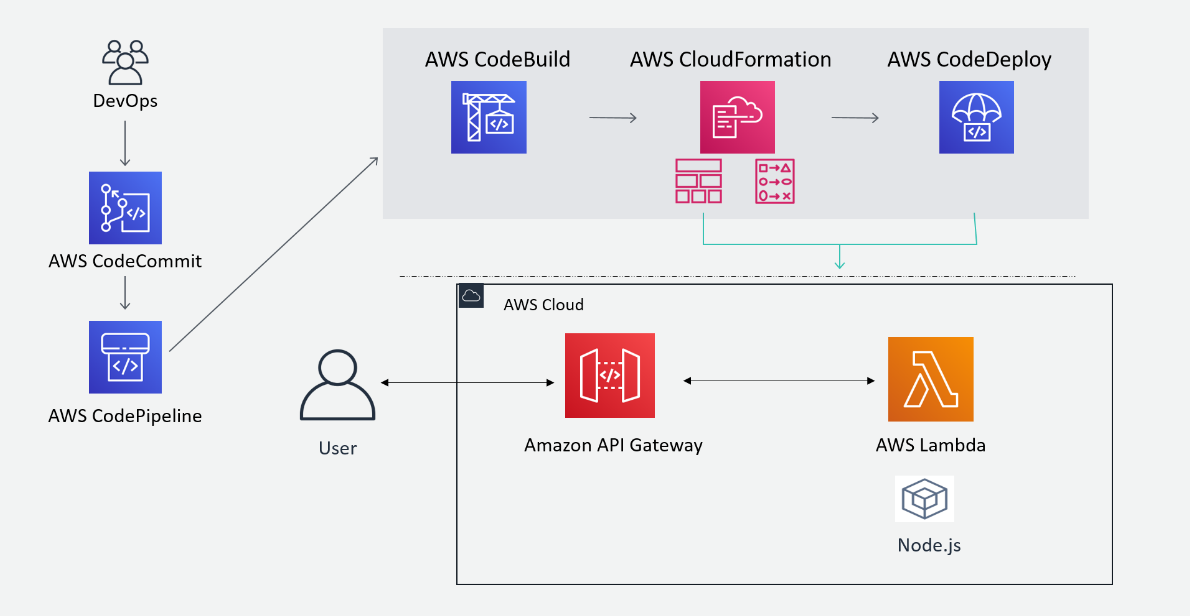
## Lab 4: Deploying a serverless application using AWS SAM and a CI/CD pipeline

Objectives:

1. Build and locally test your application.
2. Use AWS SAM to publish a serverless application.
3. Automate deployment with AWS CodeDeploy and AWS CodePipeline.
4. Implement traffic shifting deployment to gradually roll out updates.

Overview Diagram:  


Explanation:

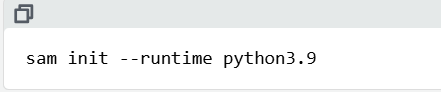
1. Code Commit: The DevOps team pushes code changes to the AWS CodeCommit repository.
2. Pipeline Trigger: AWS CodePipeline detects the change in the CodeCommit repository and starts the pipeline.
3. Build Stage: CodePipeline triggers AWS CodeBuild to compile the code, run tests, and prepare the deployment package.
4. Infrastructure Provisioning: After a successful build, AWS CodePipeline uses AWS CloudFormation to provision or update the necessary infrastructure, ensuring all resources (API Gateway, Lambda functions) are correctly set up.
5. Deployment: AWS CodePipeline then triggers AWS CodeDeploy to deploy the new version of the application.
6. AWS CodeDeploy

A service that automates application deployments to various compute services, including AWS Lambda. It can implement a **traffic shifting strategy**, gradually shifting traffic to the new version to ensure stability and allow for quick rollback if needed.

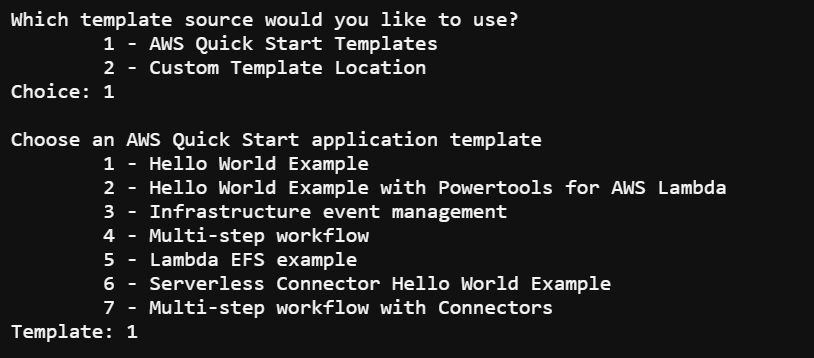
1. User Interaction: The end user interacts with the application via the Amazon API Gateway, which routes the request to the AWS Lambda function.
2. The Lambda function processes the request and returns a response to the user.

**Objective 1 - AWS SAM**

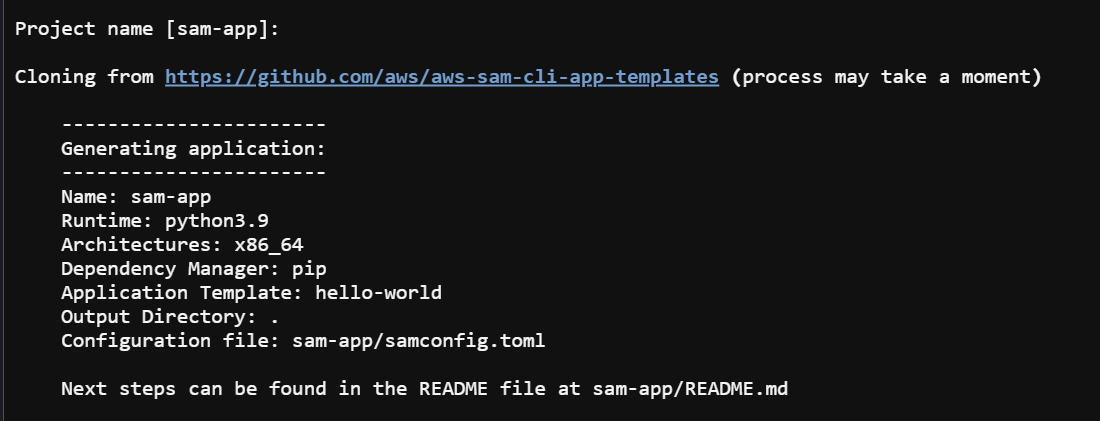
1.1 Open Cloud9 IDE and run the following command to initialize a SAM project



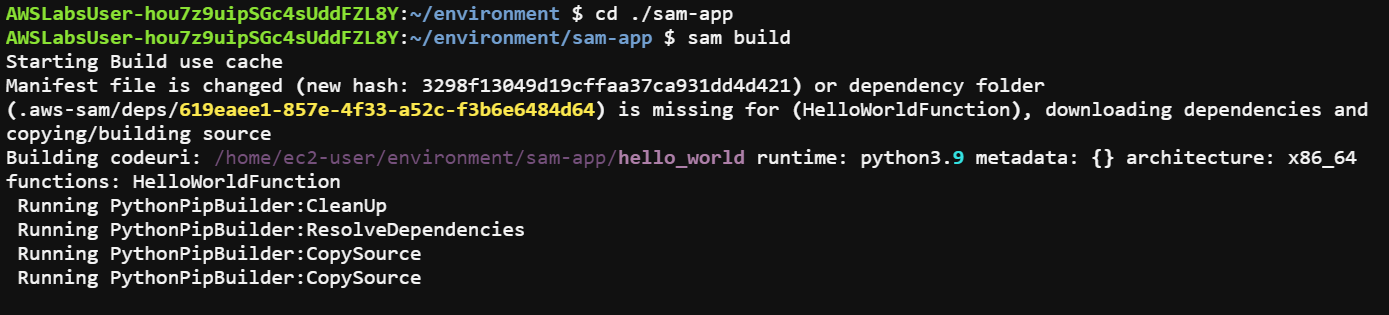
1.2 Choose the options to get a hello world example project from AWS quick start templates, and choose the name - ‘sam-app’



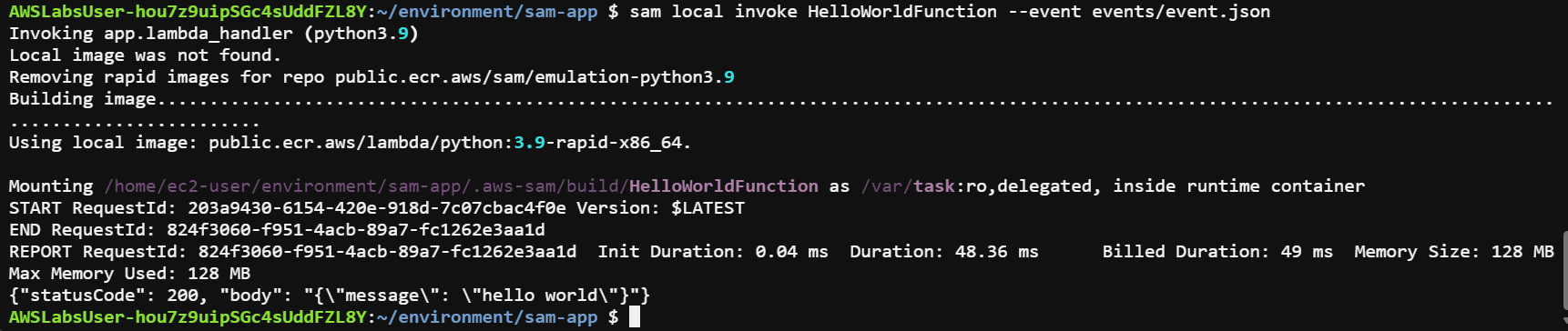
1.3 The app will be cloned from git



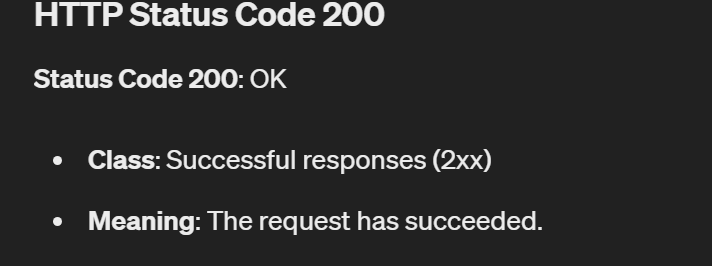
1.4 Go to the app directory and perform the build (of the app)



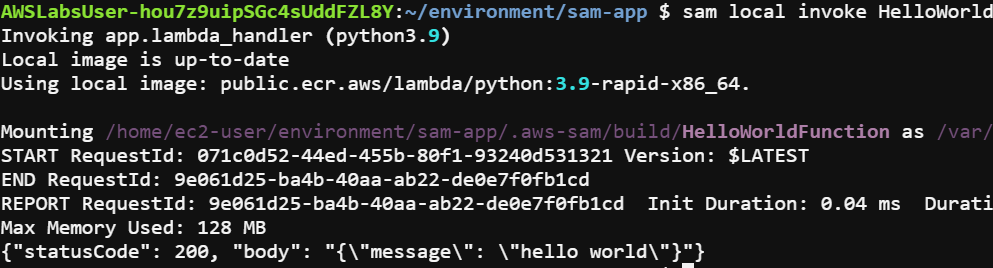
1.5 Test your application locally. Use AWS SAM to test your application, when reviewing the test output, look for a status code 200



Meaning of SC 200-



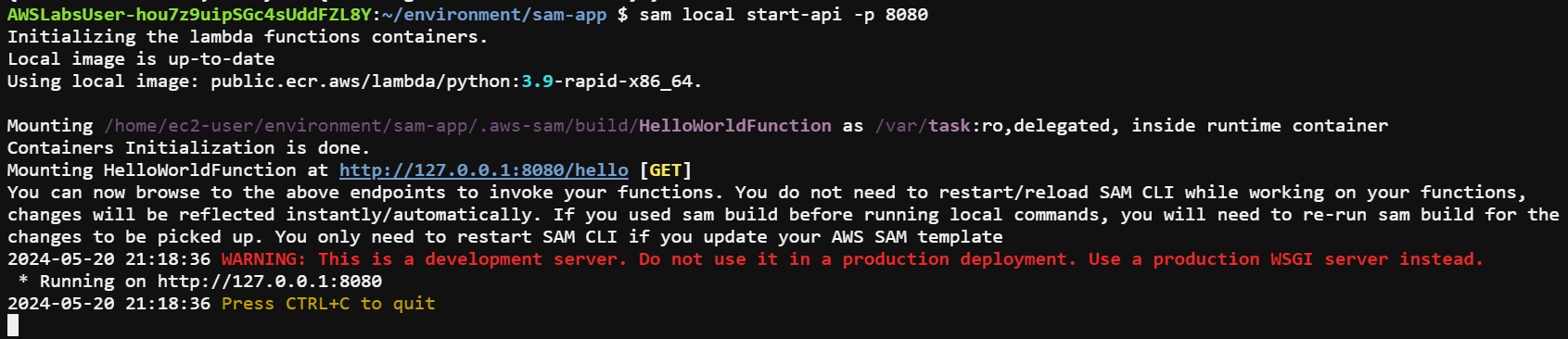
1.6 Run the following command to test the application



It again produces SC200

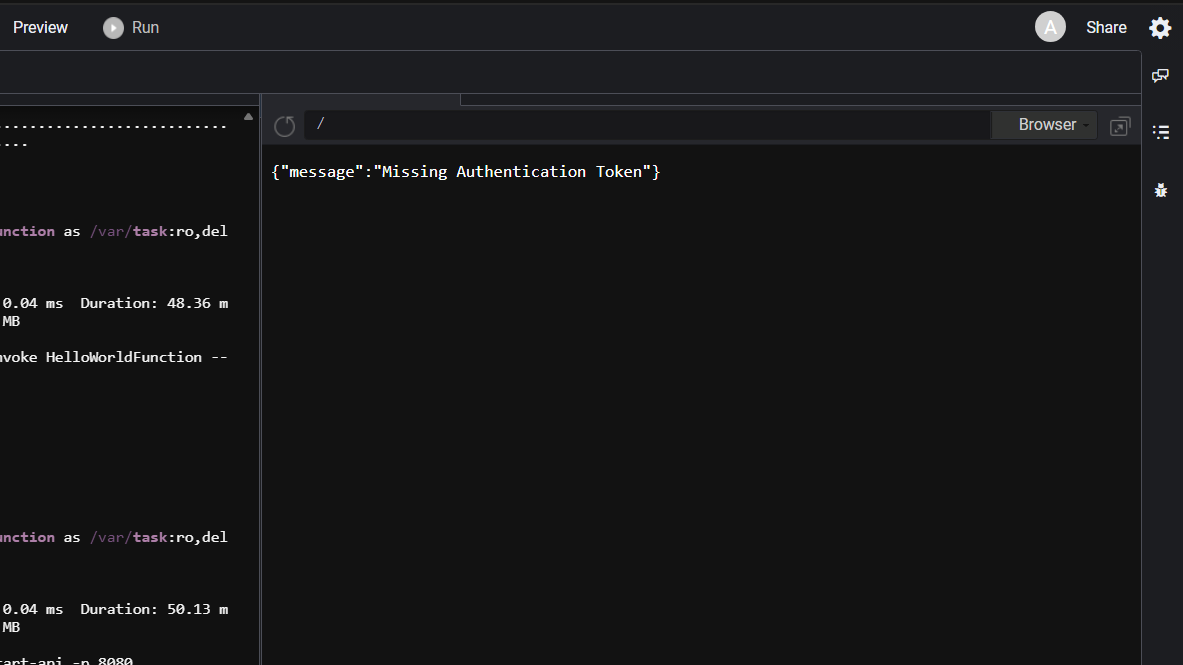
Meaning test has been successful

1.7 Start the application



Now: The start-api command directs you to browse to port 8080 on the local system loopback address (127.0.0.1). AWS Cloud9 does not allow this and you receive an error if you use the address included in the command output. AWS Cloud9 does offer a function to securely view your application through the Preview feature.

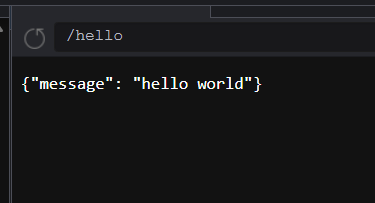
1.8 Preview the app, by clicking on preview option on top of Cloud9 UI



This is the expected output as per Lab instructions

1.9 Enter “hello” at the end of address bar





App run Test - successful

**Objective 2 - USING AWS SAM TO DEPLOY THE APPLICATION**

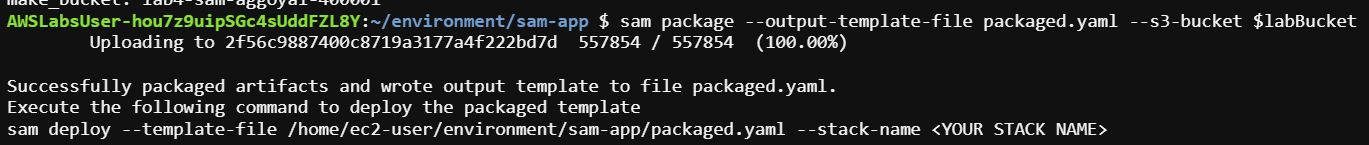
2.1 To deploy the hello world application the first step is to create an (Amazon S3) bucket to host your application.



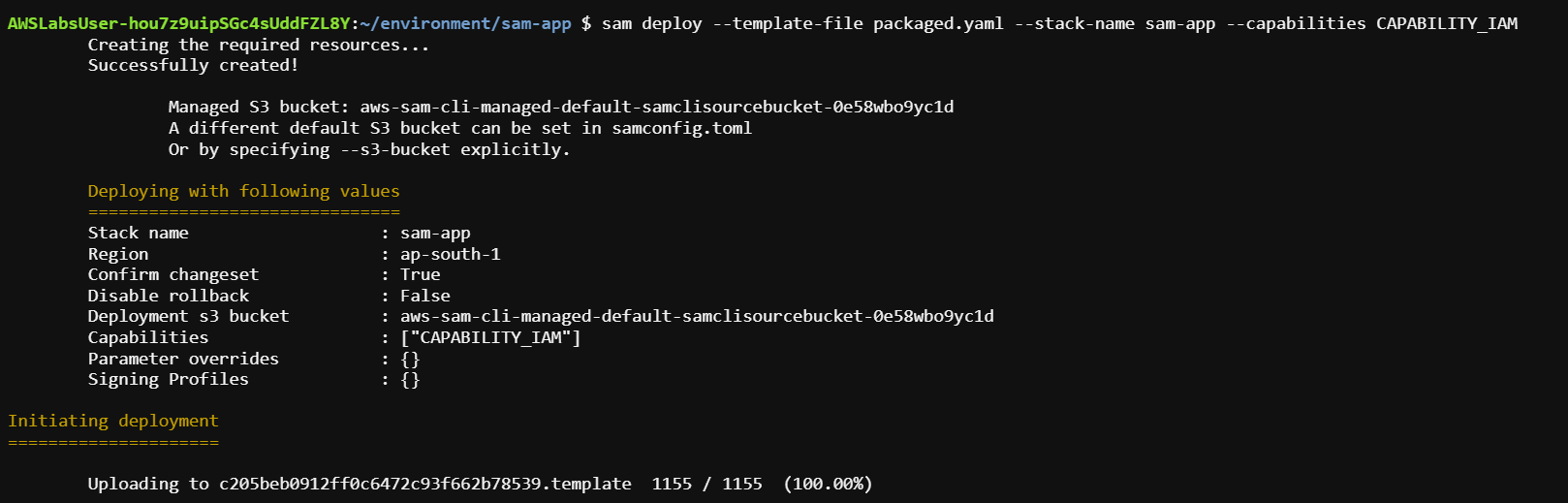
2.2 Use the variable to create a globally unique bucket



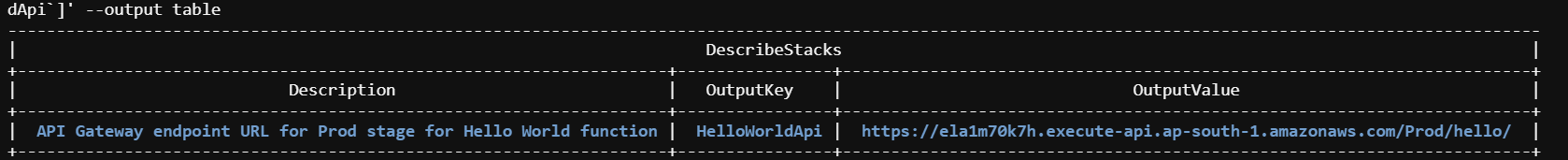
2.3 Run the following command to package your application and push it to the S3 bucket you created



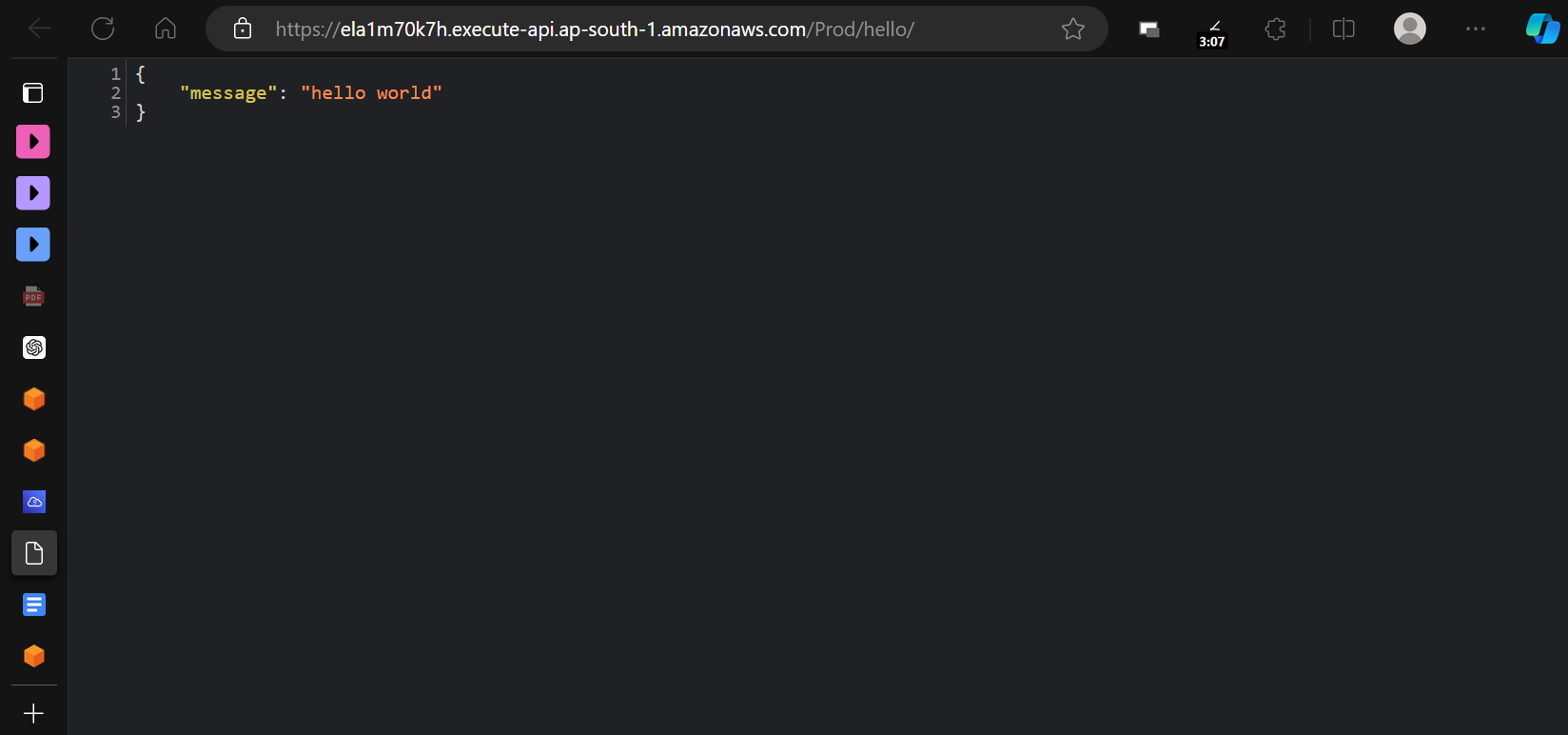
2.4 Run the command below to deploy your application package







2.5 Click on the link, in a new tab

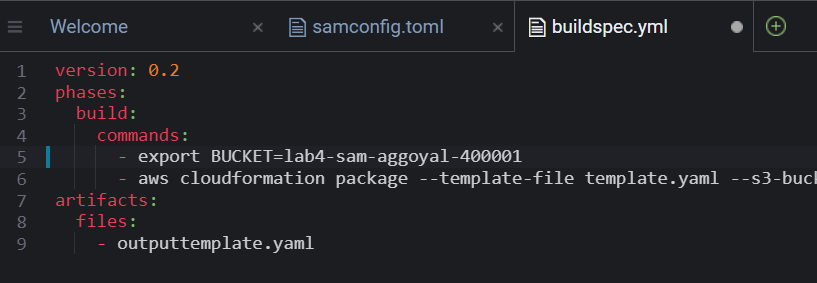


Deployed, successfully

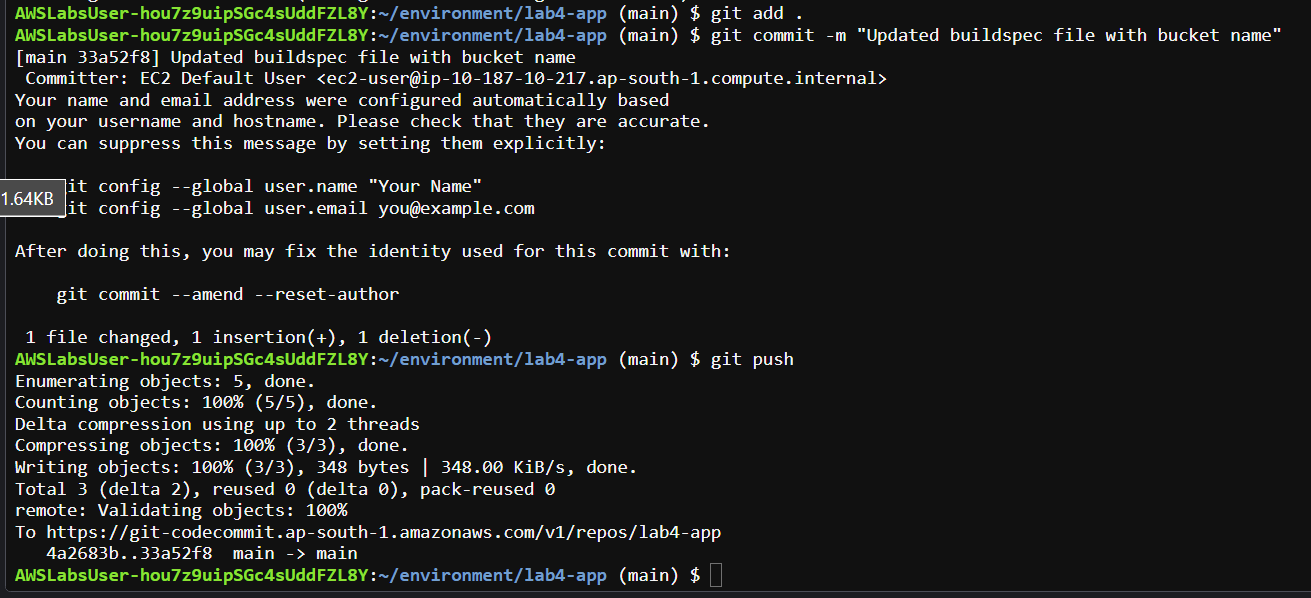
**Objective 3: Automate application deployment with AWS CodePipeline**

Note - For this task, you create a CI/CD pipeline to automate the deployment of an application. You create a three-stage pipeline that uses AWS CodeCommit, AWS CodeBuild, and AWS CloudFormation. Your initial HelloWorld application was initialized using Python as the language but you’ve chosen to change to Node.js for the deployment. You overwrite your existing SAM deployment with this new one. The new SAM project is in the Cloud9 environment. You need to add a new bucket for your deployment and make the appropriate code changes before you can automate the deployment.

3.1 Go to the lab4 directory, verify the bucket name and put the bucket name in buildspec.yml file



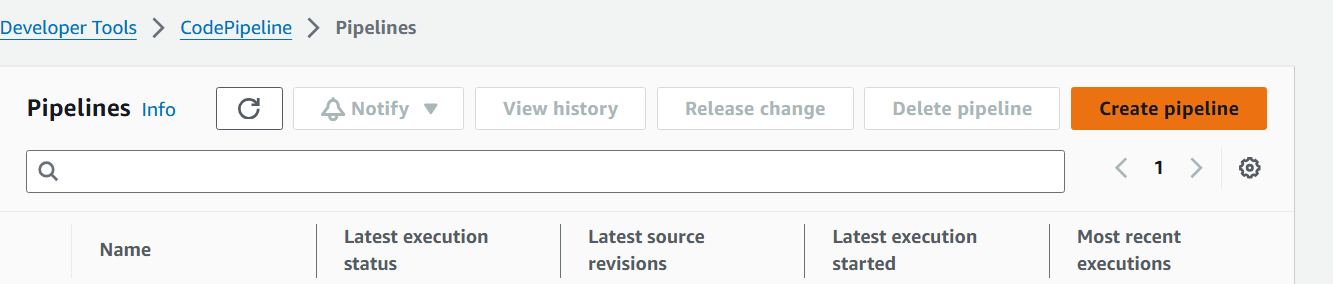
3.2 Update the remote AWS CodeCommit repository by running the following commands



**Objective 4 - CREATE THE CONTINUOUS DELIVERY PIPELINE FOR YOUR LAB4 APPLICATION**

4.1 Open AWSCodePipeline for this

4.2 Create Pipeline, using the following SSs:

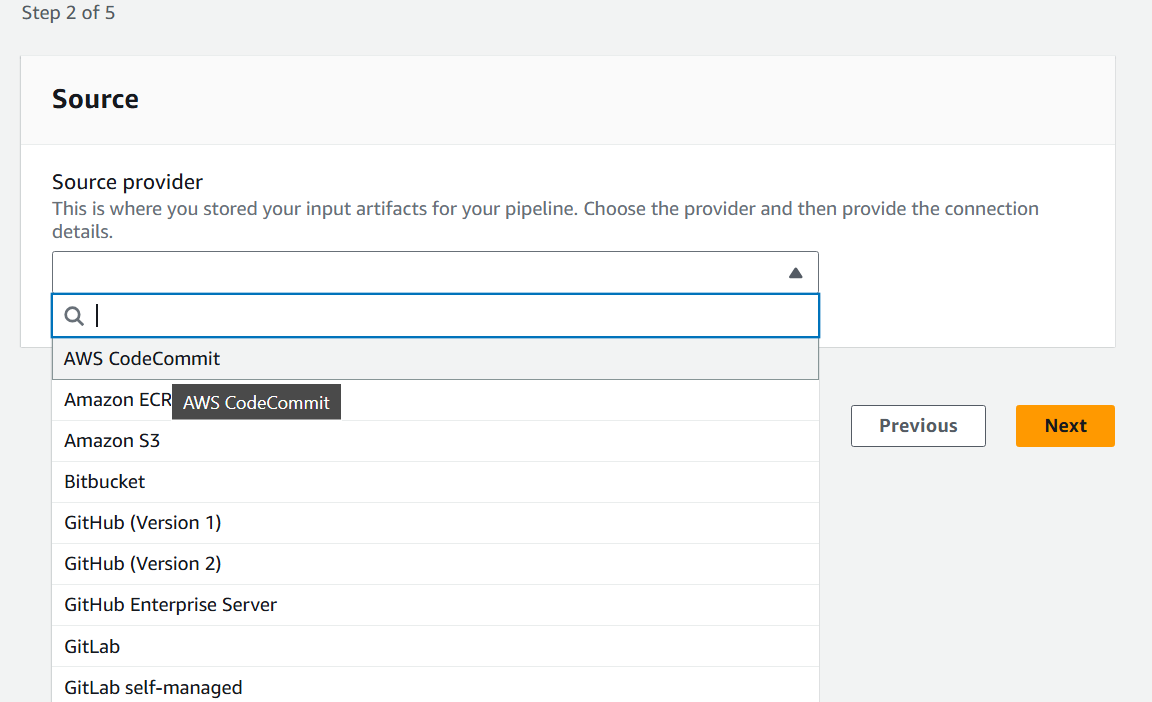




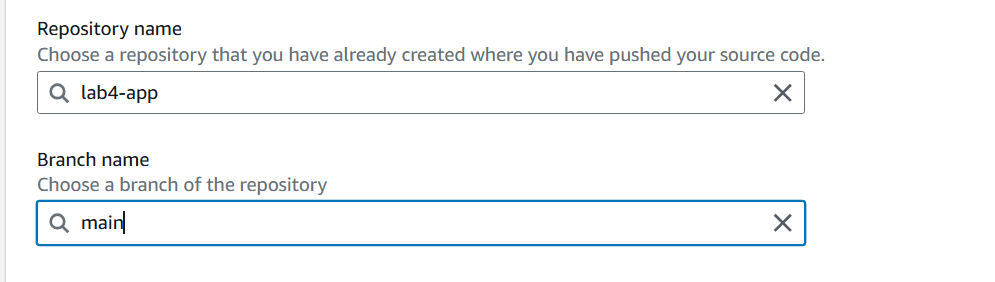
Let all values remain the same

Click next

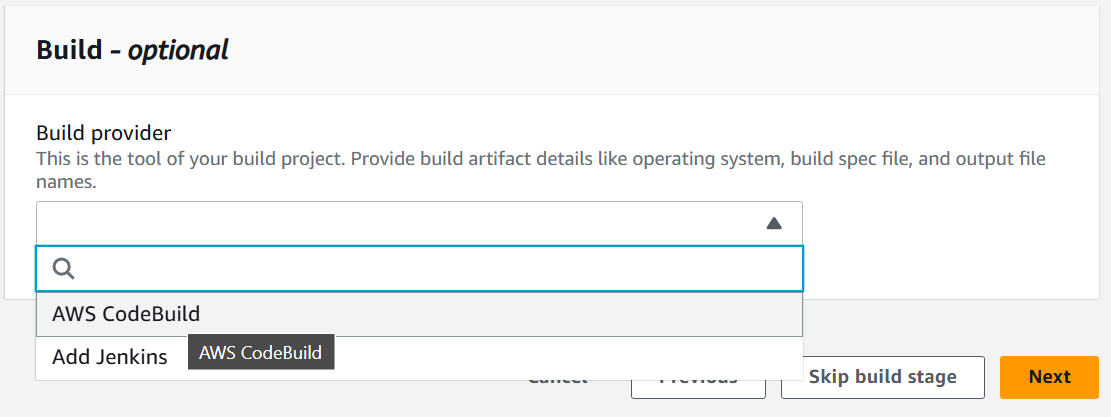
4.3 choose AWS CodeCommit from the drop-down menu for Source provider.



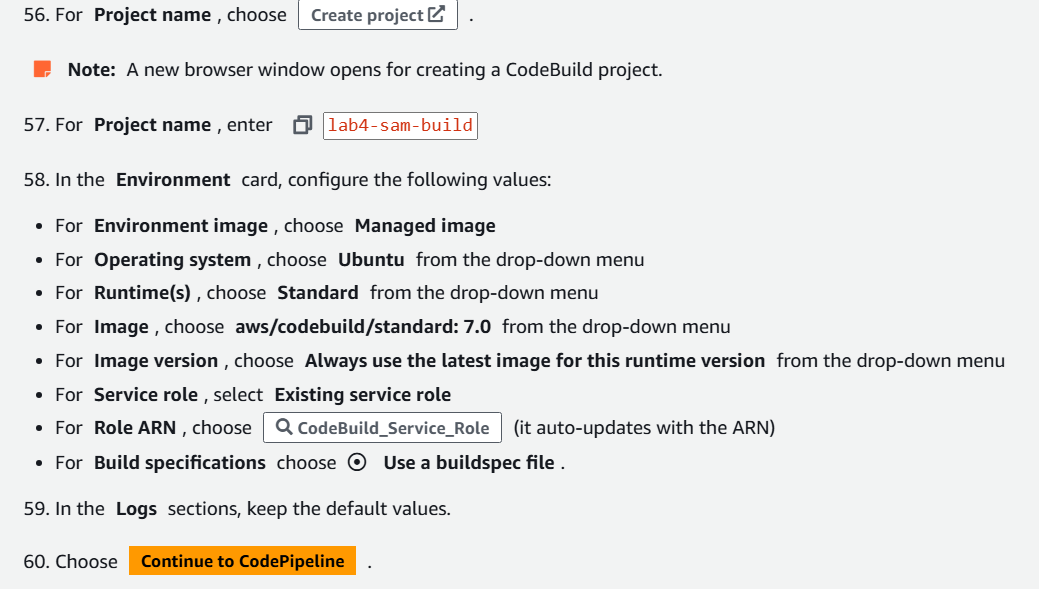
4.4 Update repo and branch name

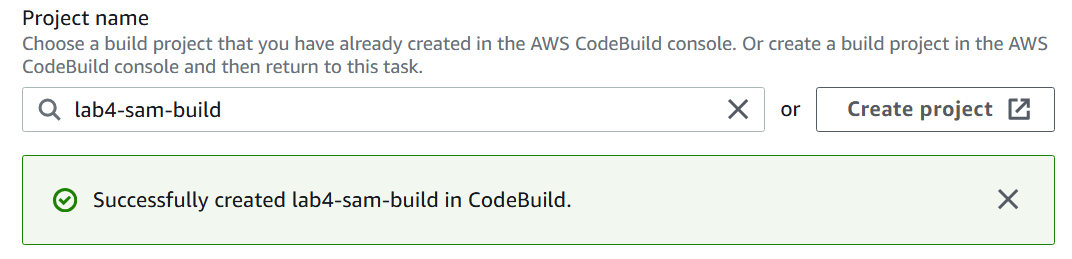


4.5 Choose AWS CodeBuild in build stage

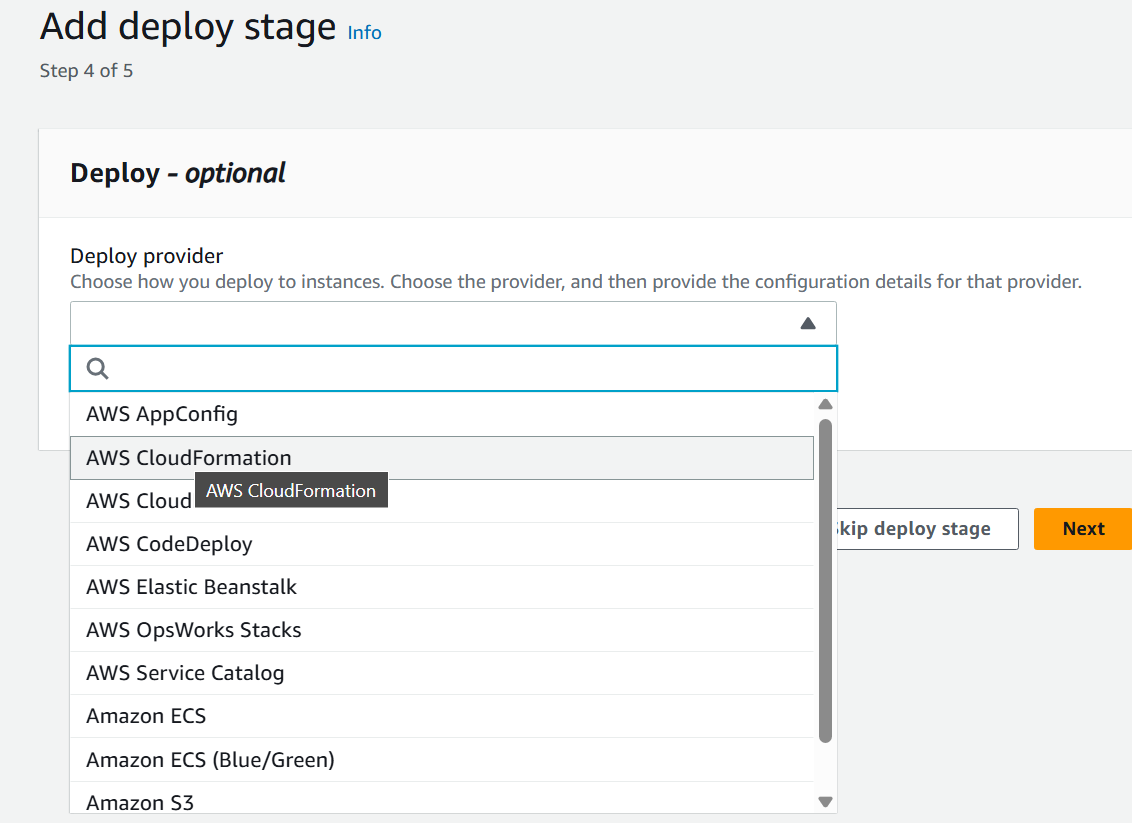


4.6 Do the following:



Output - 

4.7 For Deploy provider, choose AWS CloudFormation from the drop-down menu.

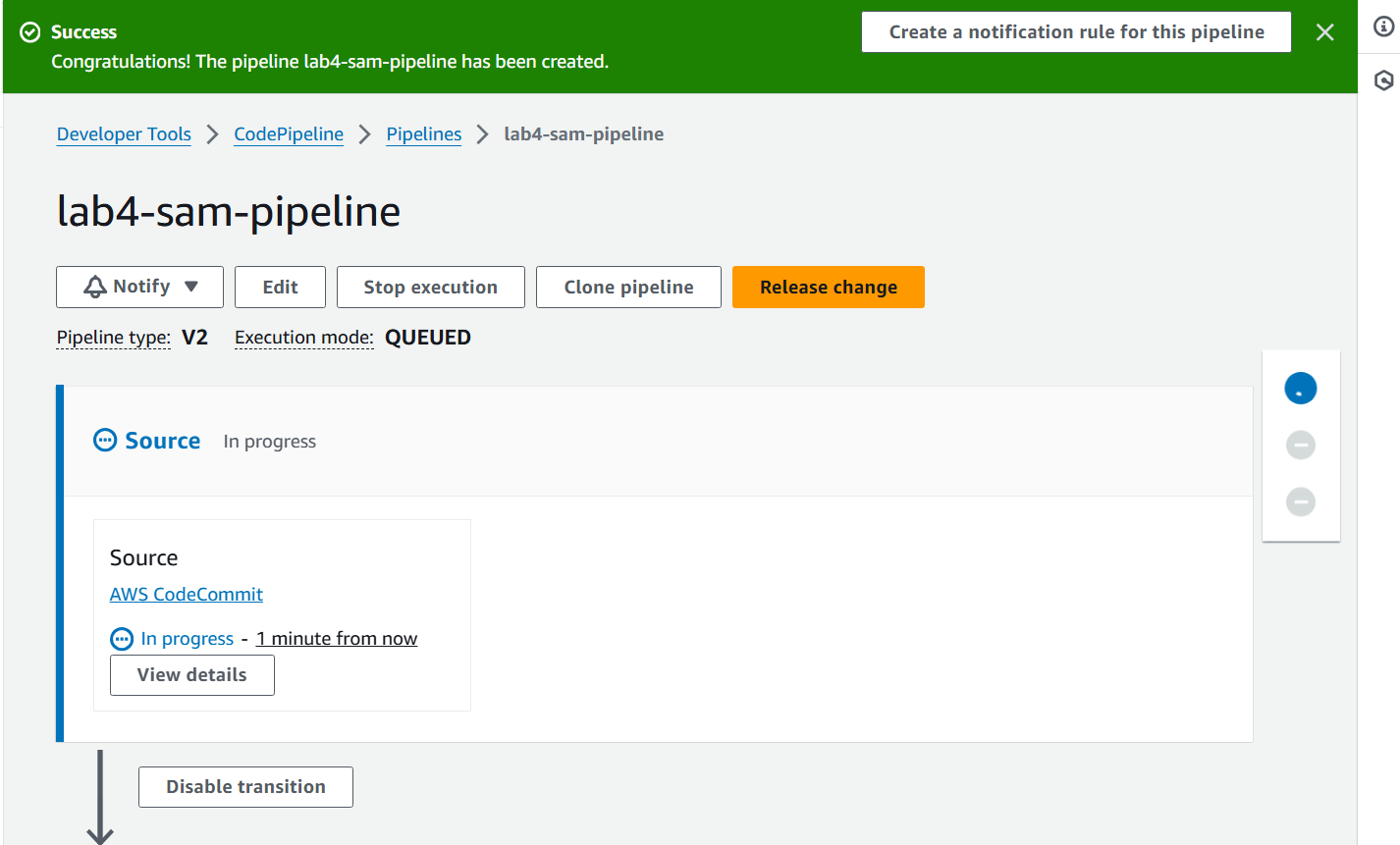


Note - 

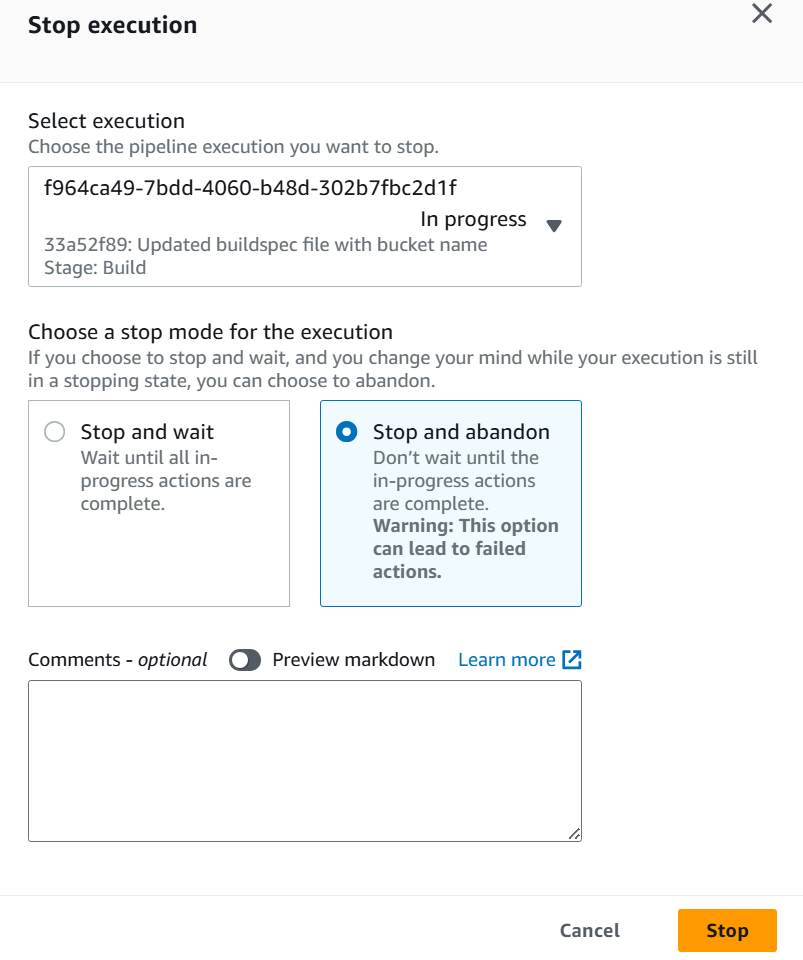
4.8 Review and create the pipeline

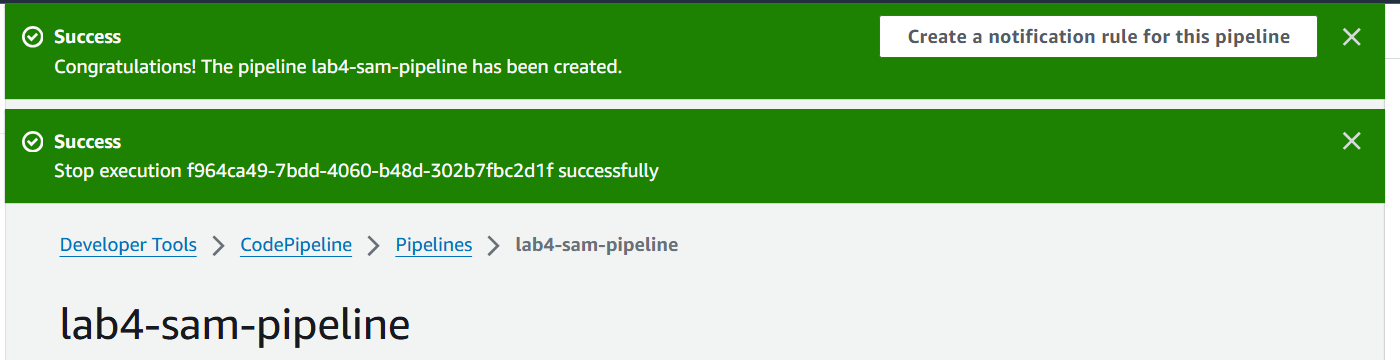
The pipeline lab4-sam-pipeline has been created.

4.9 Choose stop execution



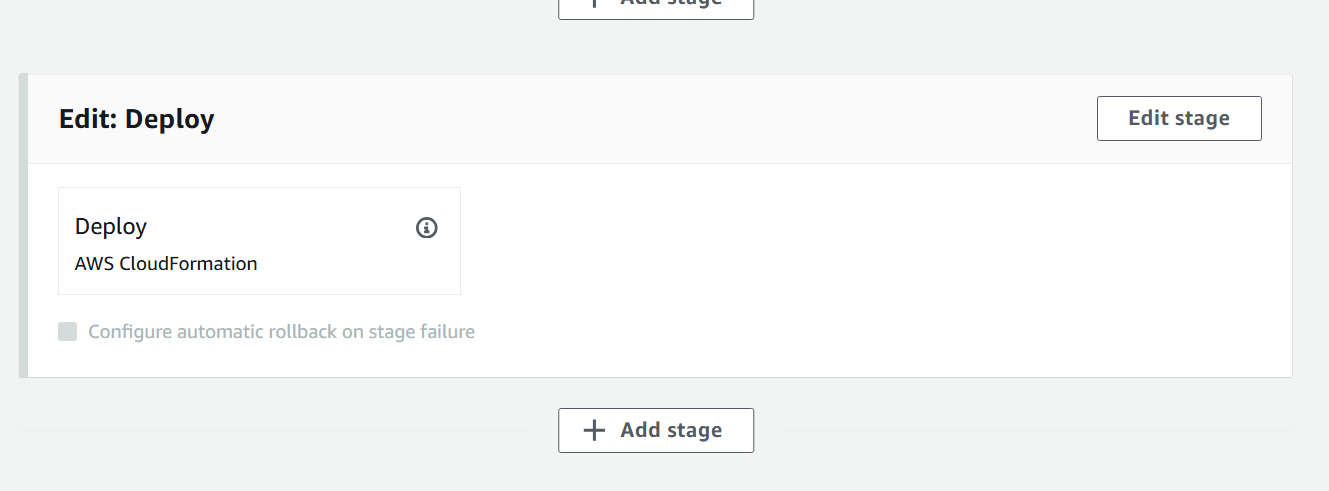
4.10 Select the default values



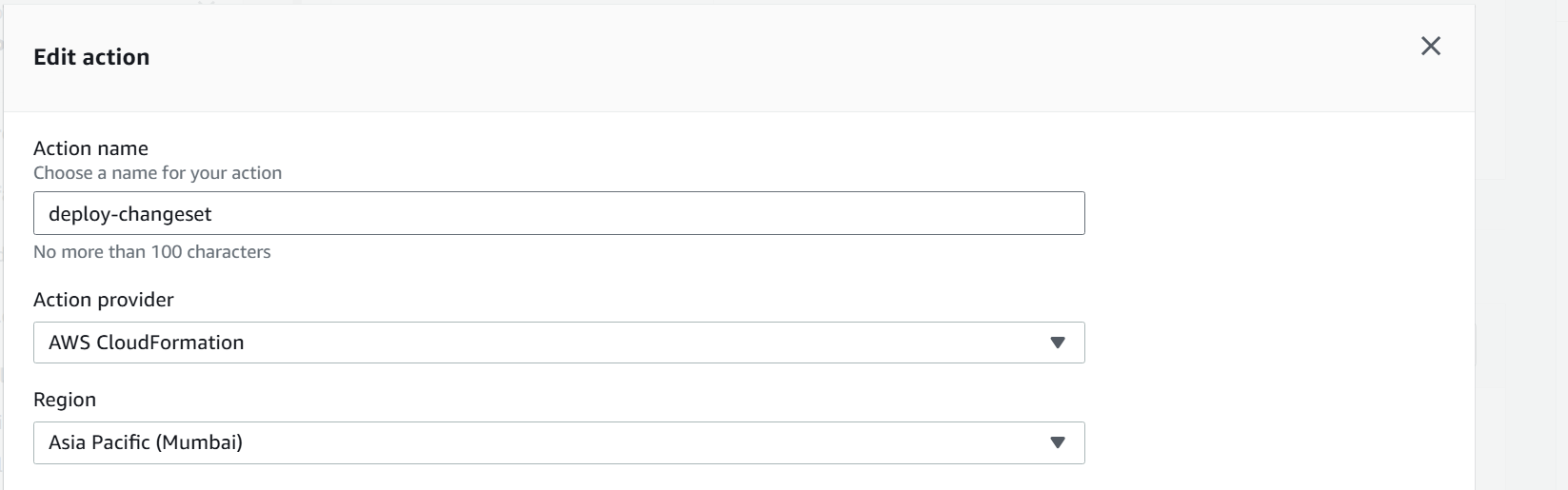
4.11 

4.12 Now you can complete your deploy stage. The current configuration creates an AWS CloudFormation change set but you still need to run it to shift traffic to the new version of your application.

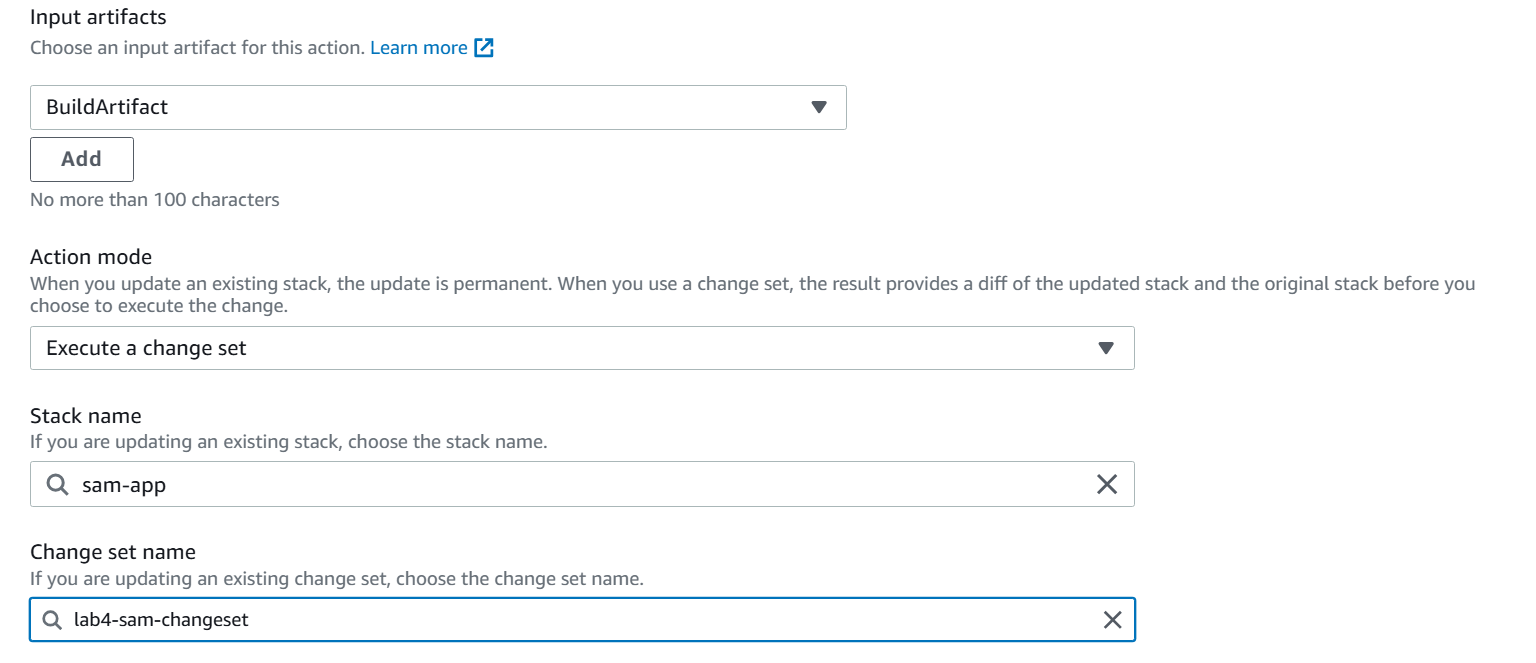
4.13 Select the edit button, and choose edit stage



4.14 Select add action group and select the following values



4.15 Enter the following details

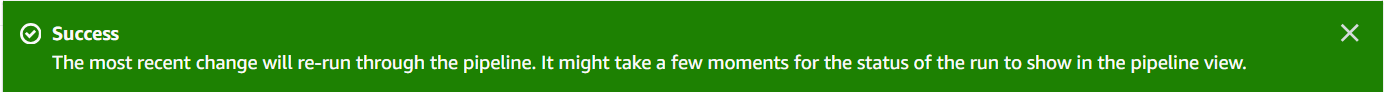


4.16 Click on Done->Done->Save->Save

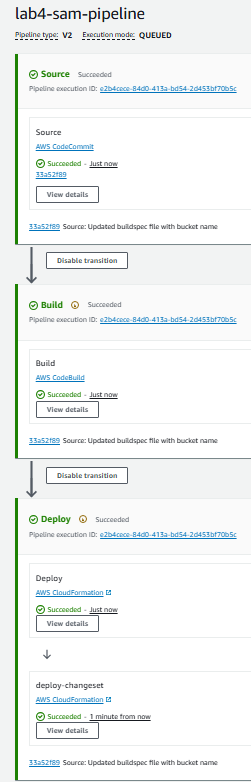


4.17 Choose the Release Change option



4.18 

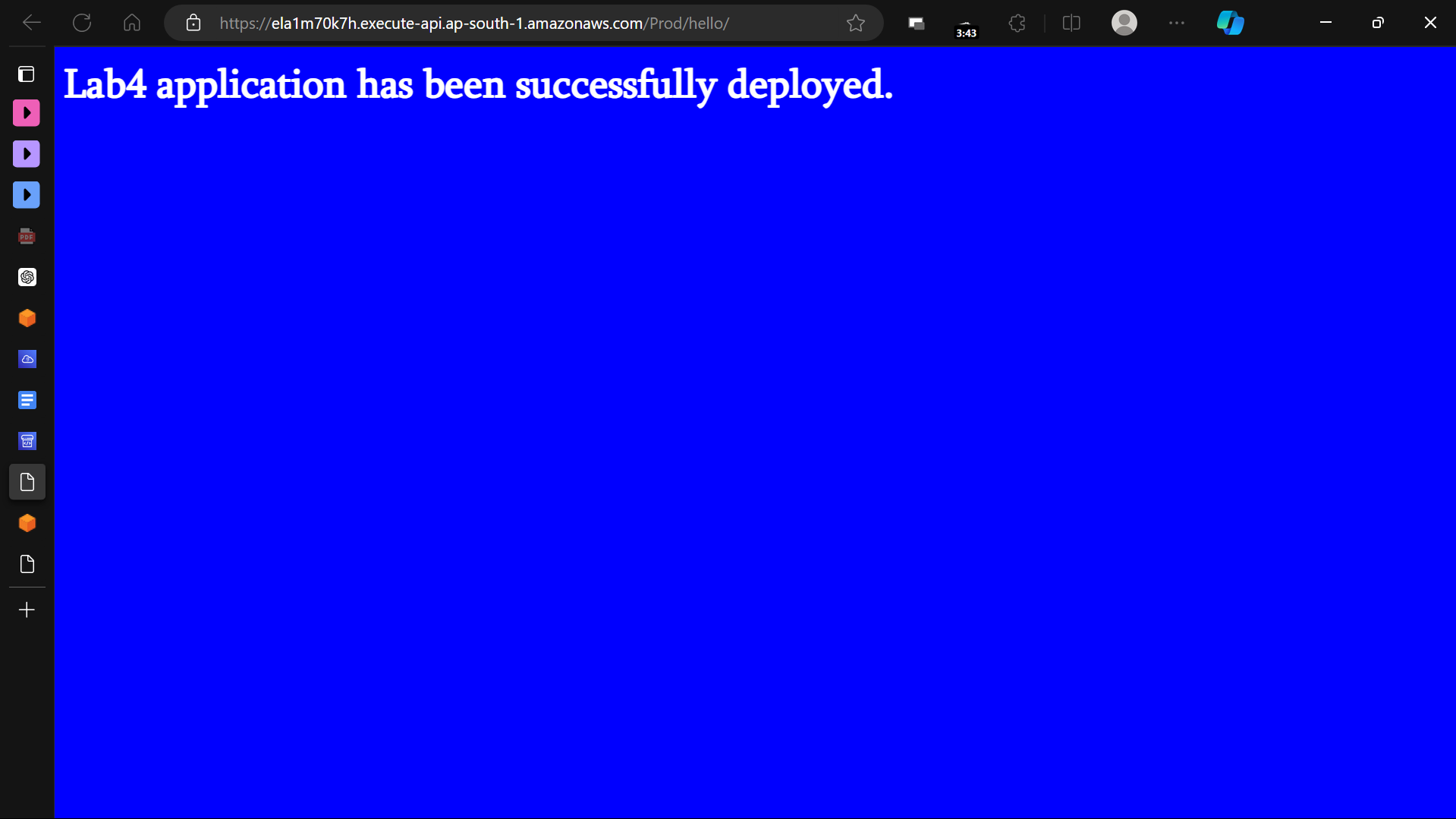
4.19 All steps of the pipeline have been completed



DOUBT - Where did we implement logic for Shifting Traffic here?

**Objective 5 - Verify the deployment, ny clicking on the link in deploy-changeset card in above step.**

Output-



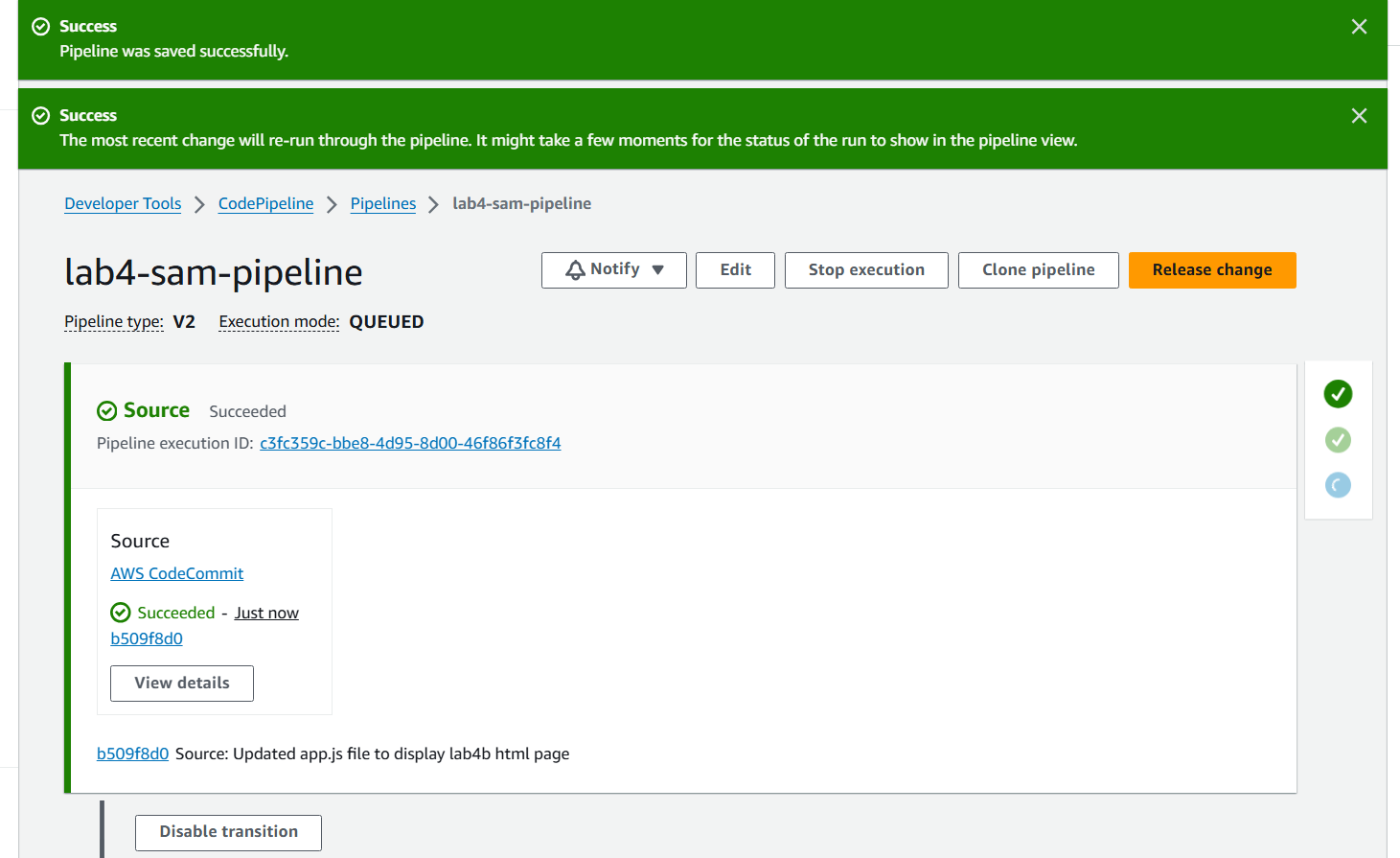
**Objective 6 - Make a new change to the application and re-deploy, to check for shifting traffic**

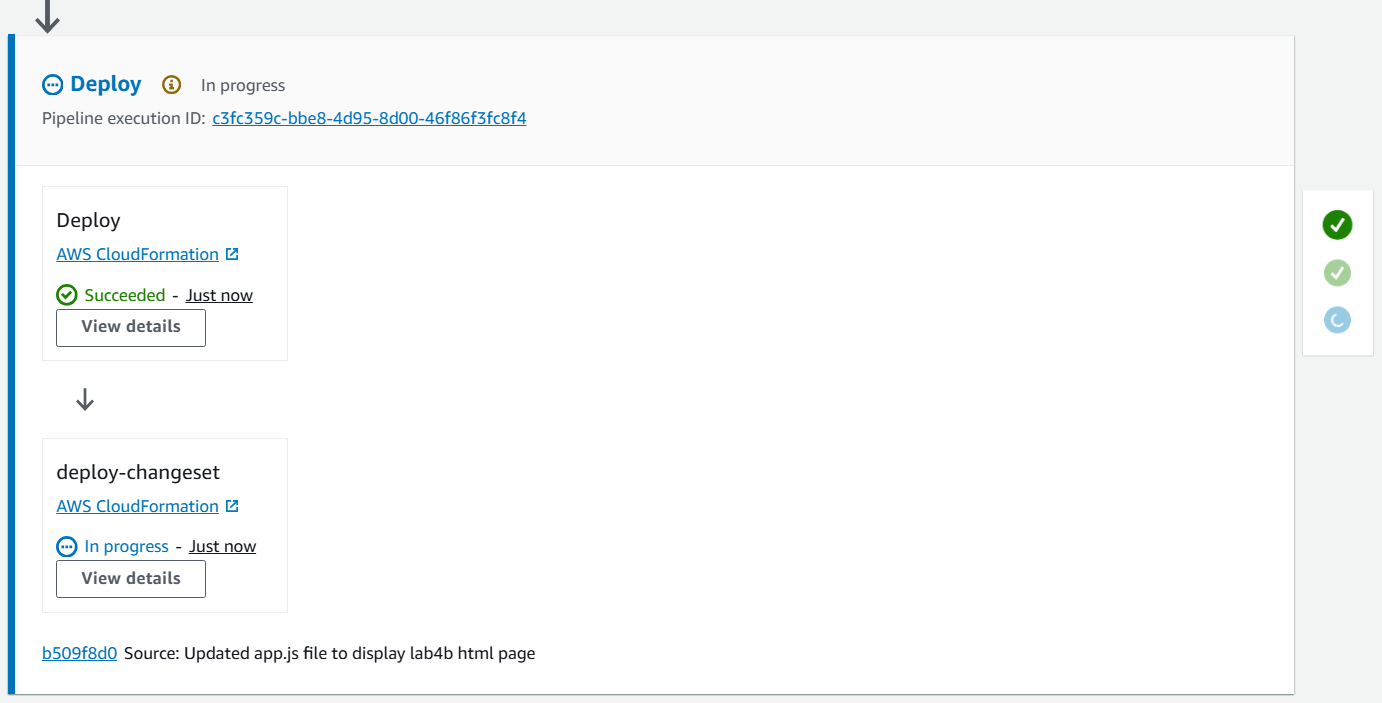
6.1 For making the changes, I’ll just do the steps given -

Which includes making some change in code for app, and pushing the change in git

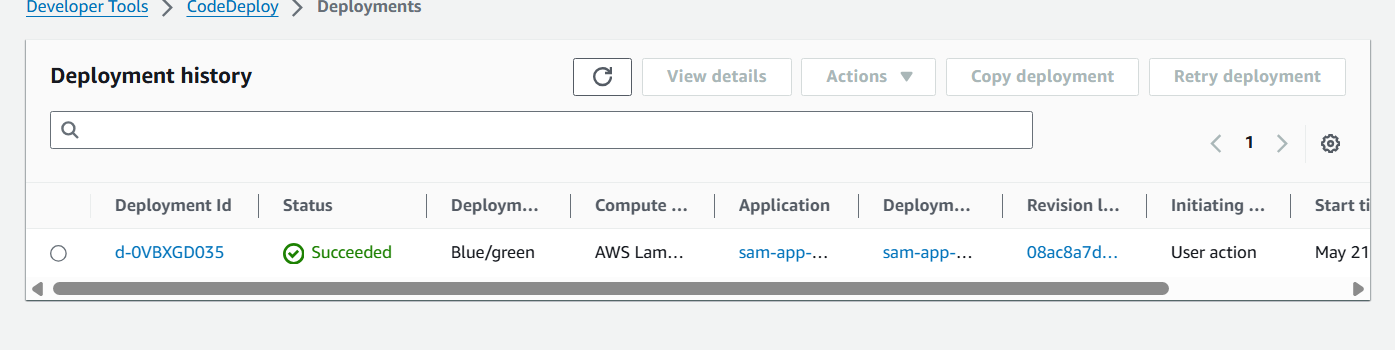
, and not show them here, as that is not our main motive, our main motive is the deploy the app with those changes, and the observe the traffic shift

6.2 Open the pipeline window, and notice that the pipeline has observed changes in the app, and automatically, it initiates a re-build of the lab4 application.





6.3 View deployment history from left pane



6.4 Clicking on the deployment ID will show the following

