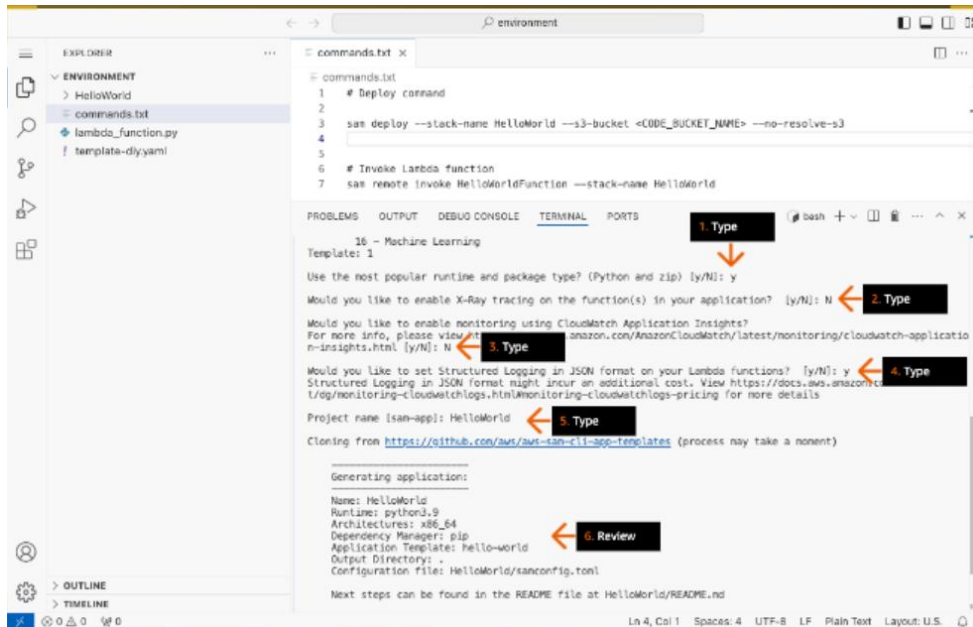


8. 1. For "Choose an AWS Quick Start application template," review available options.
2. For Choice, type:
- 1
- and press Enter.

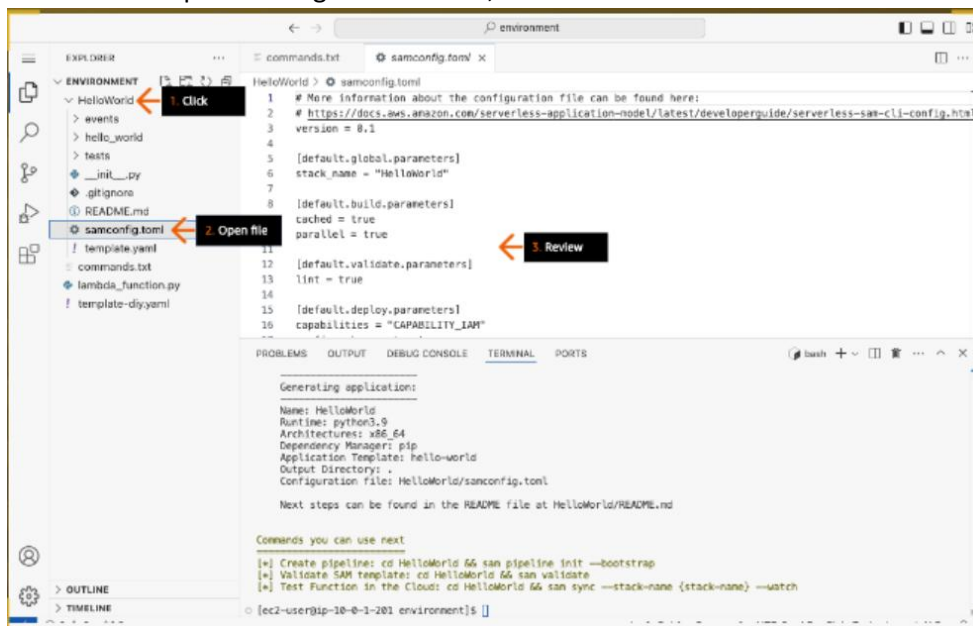


9. - For the following questions, press Enter after entering each value in the terminal.
1. For "Use the most popular runtime and package type?", type:
- y
2. For "Would you like to enable X-Ray tracing on the function(s) in your application?", type:
- N
3. For "Would you like to enable monitoring using CloudWatch Application Insights?", type:
- N
4. For "Would you like to set Structured Logging in JSON format on your Lambda functions?", type:
- y
5. For Project name [sam-app], type:
- HelloWorld
- and press Enter.
- Make sure you use only this project name.
6. Review the details of the HelloWorld project configuration.

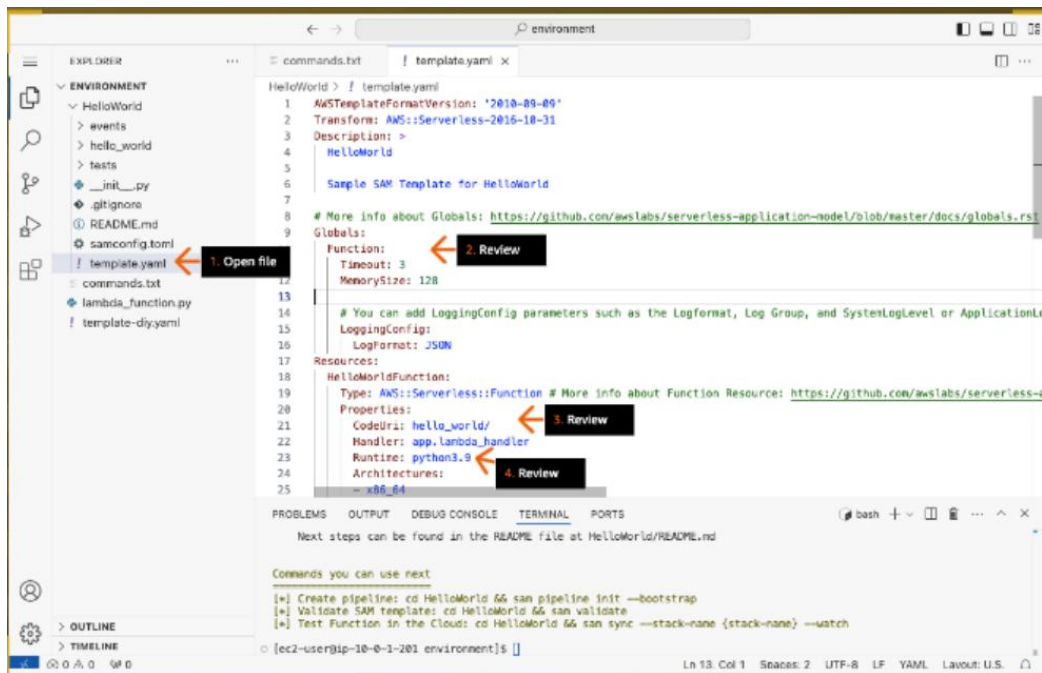
- The SAM CLI downloads your starting template and creates the application project directory structure on your local machine.



1. In the left Explorer window, click to expand the HelloWorld folder.
2. Open (double click) the samconfig.toml file.
3. In the the top samconfig.toml window, review the file.



1. In the left Explorer window, open the template.yaml file.
2. In the top the template.yaml window, review the Globals section of the template.
3. Review the HelloWorldFunction resource.
  - AWS::Serverless::Function creates a Lambda function, IAM execution role, and event source mappings which invoke the function.
4. Review to confirm that the runtime is set to Python 3.9.
  - If any other value is chosen, change it to Python 3.9 and save the file.



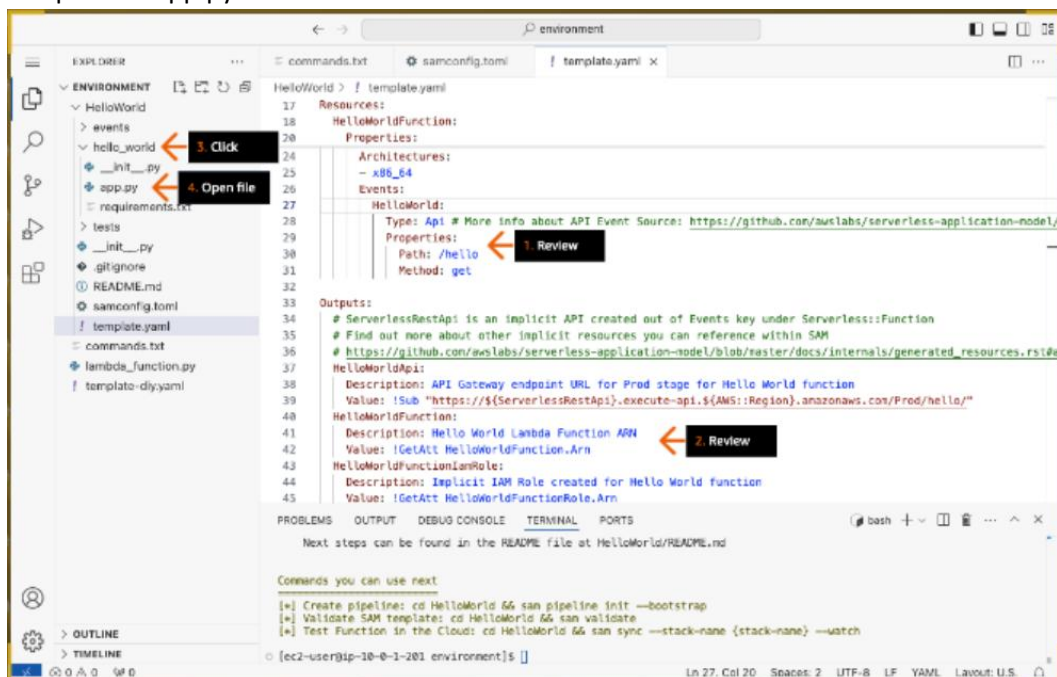
## 12. 1. Review the Events property of the HelloWorldFunction.

- Behind the scenes, SAM creates an implicit AWS::Serverless::Api as the event source for HelloWorldFunction function. This API defaults to a StageName called "Prod" that cannot be configured.

## 2. Review the Outputs section of the template.

3. In the left Explorer window, click to expand the hello\_world folder.

4. Open the app.py file.

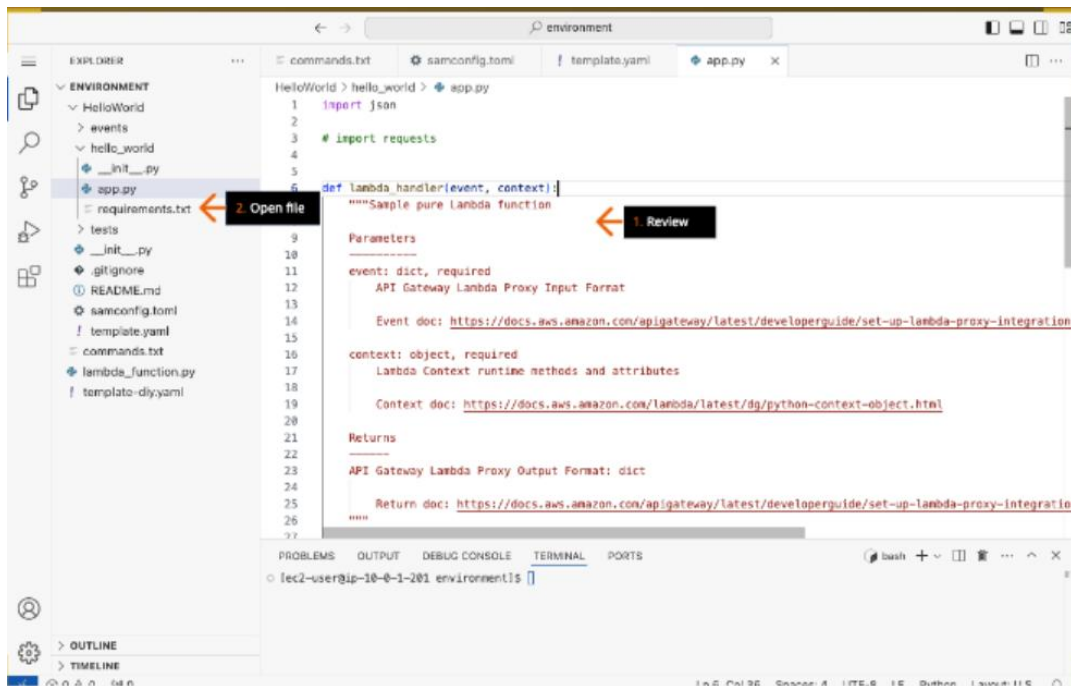


## 13. 1. In the top app.py window, review the file.

- This file contains the Lambda function code.

2. In the left Explorer window, open the requirements.txt file.





#### 14. 1. Review the requirements.txt file.

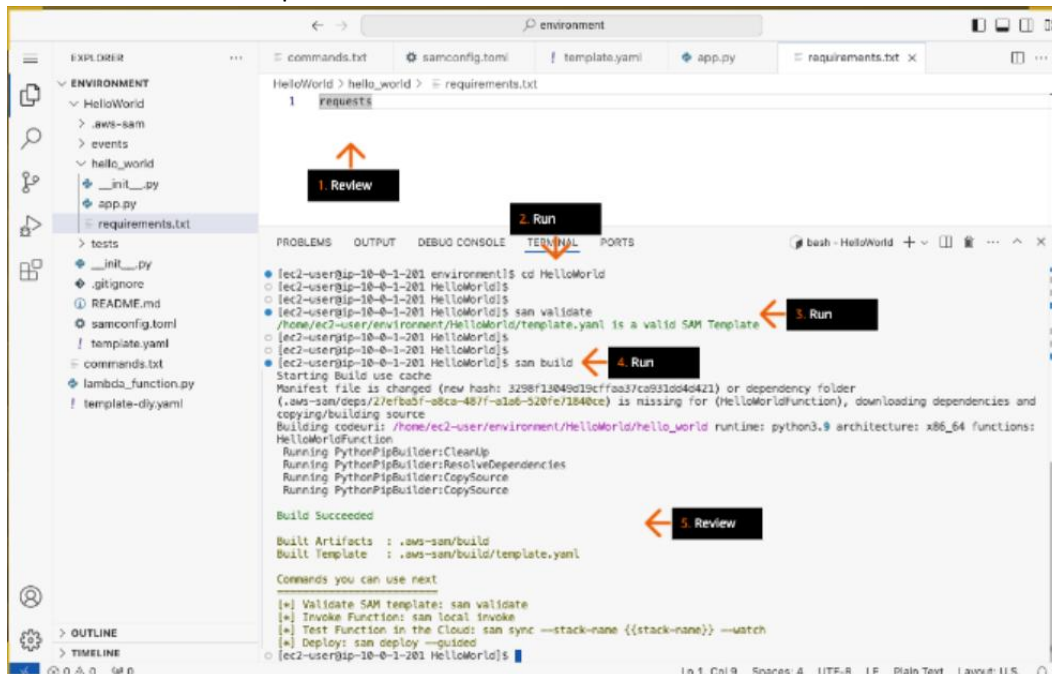
- This file contains any Python dependencies that your Lambda function requires.

2. In the bottom terminal, to change current directory to the project HelloWorld folder, run:  
cd HelloWorld

3. To validate that the template.yaml file is valid, run:  
sam validate

4. To build your application and prepare for deployment, run:  
sam build

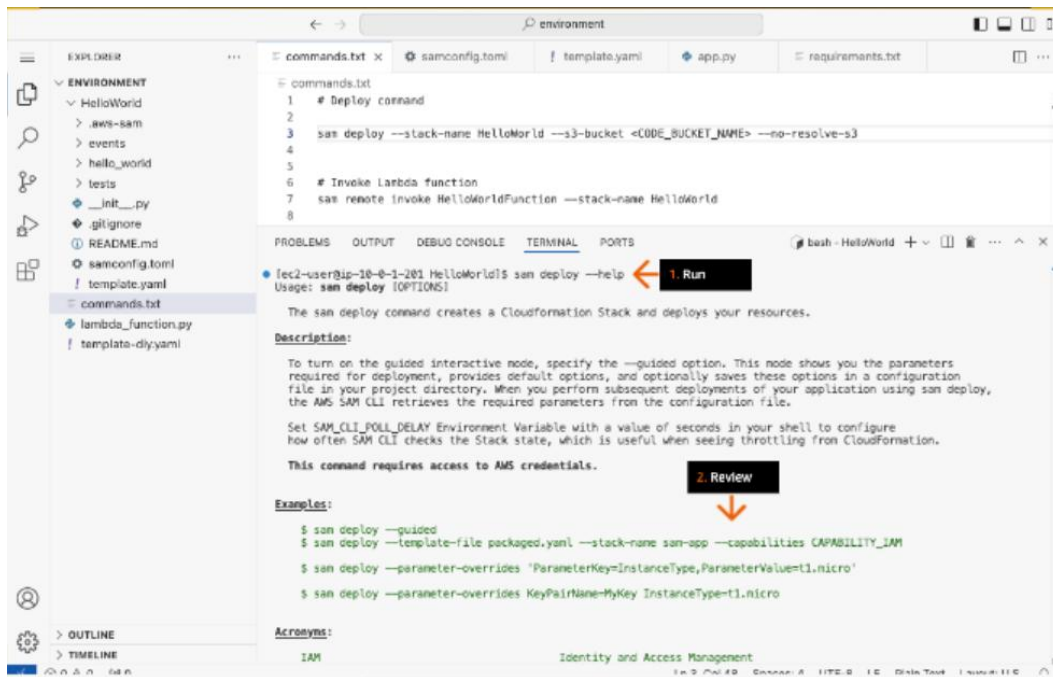
5. Review the build output.



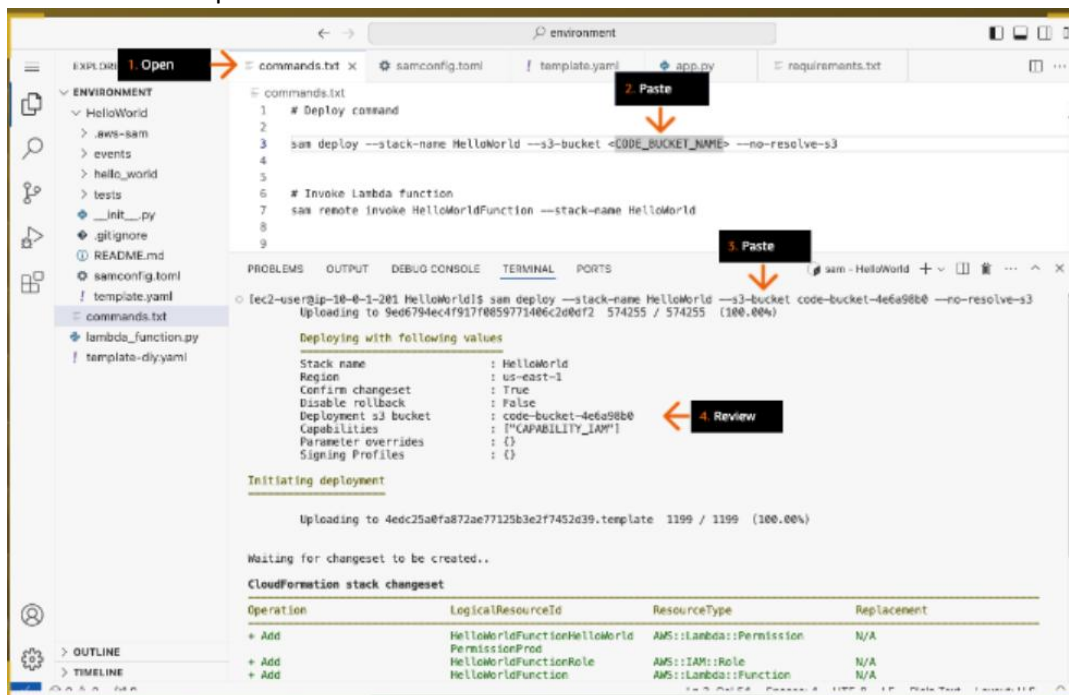
#### 15. 1. In the terminal, run:

sam deploy --help

2. Review the "sam deploy" command options listed in the output.

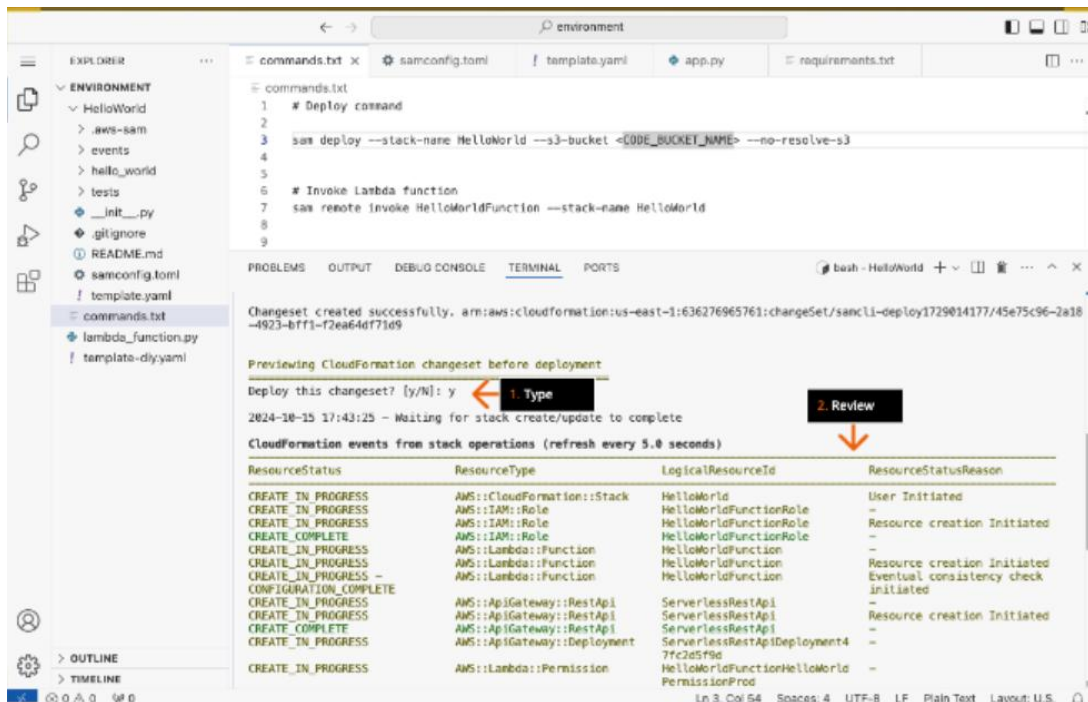


16. 1. To open the `commands.txt` window, click the tab.
2. On line 3, to replace `CODE_BUCKET_NAME`, paste the S3 bucket name that you copied earlier.
  - In this command, we are specifying a preprovisioned S3 bucket as the location where artifacts referenced in the template are uploaded.
  - We use the `--no-resolve-s3` option so that the SAM CLI will not automatically create and resolve a managed default S3 bucket.
3. From the top window, copy-paste the line 3 command to the bottom terminal.
  - If a pop-up box appears that asks if a cloud domain can see text and images copied to the clipboard, click Allow.
4. Review the output.

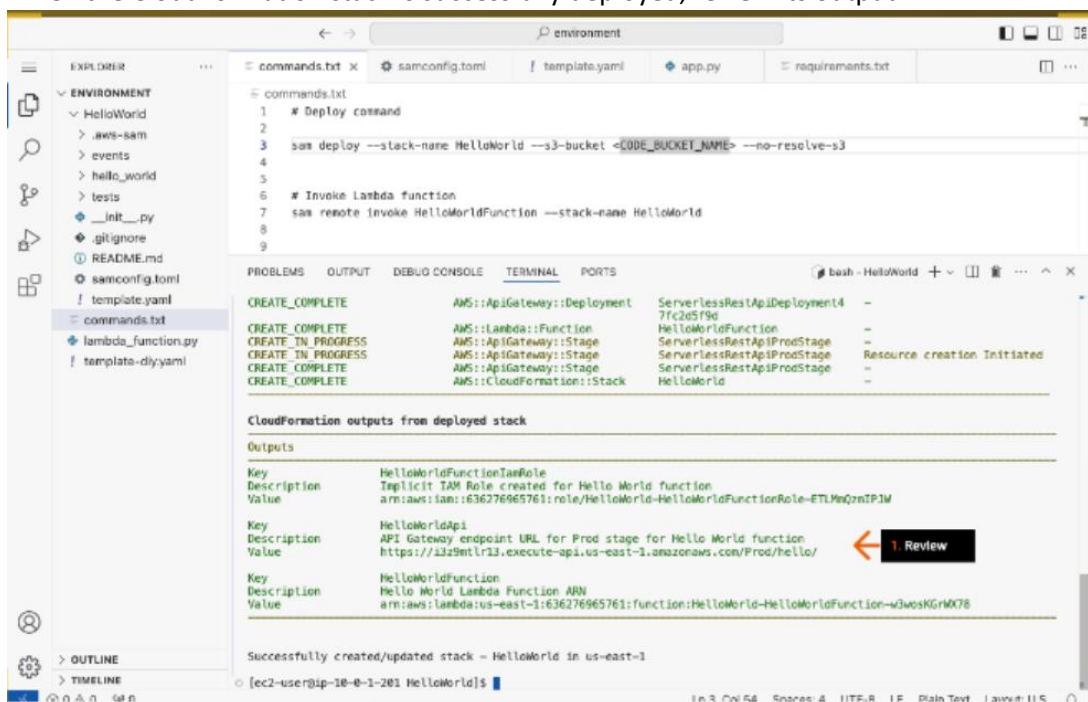


17. 1. For "Deploy this changeset?", type:
  - y
  - and press Enter.
2. Review the CloudFormation events while the stack is creating.

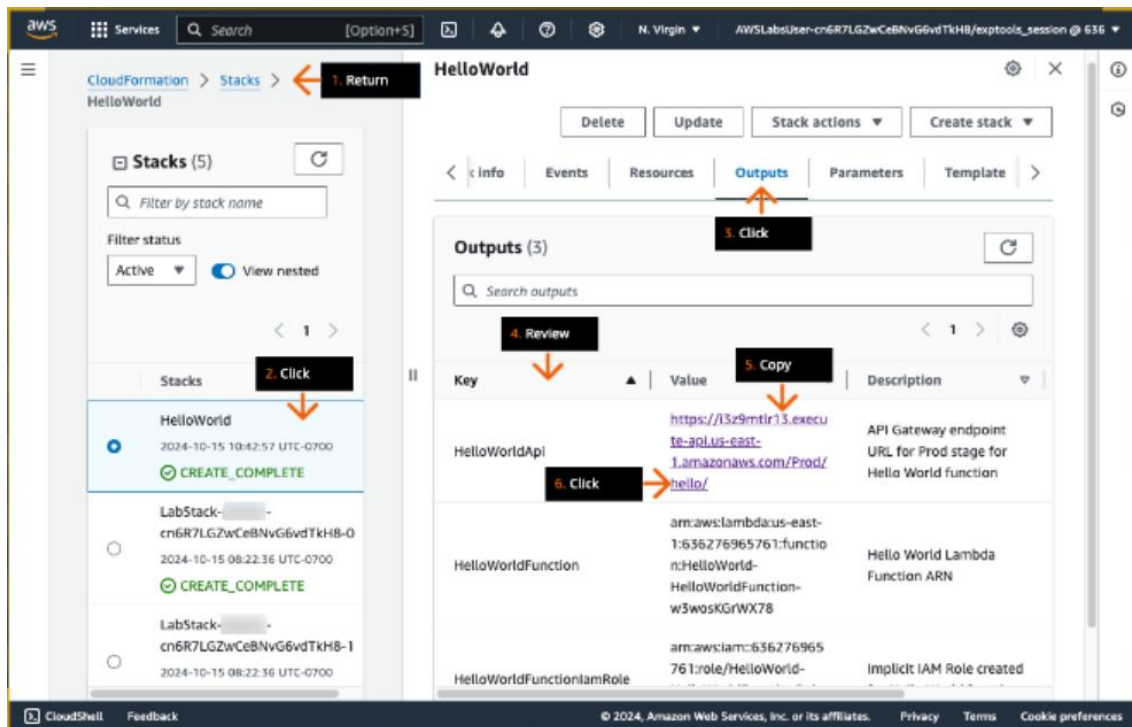




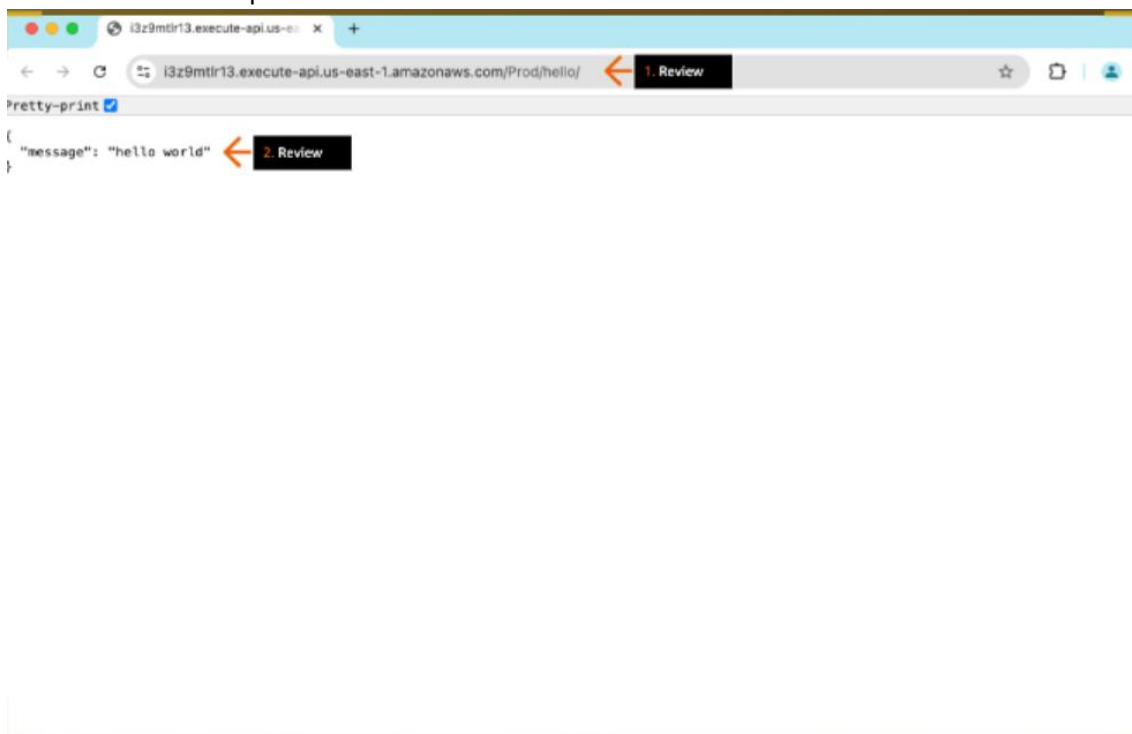
18. When the CloudFormation stack is successfully deployed, review its output.



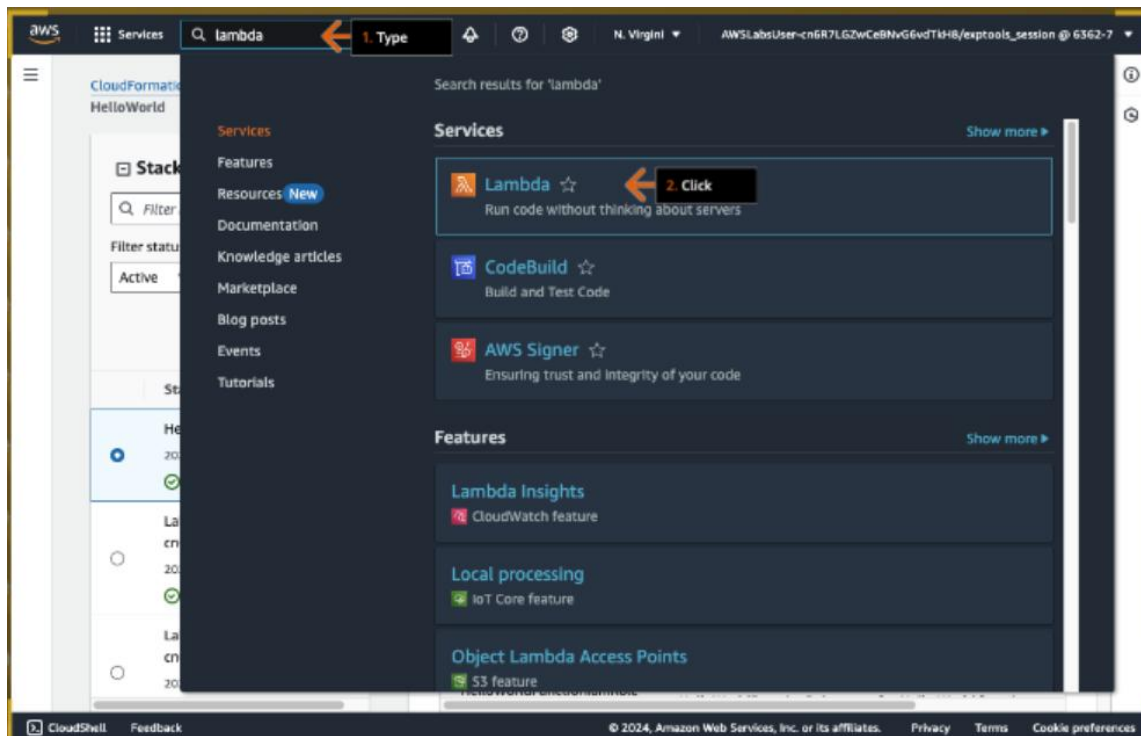
- Return to the AWS CloudFormation console in the other browser tab.
- In the left Stacks pane, click HelloWorld.
- In the right HelloWorld pane, click the Outputs tab.
- Review three resources created by this CloudFormation stack.
- For the HelloWorldApi key, under Value, select (highlight) and copy the provided URL, and then paste it in your text editor.
  - You use the URL in a later step.
- Click the same URL.
  - It opens in a new browser tab (or window).



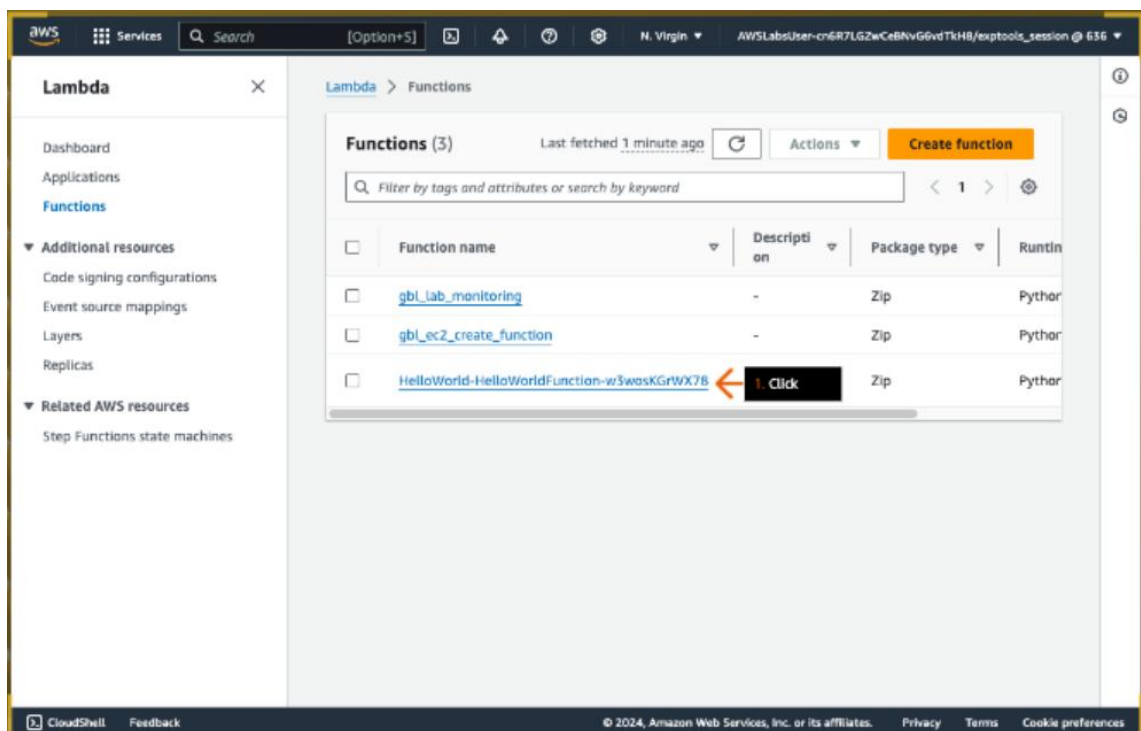
20. 1. In the new browser tab, review the API Gateway endpoint URL.  
- This API is configured in the Events property section of the template.yaml file.
2. Review the response from this API.
3. Go to the next step.



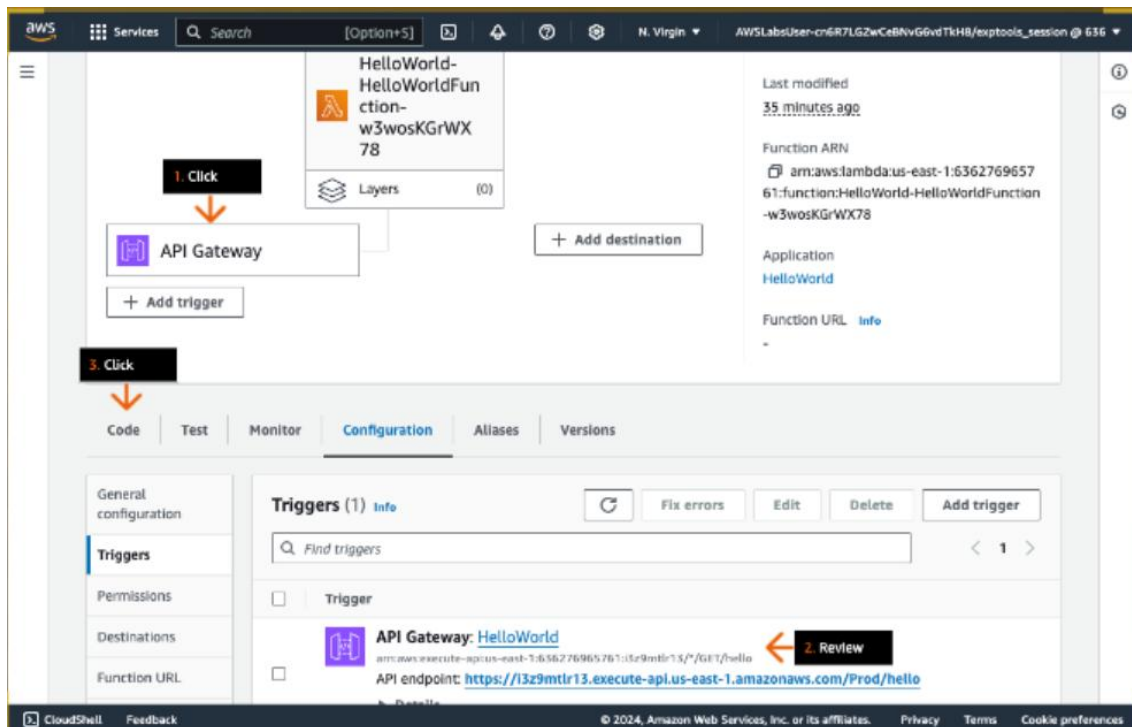
21. 1. In the top navigation bar search box, type: lambda
2. In the search results, under Services, click Lambda.



22. In the Functions section, click the function that starts with HelloWorld-HelloWorldFunction-.  
 - The function name follows the format [SAM project name]-[function name]-[unique id]

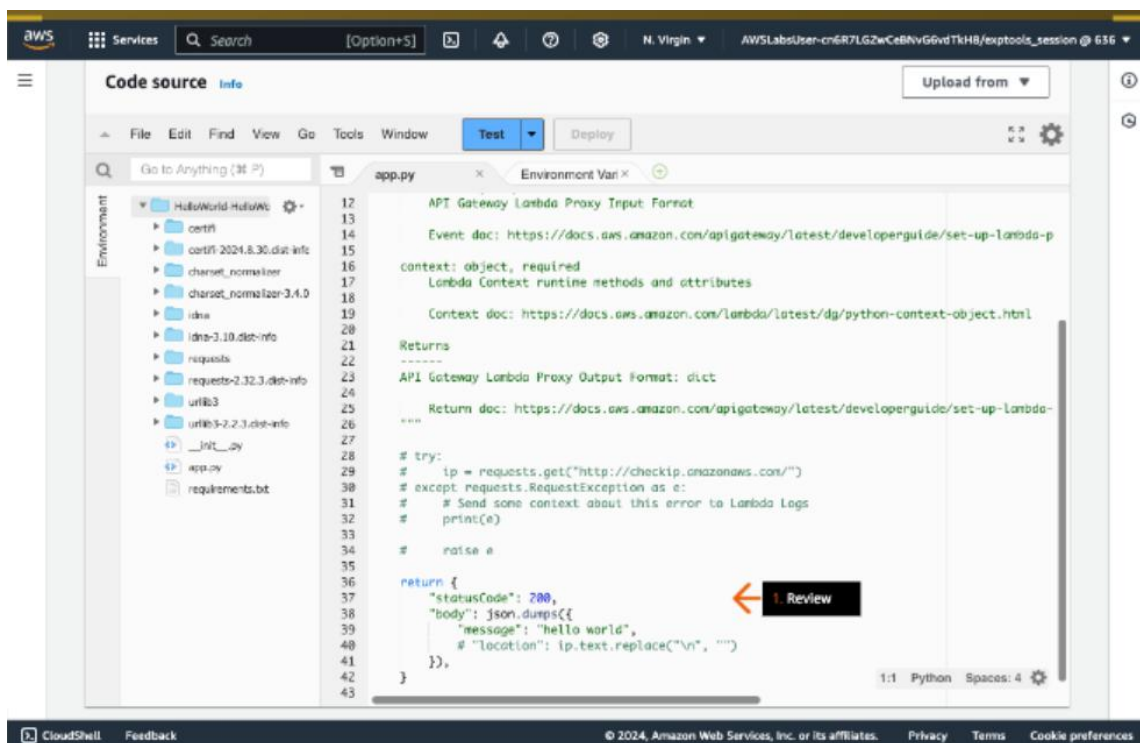


23. 1. In the Function overview section, click API Gateway.  
 2. On the Configuration tab, in the Triggers sections, review the API endpoint for HelloWorld API Gateway.  
 - You opened this API endpoint in an earlier step to invoke the HelloWorld-HelloWorldFunction.  
 3. To view the function code, click the Code tab.



24. 1. Review the function code.

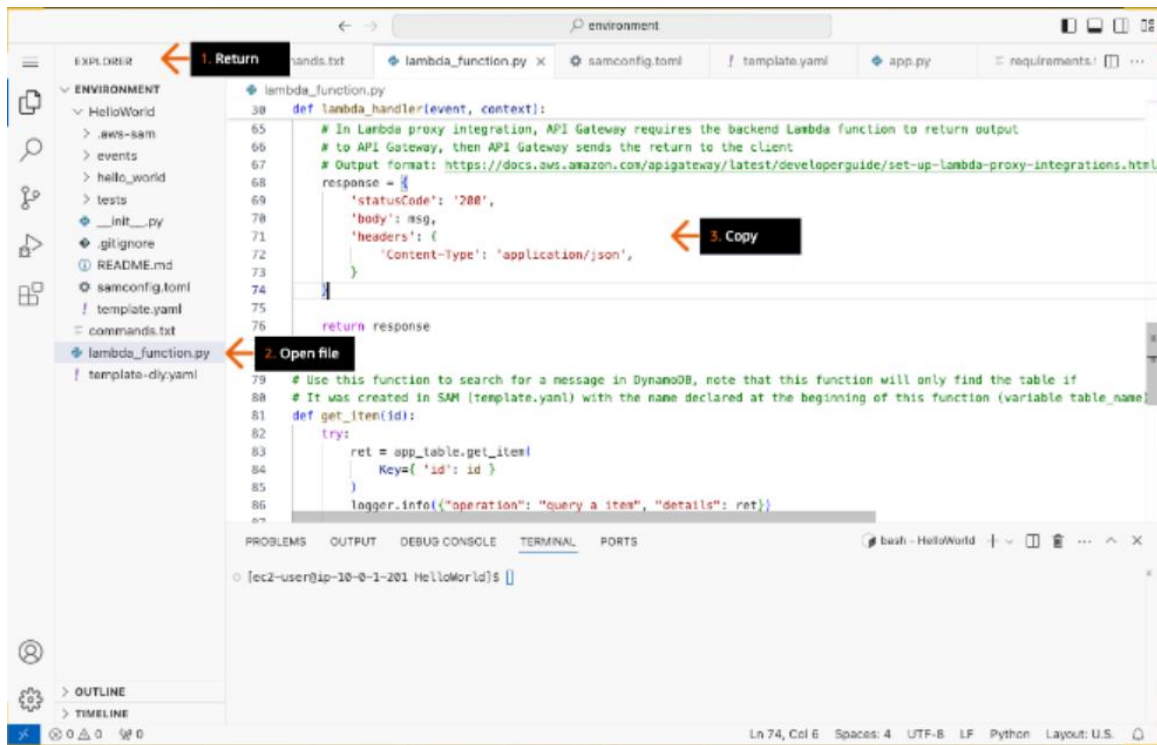
- For now, the function returns only a basic "hello world" message when it is invoked by the API Gateway endpoint.



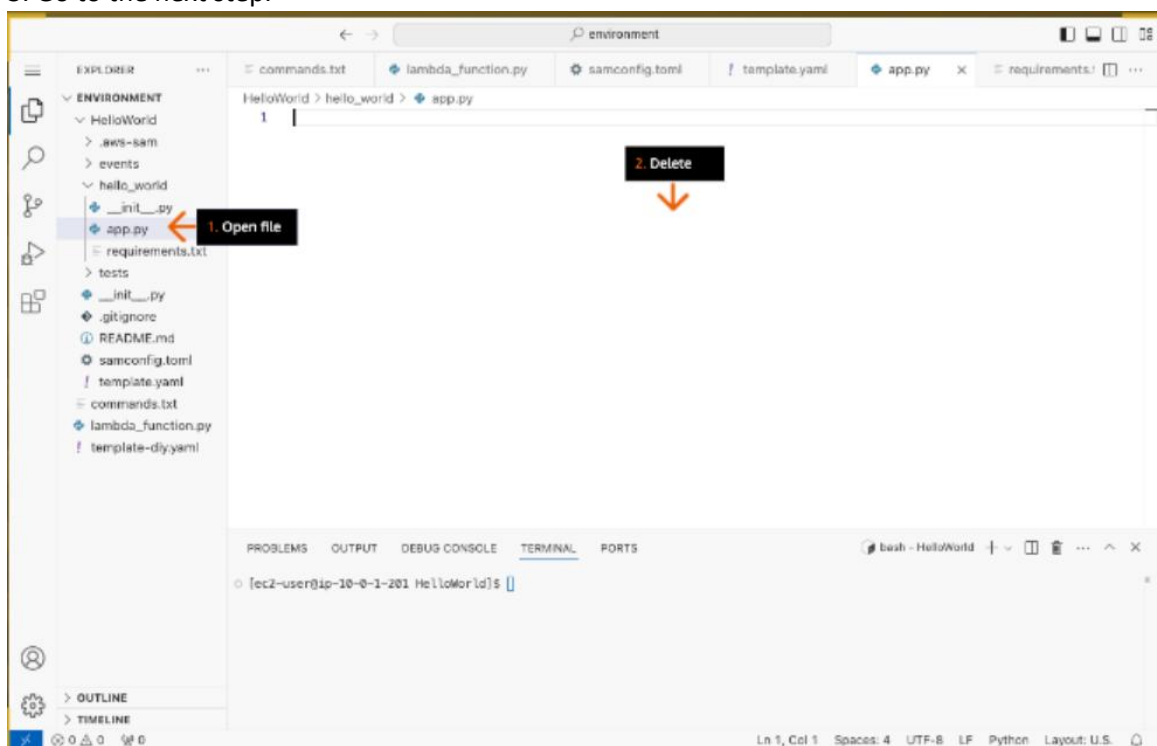
25. 1. Return to the VS Code IDE browser tab.

2. In the left Explorer window, open the lambda\_function.py file.

3. In the top lambda\_function.py window, copy the file contents.

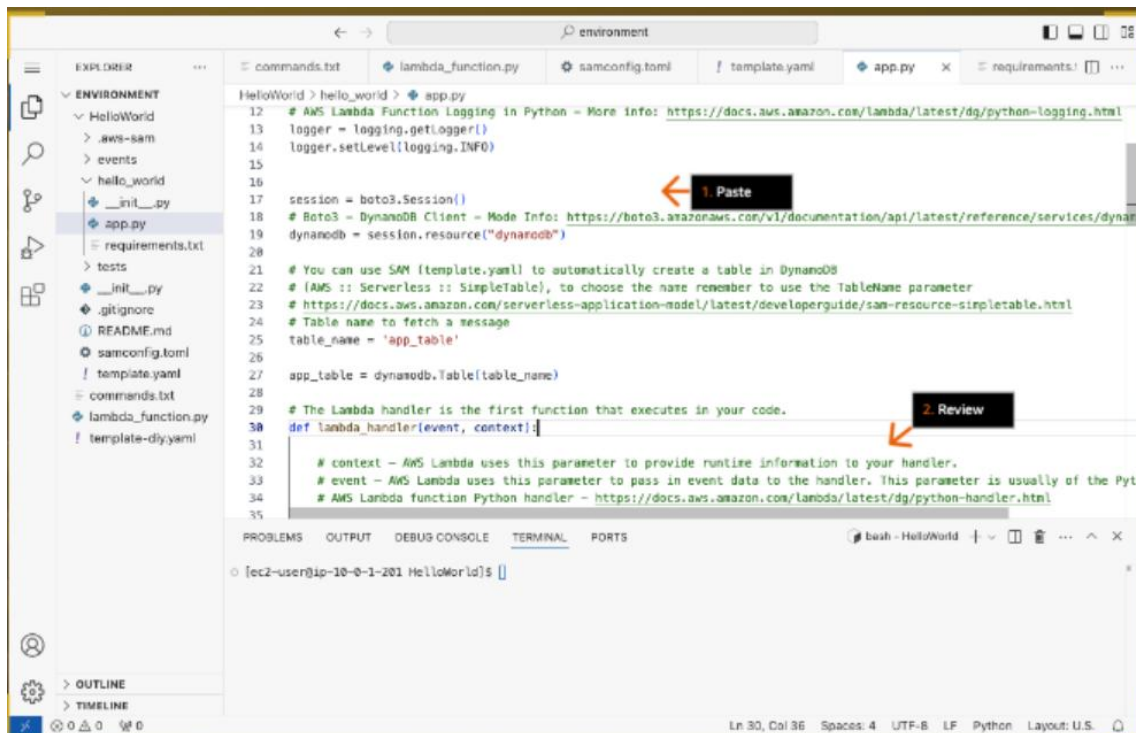


26. 1. In the Explorer window, open the app.py file.
2. In the app.py file window, delete the existing code.
3. Go to the next step.

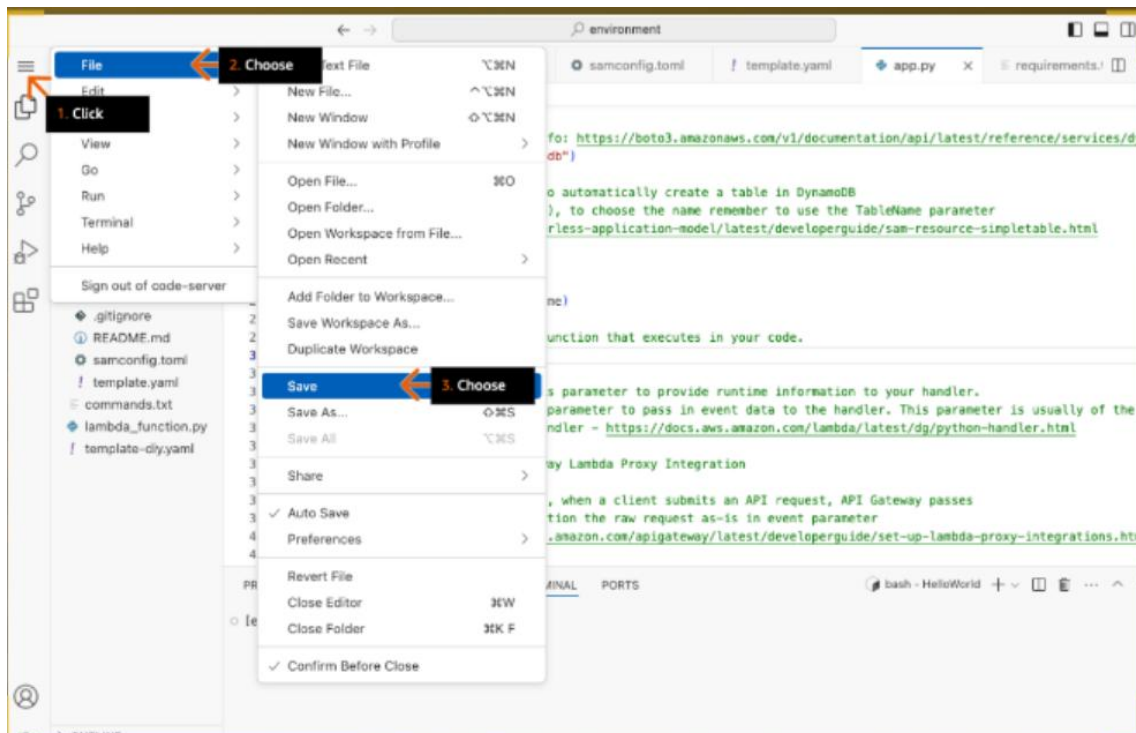


27. 1. In the app.py window, paste the sample code from lambda\_function.py.
  - Make sure to keep the indentations in the Python code blocks.
2. Review the comments for each code block.
  - Pay close attention to these comments. You use them to complete the later DIY section of this solution.

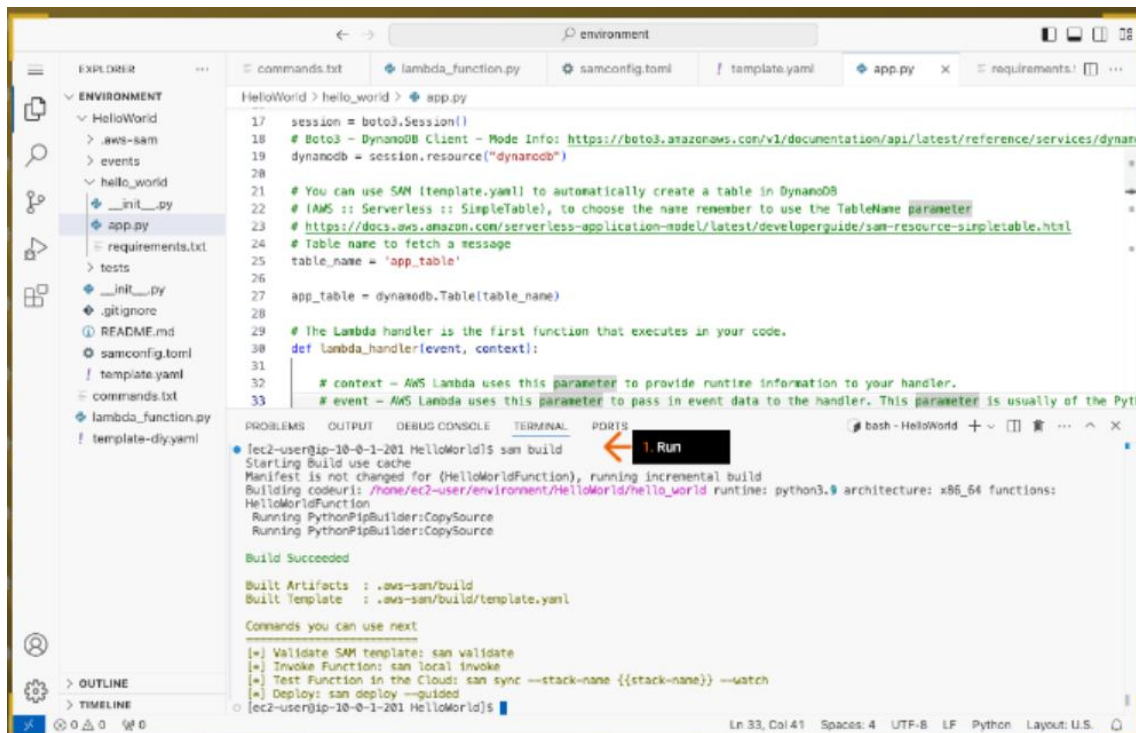




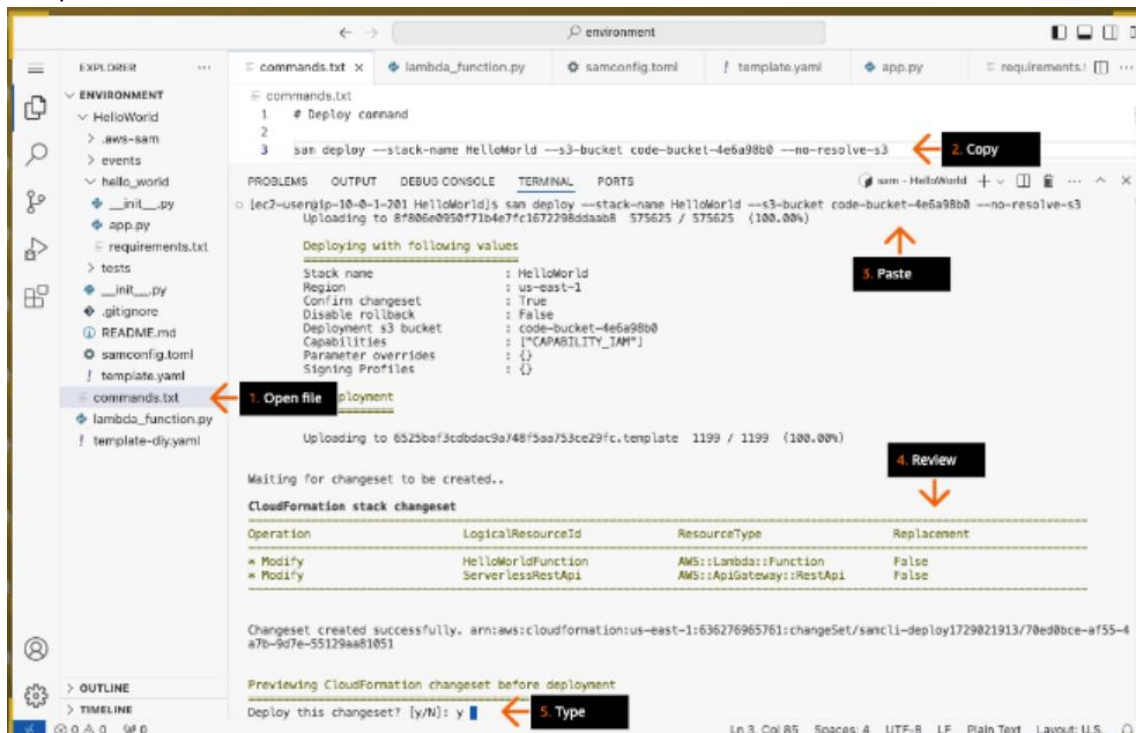
28. 1. In the left sidebar, click the menu icon (three lines) to expand the menu.
2. Choose File.
3. Choose Save.



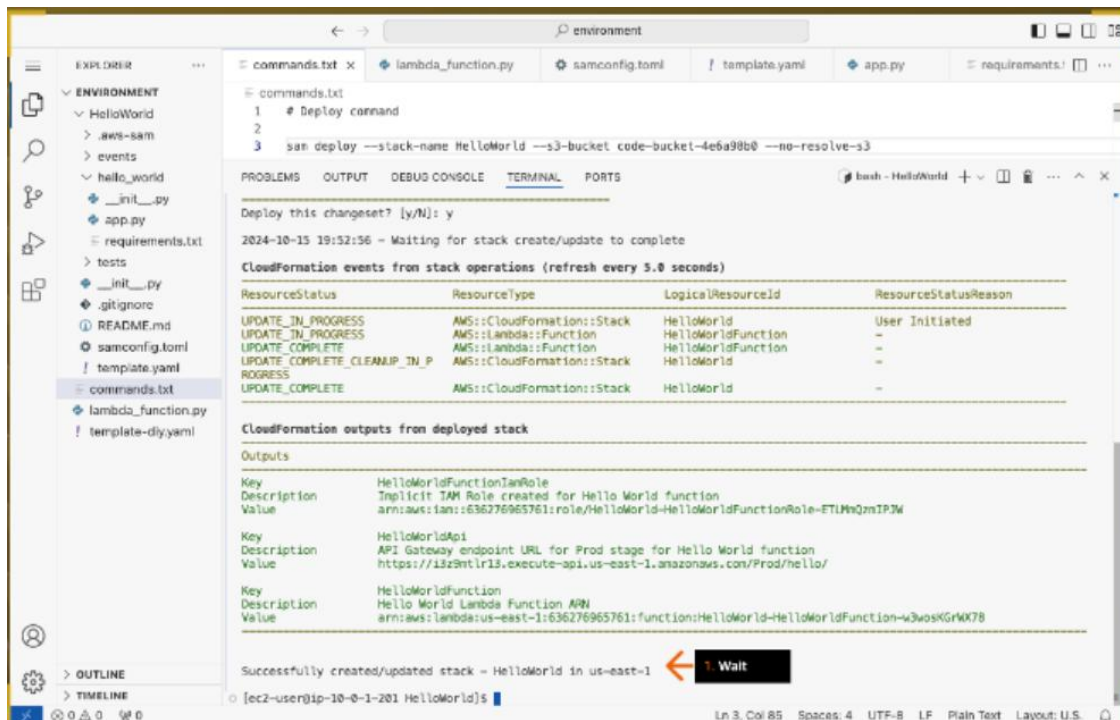
29. In the bottom terminal window, run:  
sam build



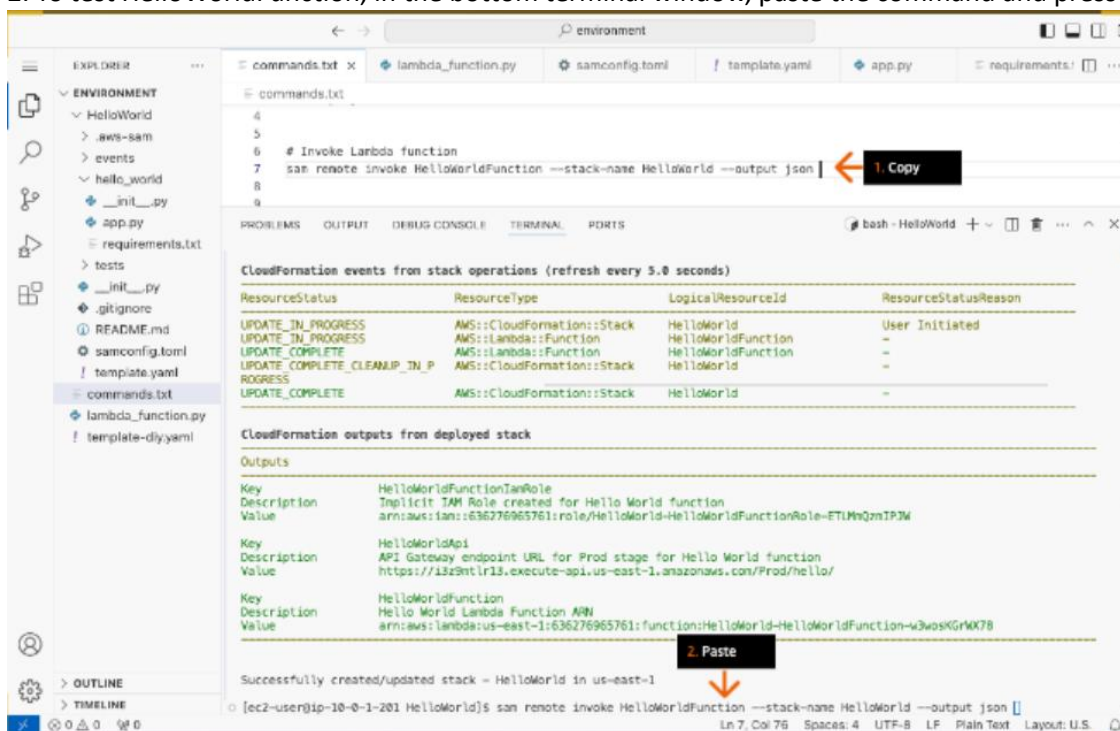
30. 1. In the left Explorer window, open the `commands.txt` file.
2. In the top `commands.txt` window, on line 3, copy the deploy command.
3. In the bottom terminal window, paste the command and press Enter.
4. Review the CloudFormation stack changeset.
5. For "Deploy this changeset?", type `y` and press Enter.



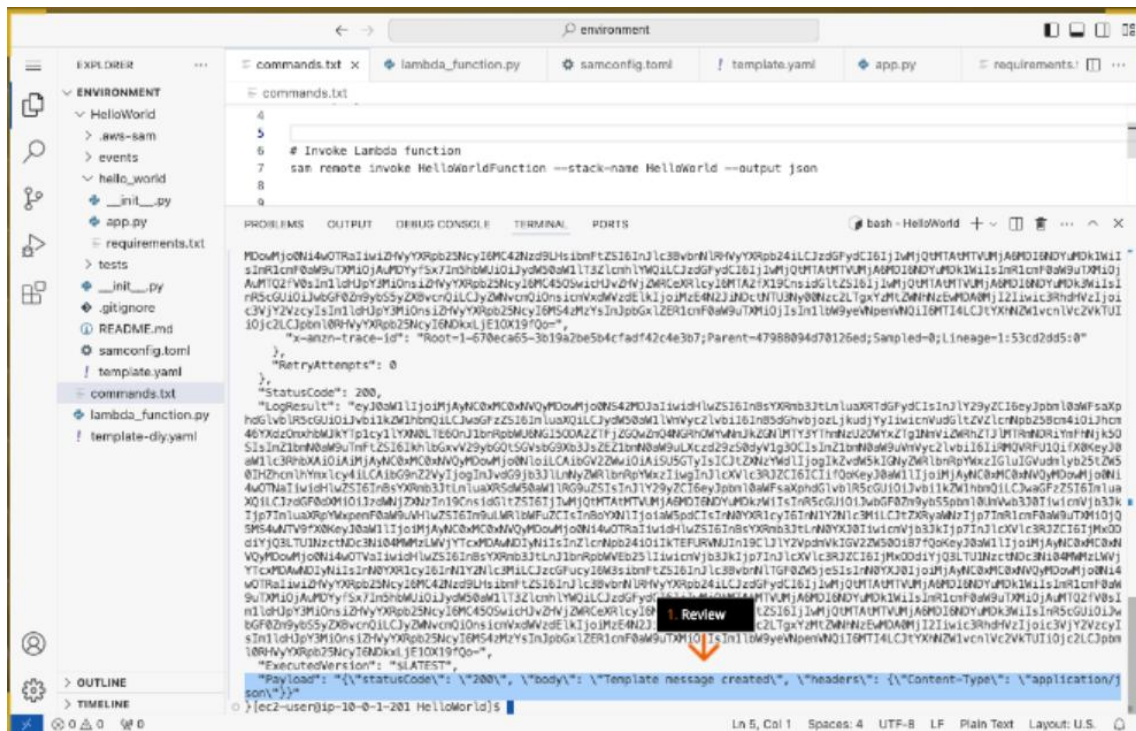
31. 1. Wait until the deployment is successful.
- This deployment updates the Lambda function, `HelloWorldFunction`, with the code you copied from `lambda_function.py` file.



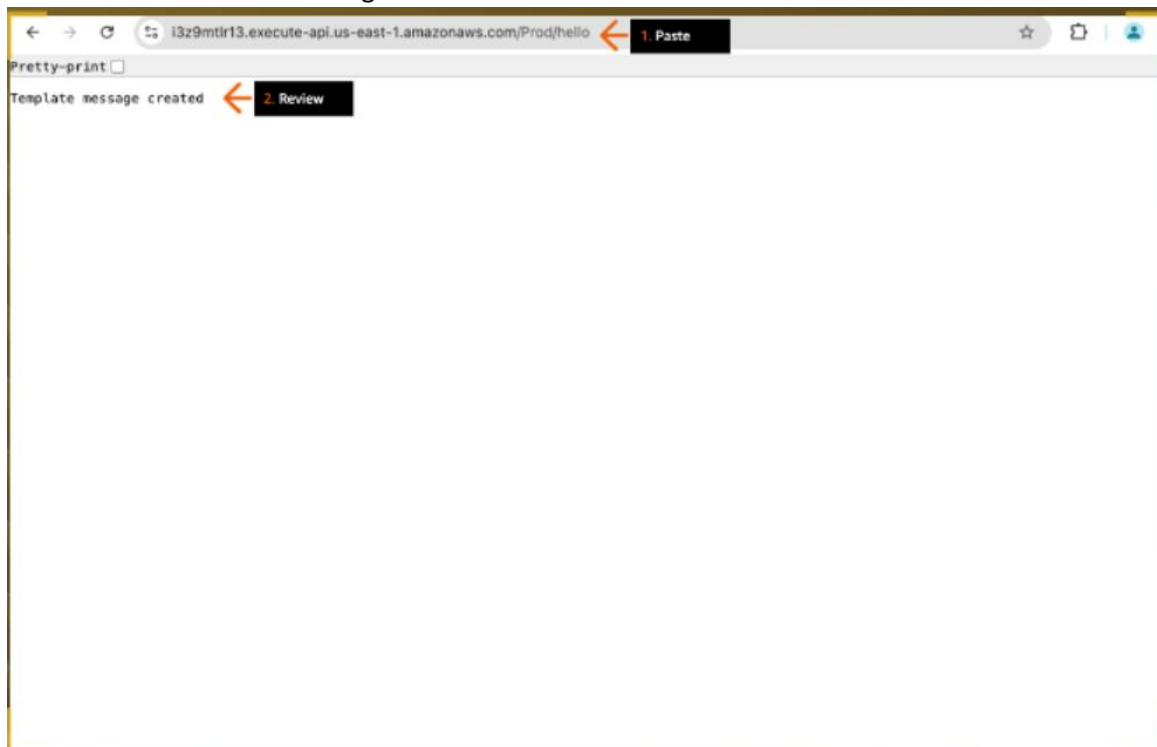
32. 1. In the top commands.txt window, on line 7, copy the command.
2. To test HelloWorldFunction, in the bottom terminal window, paste the command and press Enter.



33. 1. Review the output.
- The Payload (response from the Lambda function) now returns a different response.

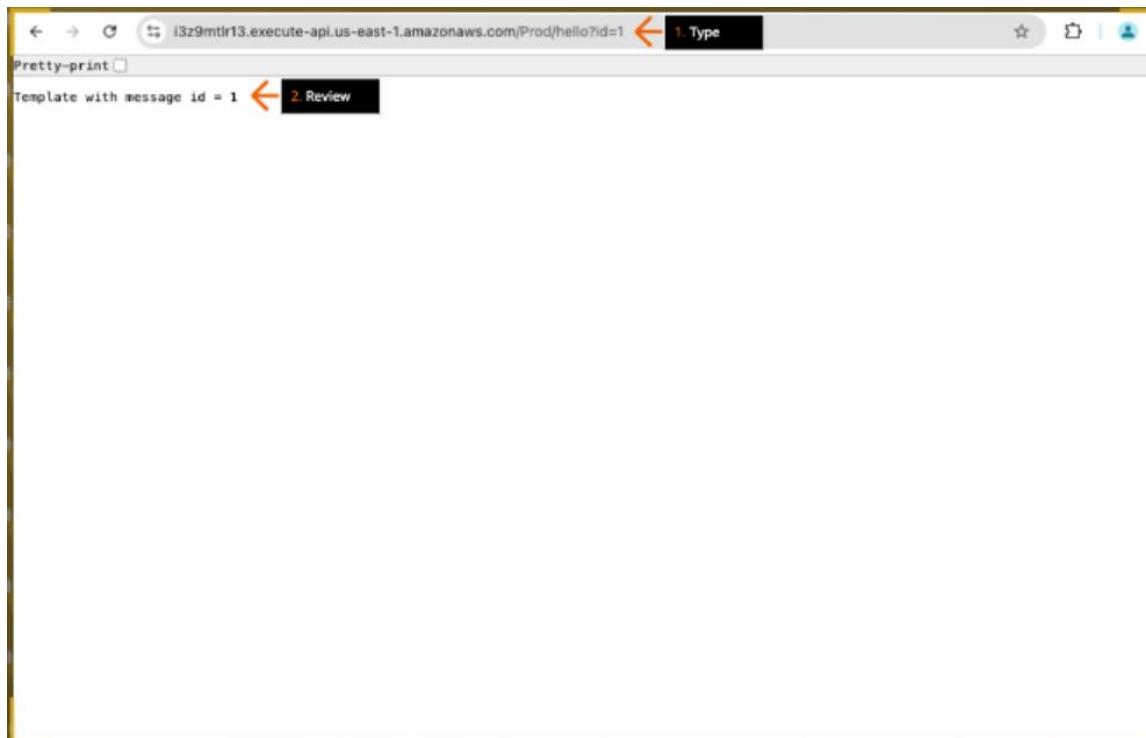


34. 1. In a new browser tab address bar, paste the HelloWorldApi API Gateway endpoint URL that you copied in an earlier step and press Enter.
2. Review the returned message.

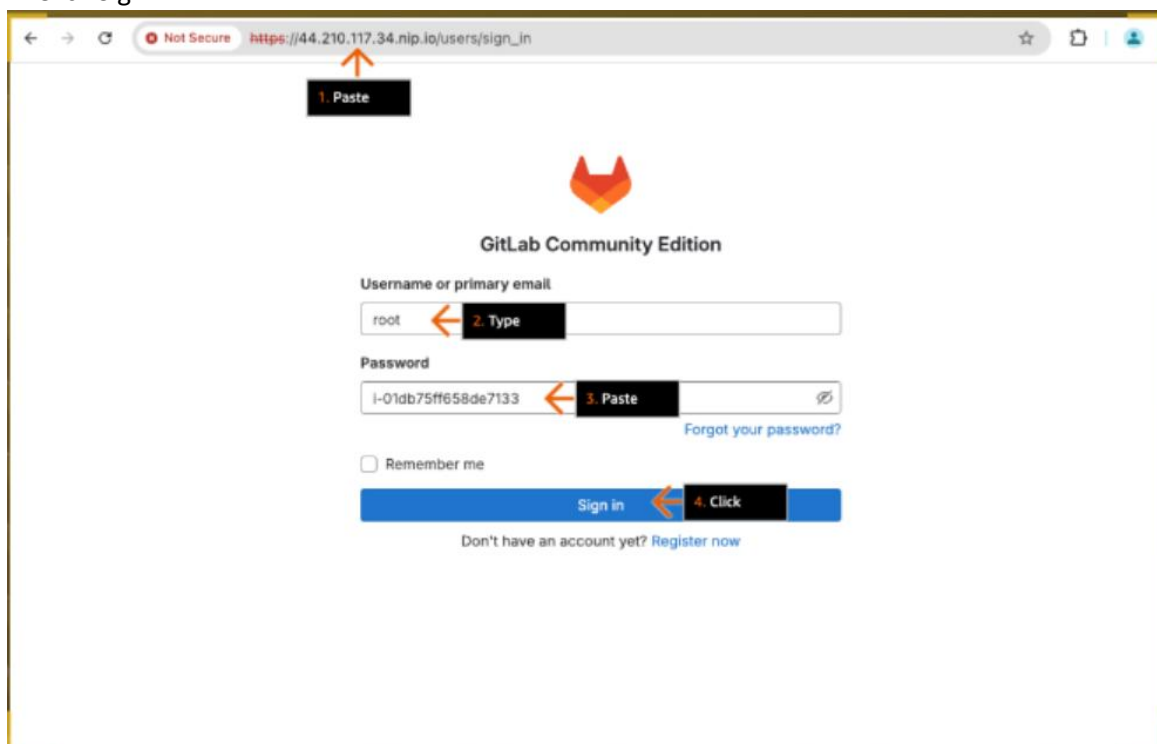


35. 1. At the end of the URL, type: `?id=1` and press Enter.
- Your full URL should look similar to the screenshot example.
2. Review the returned message.



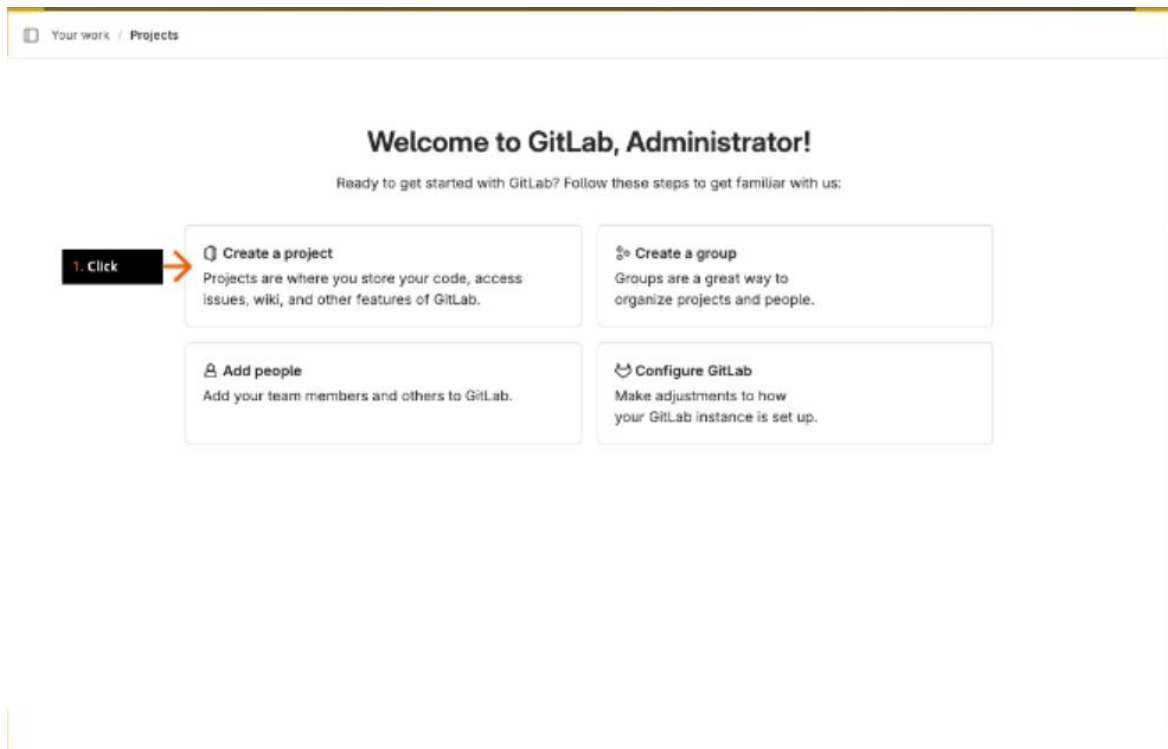


36. - In the next steps, you create a new code repository in GitLab self-managed on EC2, and push new code to the repository.
1. In a new browser tab address bar, paste the GitLabInstanceURL that you copied in an earlier step and press Enter.
  2. For Username or primary email, type:  
root
  3. For Password, paste the GitLabPassword value that you copied earlier.
  4. Click Sign in.

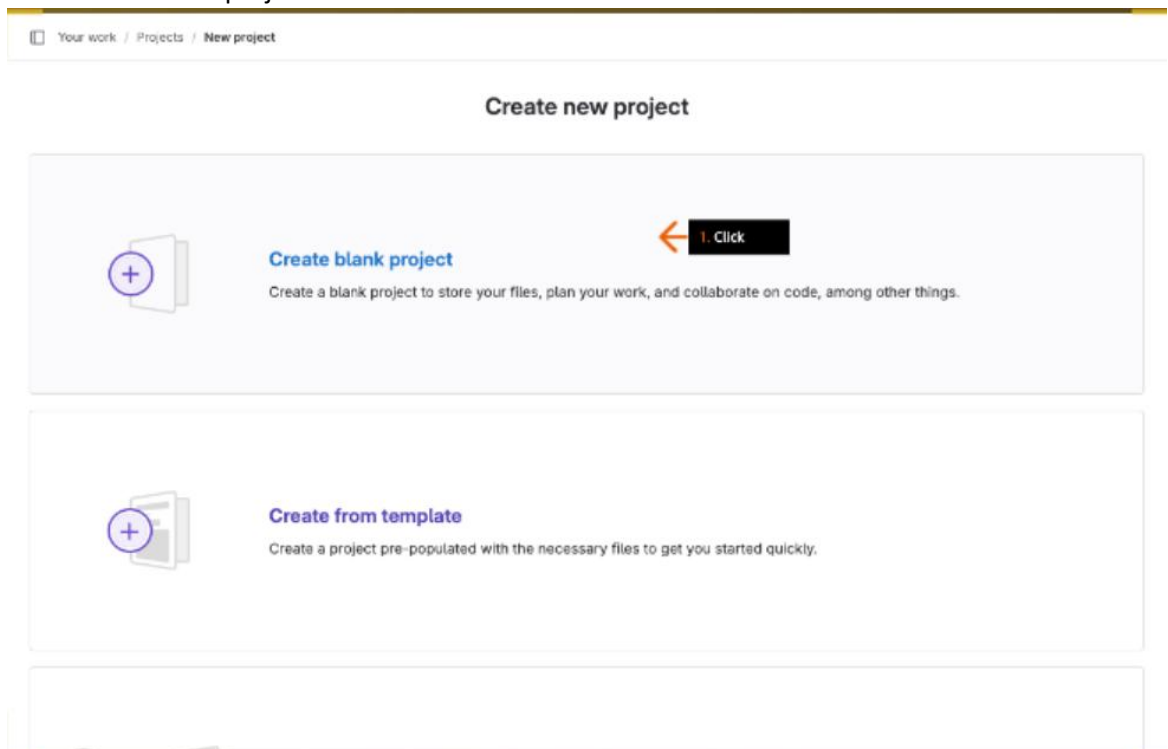


37. Click Create a project.





38. Click Create blank project.



39. 1. For Project name, type:  
sam\_application  
2. For Project URL, 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.

Your work / Projects / New project / Create blank project

## Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**  
 **1. Type**

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

**Project URL**  
  **2. Choose**

**Project slug**

**Visibility Level** **3. Review**

☒ Private  
 Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Internal  
 The project can be accessed by any logged in user except external users.

☐ Public  
 The project can be accessed without any authentication.

**Project Configuration**

☐ Initialize repository with a README **4. Clear**  
 Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ **5. Click** Application Security Testing (SAST)  
 Analyze your source code for known security vulnerabilities. [Learn more.](#)

40. 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.

Administrator / sam\_application

Project 'sam\_application' was successfully created.

## S sam\_application

**Code** **1. Click**

**Clone with SSH**  
 git@44.210.117.34.nip.io:root/sa

**Clone with HTTPs**  
 https://44.210.117.34.nip.io/roo **2. Click**

**Open in your IDE**  
 Visual Studio Code (SSH)  
 Visual Studio Code (HTTPS)  
 IntelliJ IDEA (SSH)  
 IntelliJ IDEA (HTTPS)

**Invite your team**  
 Add members to this project and start collaborating with your team.

**Upload File**  
 + New file  
 + Add README  
 + Add LICENSE  
 + Add CHANGELOG  
 + Add CONTRIBUTING  
 + Add Wiki  
 + Configure Integrations

**Command line instructions**  
 You can also upload existing files from your computer using the i

**Git global setup**  

```
git config --global user.name "Administrator"
git config --global user.email "gitlab_admin_bBoc"
```

**Create a new repository**  

```
git clone https://44.210.117.34.nip.io/root/sam_application.git
cd sam_application
git switch --create main
touch README.md
git add README.md
git commit -m "add README"
git push --set-upstream origin main
```

**Created on**  
 October 15, 2024

41. 1. Return to the VS Code IDE browser tab.
2. In the bottom terminal window, to check if you are in the HelloWorld folder, run:  
 pwd
3. To initiate a git repository in the HelloWorld folder, run:  
 git init
4. To add all files in the working directory to the staging area, run:  
 git add .  
 - Make sure to include the period (.) at the end.
5. To add a commit message, run:  
 git commit -m 'first commit'

```
5
6 # Invoke Lambda function

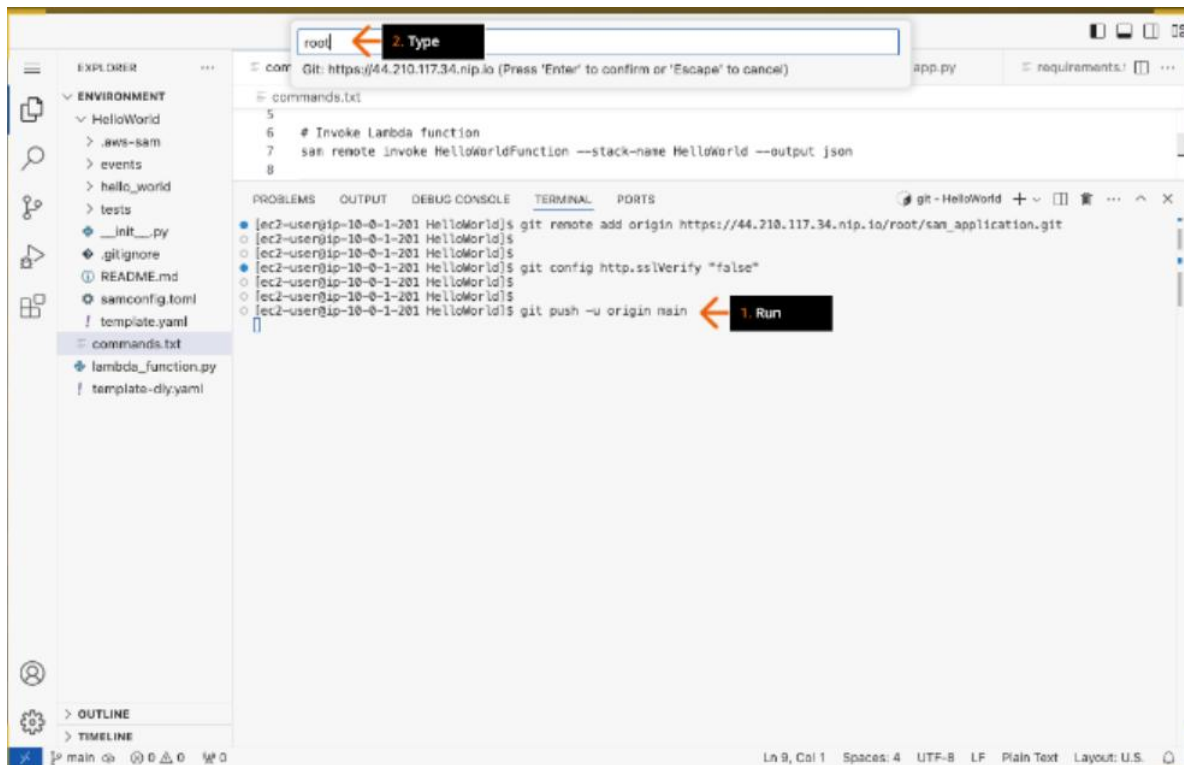
[ec2-user@ip-10-0-1-201 HelloWorld]$ pwd
/home/ec2-user/environment/HelloWorld
[ec2-user@ip-10-0-1-201 HelloWorld]$
[ec2-user@ip-10-0-1-201 HelloWorld]$ git init
Initialized empty Git repository in /home/ec2-user/environment/HelloWorld/.git/
[ec2-user@ip-10-0-1-201 HelloWorld]$
[ec2-user@ip-10-0-1-201 HelloWorld]$ git add .
[ec2-user@ip-10-0-1-201 HelloWorld]$
[ec2-user@ip-10-0-1-201 HelloWorld]$ git commit -m 'first commit'
[master (root-commit) 1264c8b] first commit
123 files changed, 43033 insertions(+)
create mode 100644 .aws-sam/build.toml
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi-2024.8.30.dist-info/LICENSE
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi-2024.8.30.dist-info/METADATA
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi-2024.8.30.dist-info/RECORD
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi-2024.8.30.dist-info/WHEEL
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi/_init_.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi/_main_.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi/cacert.pem
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi/core.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/certifi/py.typed
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer-3.4.0.dist-info/LICENSE
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer-3.4.0.dist-info/METADATA
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer-3.4.0.dist-info/RECORD
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer-3.4.0.dist-info/WHEEL
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer/_init_.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer/_main_.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer/api.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer/cli/_init_.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer/cli/_main_.py
create mode 100644 .aws-sam/deps/27efba5f-a8ca-487f-a1a6-520fe71840ce/charset_normalizer/constant.py
```

42. 1. To add the remote repository, sam\_application, 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.

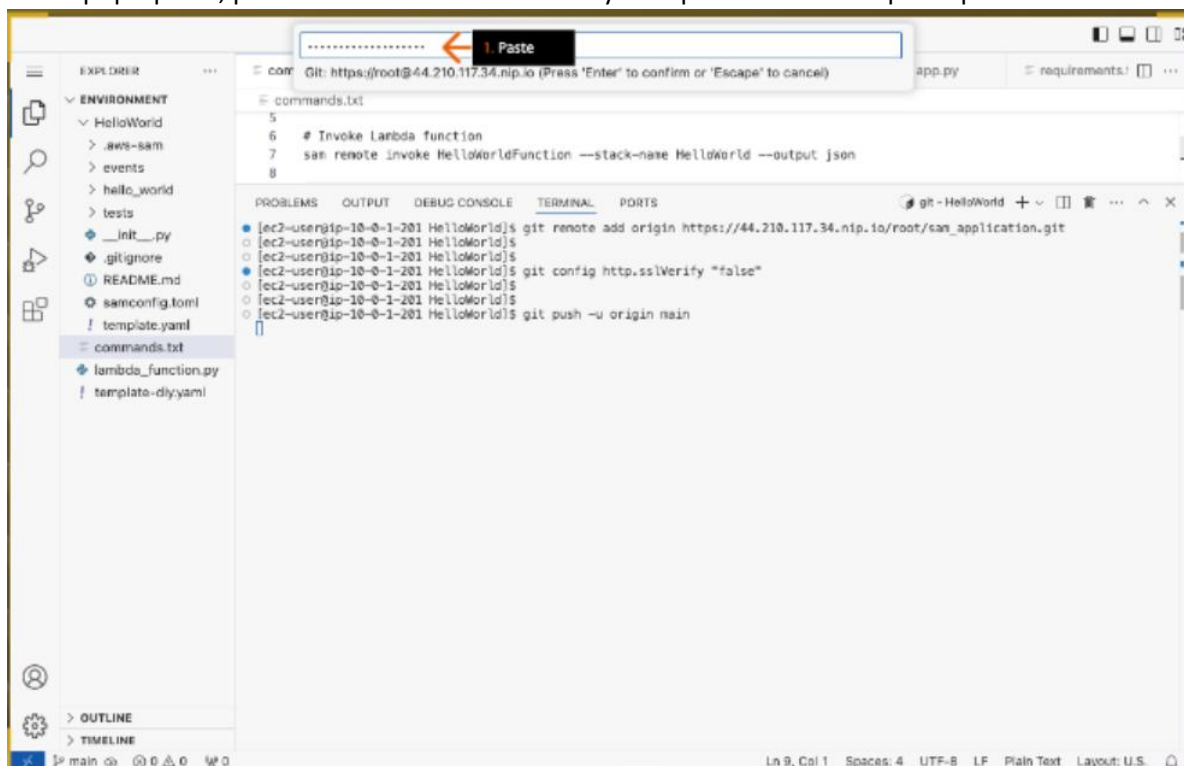
```
5
6 # Invoke Lambda function
7 sam remote invoke HelloWorldFunction --stack-name HelloWorld
8

[ec2-user@ip-10-0-1-201 HelloWorld]$ git remote add origin https://44.210.117.34.nip.io/root/sam_application.git
[ec2-user@ip-10-0-1-201 HelloWorld]$
[ec2-user@ip-10-0-1-201 HelloWorld]$ git config http.sslVerify "false"
[ec2-user@ip-10-0-1-201 HelloWorld]$
```

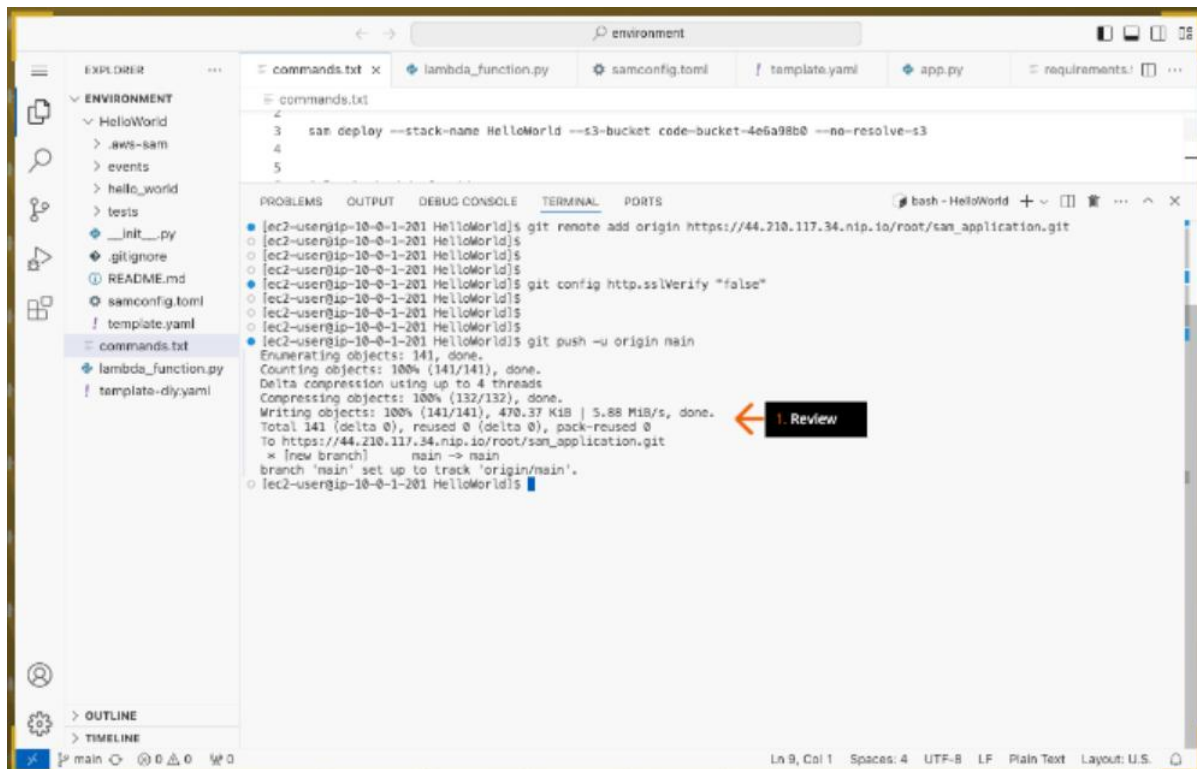
43. 1. To push the code to the Git repository, run:  
git push -u origin main  
2. In the pop-up box at the top of the IDE, to provide credentials to access the Git repository, type:  
root  
and press Enter.



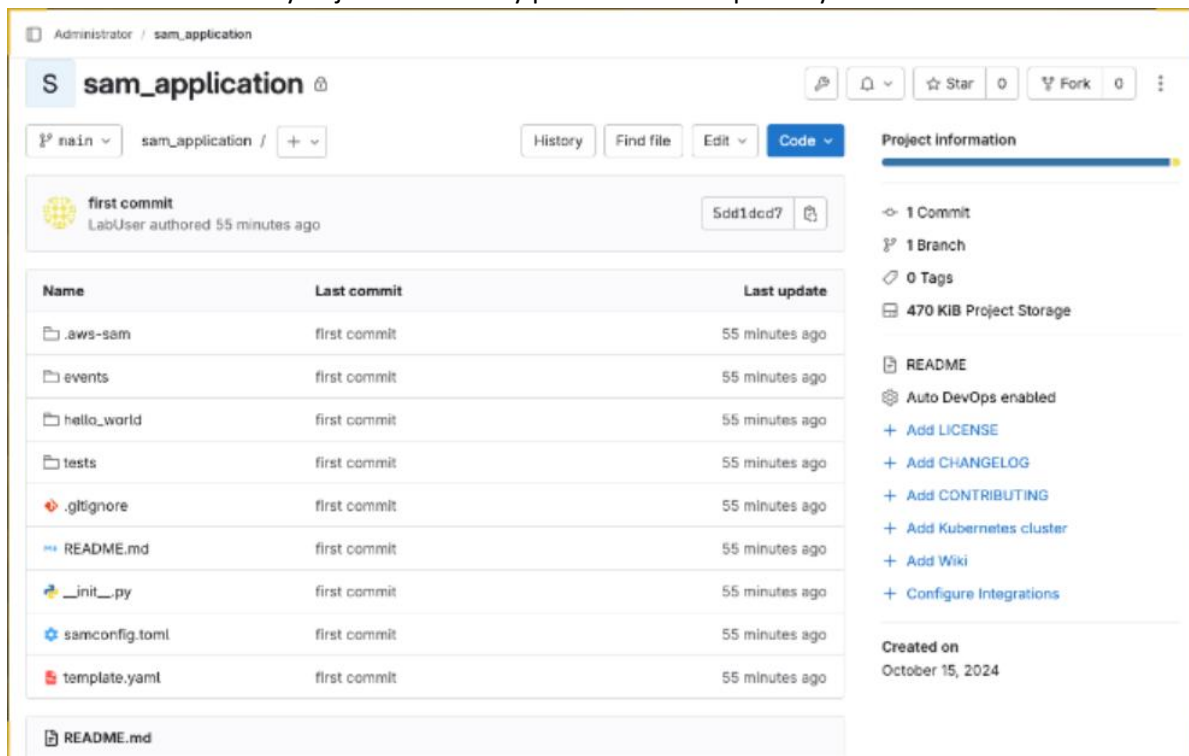
44. In the pop-up box, paste the GitLabPassword that you copied in earlier step and press Enter.



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



46. 1. Return to the sam\_application repository on the GitLab self-managed browser tab.
2. Review the files that you just successfully pushed to this repository.



DYI:

- Add a DynamoDB table to the AWS
- SAM template.
- Deploy the AWS SAM template.



1. Add these lines in template.yaml under resources section.

```
SimpleTable:
  Type: AWS::Serverless::SimpleTable
  Properties:
    TableName: app_table
```

```
16 Resources:
17   HelloWorldFunction:
18     Type: AWS::Serverless::Function # More info about Function
19     Properties:
20       CodeUri: hello_world/
21       Handler: app.lambda_handler
22       Runtime: python3.9
23       Architectures:
24       - x86_64
25       Events:
26         HelloWorld:
27           Type: Api # More info about API Event Source: https
28           Properties:
29             Path: /hello
30             Method: get
31   SimpleTable:
32     Type: AWS::Serverless::SimpleTable
33     Properties:
34     TableName: app_table
35
```

2. To validate that the template.yaml file is valid, run:  
sam validate
3. To build your application and prepare for deployment, run:  
sam build

```
Error: No changes to deploy. Stack HelloWorld is up to date
[ec2-user@ip-10-0-0-245 HelloWorld]$ sam validate
/home/ec2-user/environment/HelloWorld/template.yaml is a valid SAM Template
[ec2-user@ip-10-0-0-245 HelloWorld]$ sam build
Starting Build use cache
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /home/ec2-user/environment/HelloWorld/hello_world runtime: python3.9 architecture: x86_64
functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource
Build Succeeded
```

4. Deploy the stack again.

```
[*] Deploy: sam deploy --guided
[ec2-user@ip-10-0-0-245 HelloWorld]$ sam deploy --stack-name HelloWorld --s3-bucket code-bucket-d56e4f80 --no-resolve-s3
File with same data already exists at 8f806e0950f71b4e7fc1672298ddaab8, skipping upload

Deploying with following values
=====
Stack name           : HelloWorld
Region              : us-east-1
Confirm changeset    : True
Disable rollback     : False
Deployment s3 bucket  : code-bucket-d56e4f80
Capabilities         : ["CAPABILITY_IAM"]
Parameter overrides  : {}
Signing Profiles     : {}
```

5. For "Deploy this changeset?", type:  
y

and press Enter.

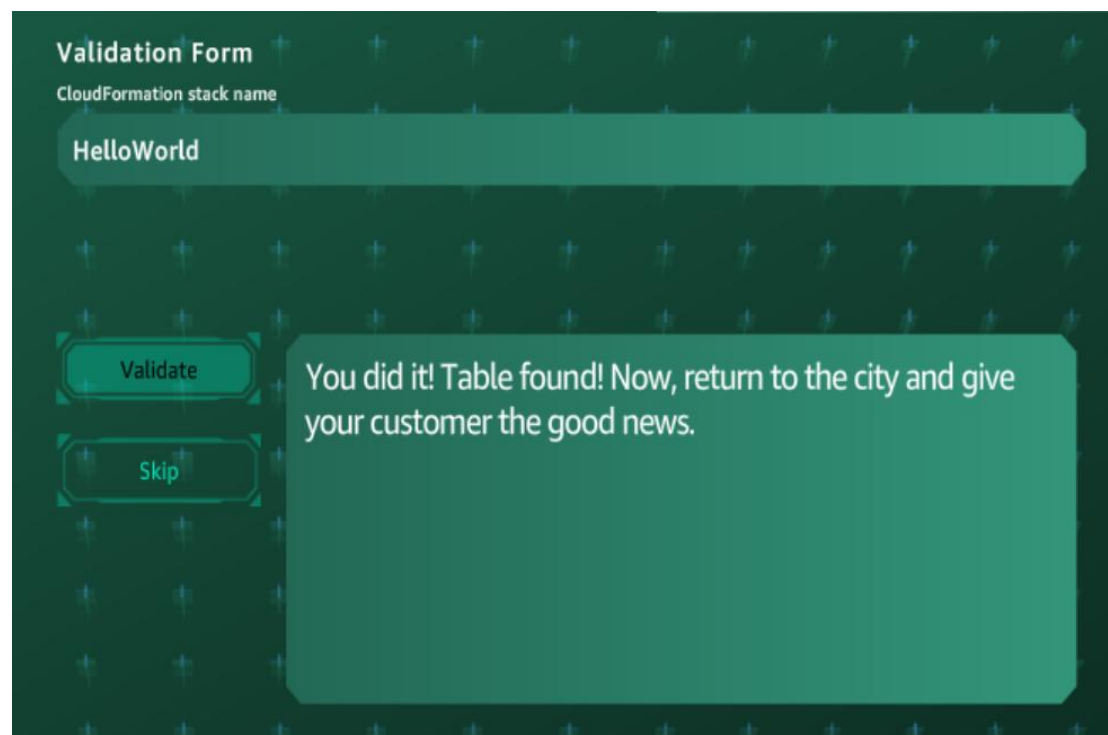
Review the CloudFormation events while the stack is creating.

```
Key           HelloWorldApi
Description   API Gateway endpoint URL for Prod stage for Hello World function
Value         https://cooctd27j0.execute-api.us-east-1.amazonaws.com/Prod/hello/

Key           HelloWorldFunction
Description   Hello World Lambda Function ARN
Value         arn:aws:lambda:us-east-1:968644191859:function:HelloWorld-HelloWorldFunction-k1UX8PvWFX1H
-----

Successfully created/updated stack - HelloWorld in us-east-1
[ec2-user@ip-10-0-0-245 HelloWorld]$
```

6. Validate the solution by mentioning the CloudFormation stack name.



The screenshot shows a 'Validation Form' with a dark green background and a pattern of small white stars. The form has a title 'Validation Form' and a label 'CloudFormation stack name'. Below the label is a text input field containing the text 'HelloWorld'. To the left of the input field are two buttons: 'Validate' and 'Skip'. To the right of the input field is a large green box containing the text: 'You did it! Table found! Now, return to the city and give your customer the good news.'

7. The CloudFormation stack is created. Thus, the solution is completed.

