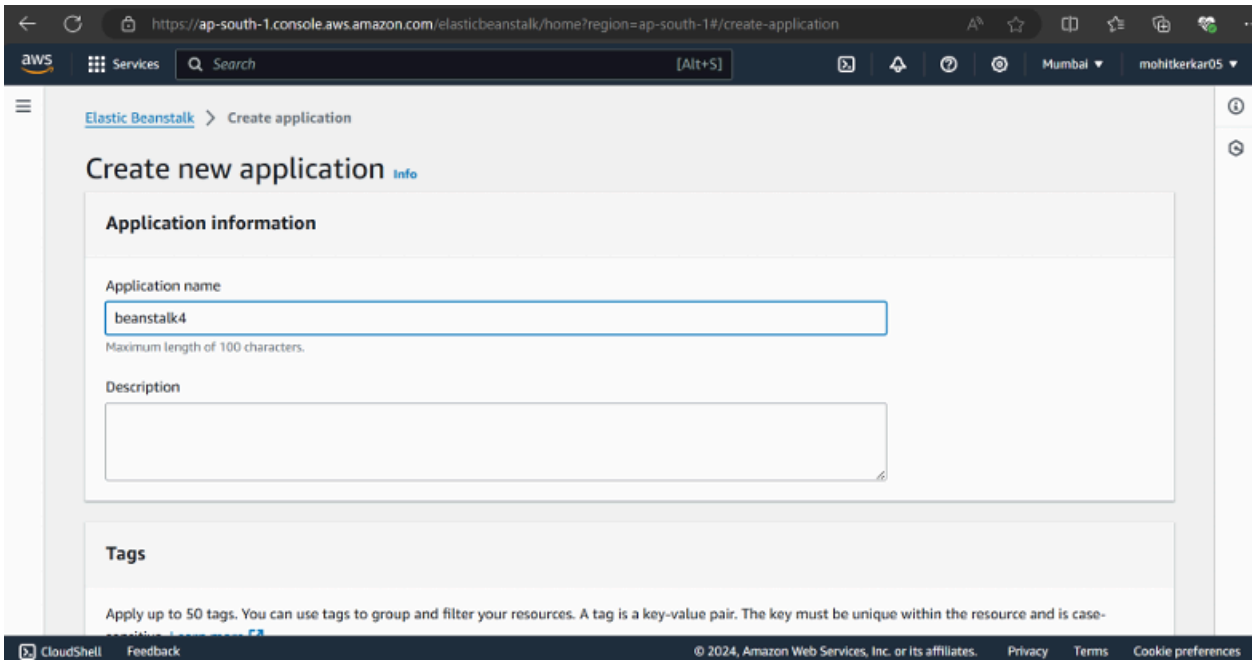
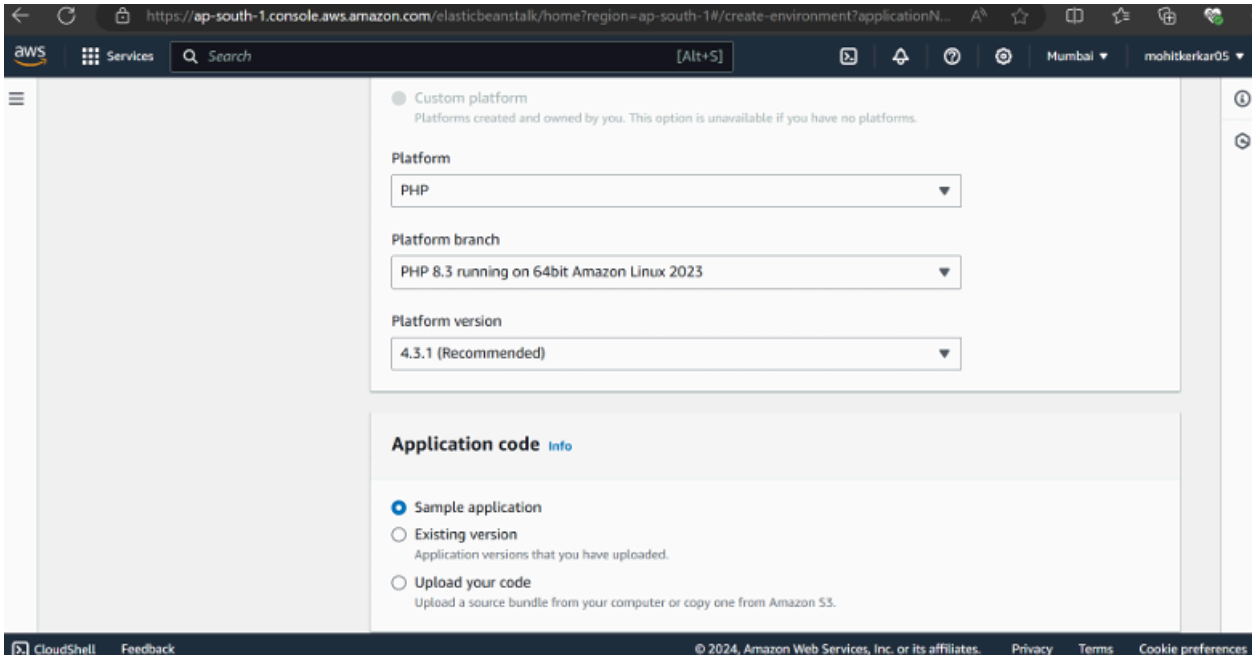
Exp 02:To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWSCodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Step 1: Create our ElasticBeanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. Rest of the configuration settings are to be kept as default.

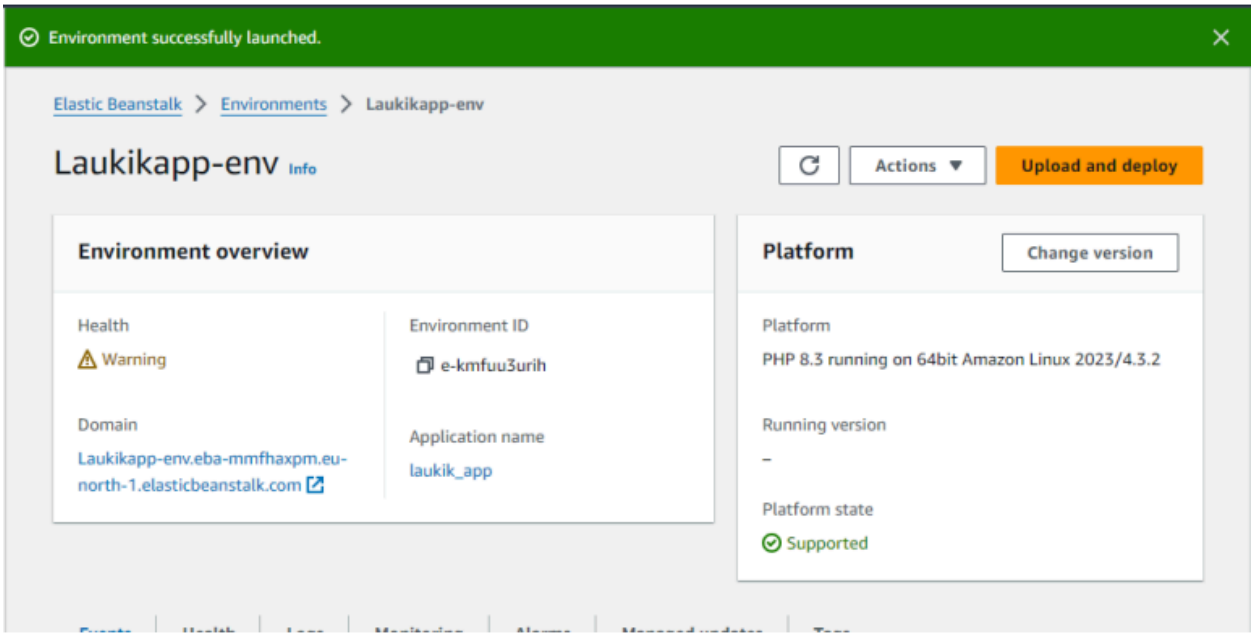


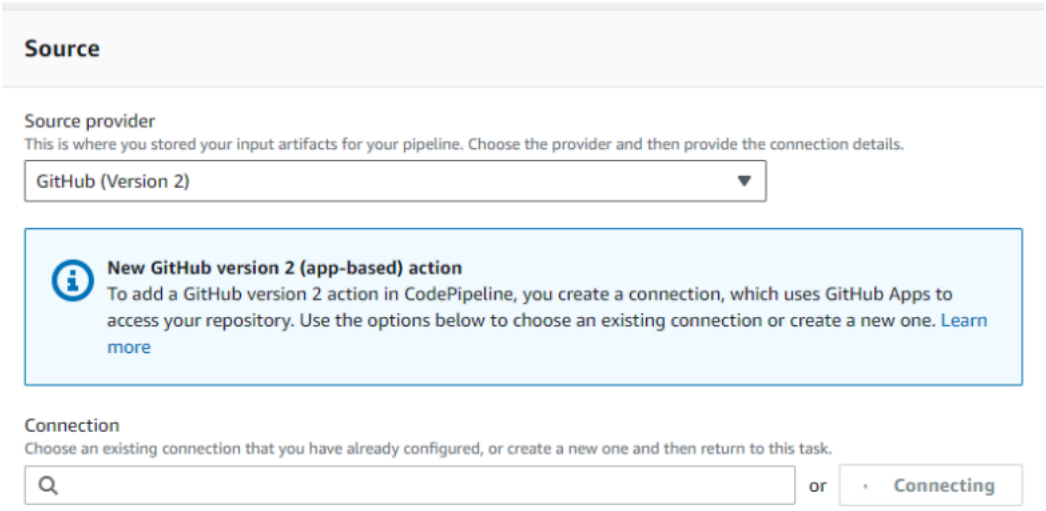


Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

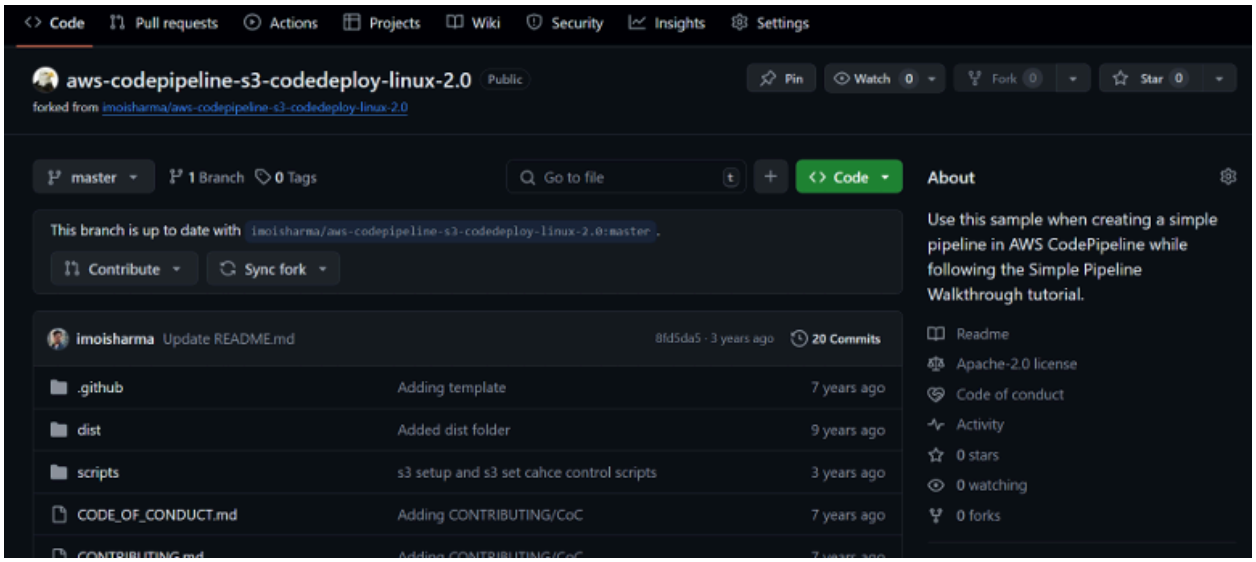
1. AWSElasticBeanStalkWebTier
2. AWSElasticBeanStalkWorkerTier
3. AWSElasticBeanStalkMulticontainerDocker

Wecan skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.

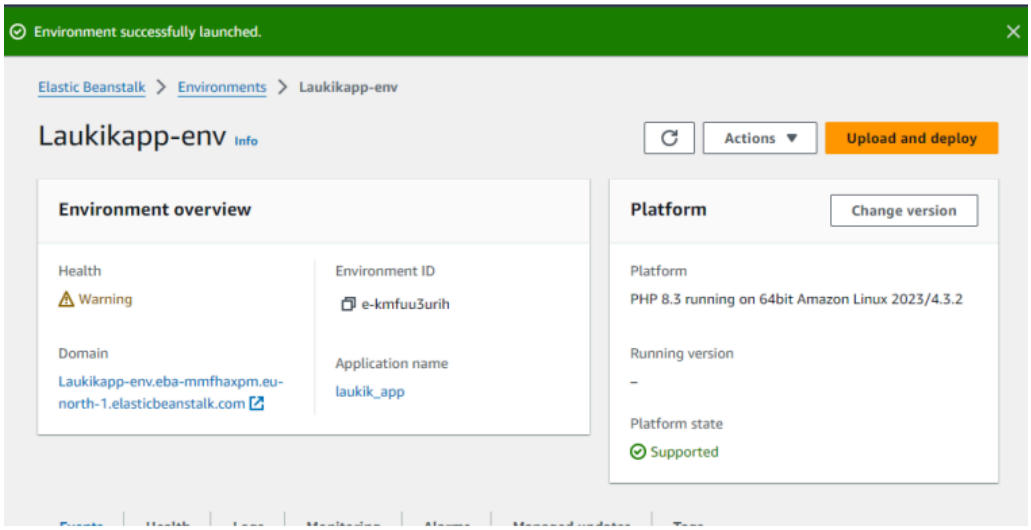




Step 2: Fork the required repository onto our github account
The repository to be forked is- imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0
This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.



Step 3: Creation of the Pipeline
Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.
And click on next …

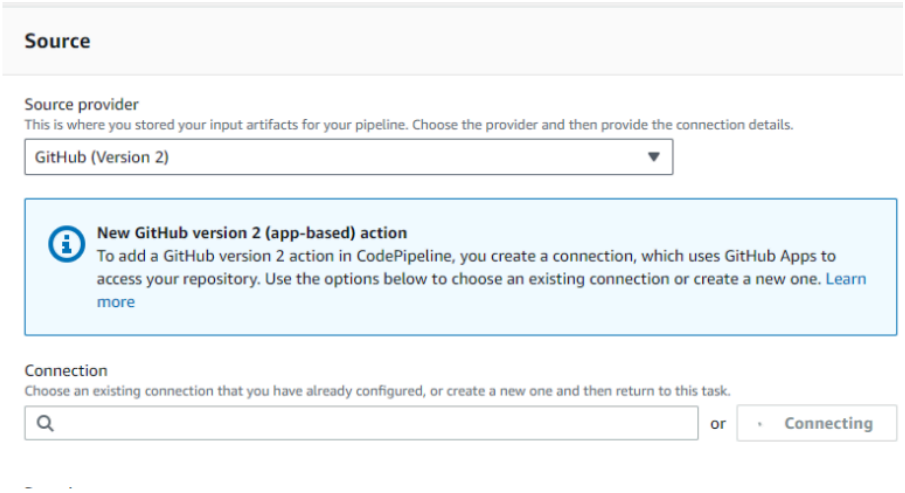


Step 4: Github connection

In this step, we are supposed to create a github connection and add our existing repository over

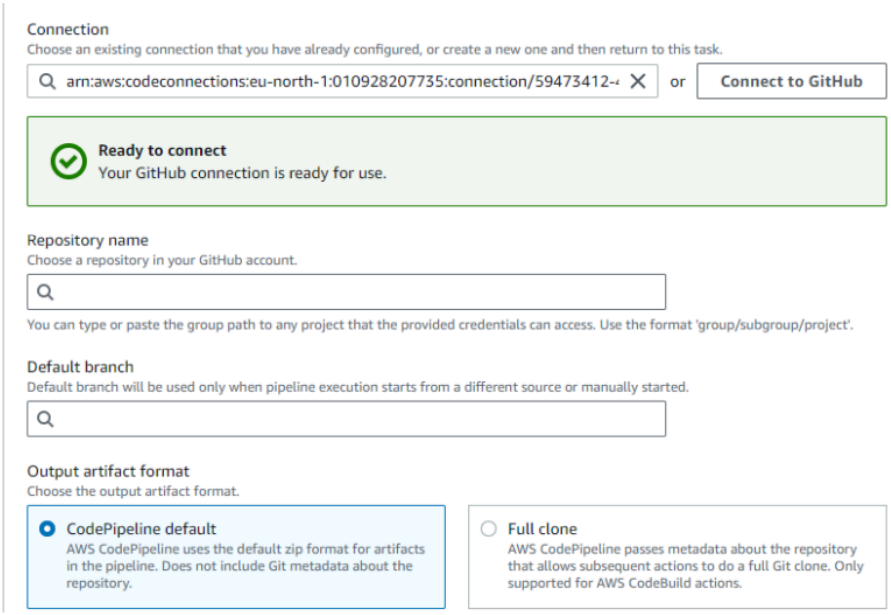
here i.e the one we forked earlier

Weare supposed to enter our github username so as to proceed towards making the connection

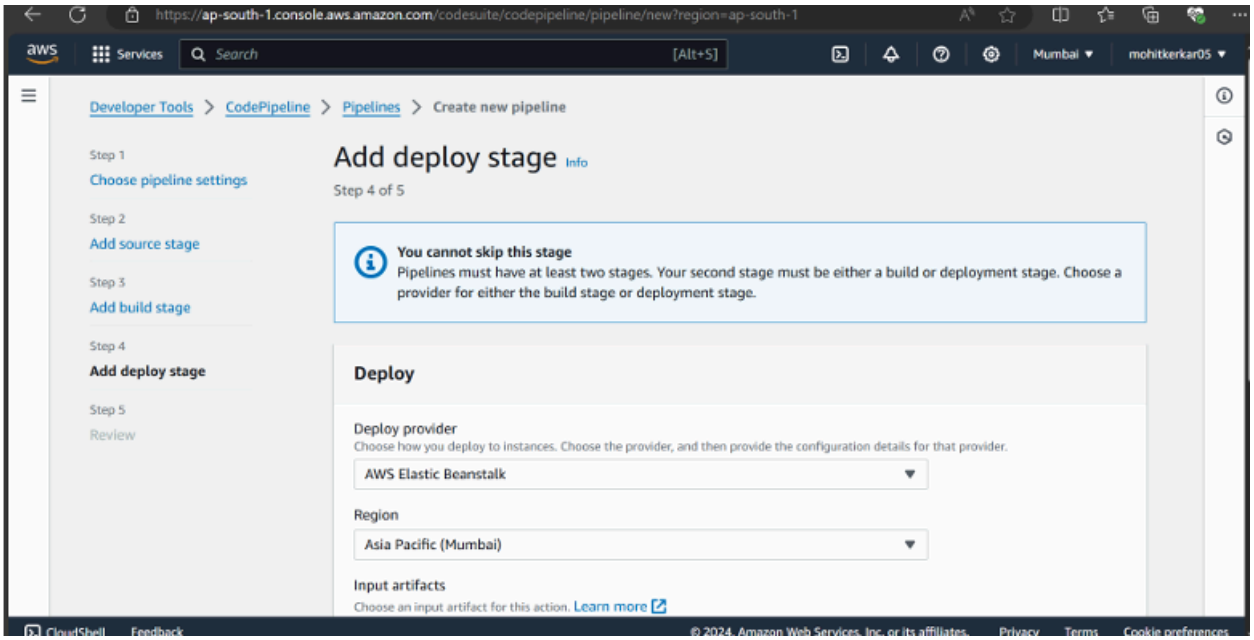


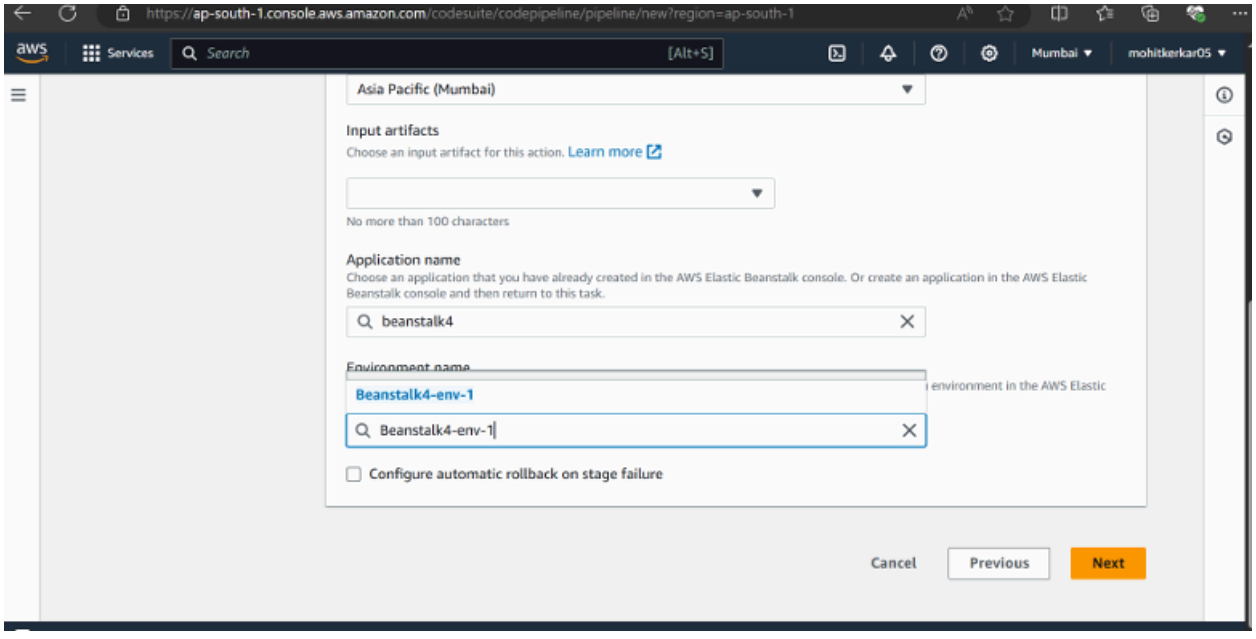
Now to finalize our connection, we are to install an application which connects AWS to our github account and repository.

Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

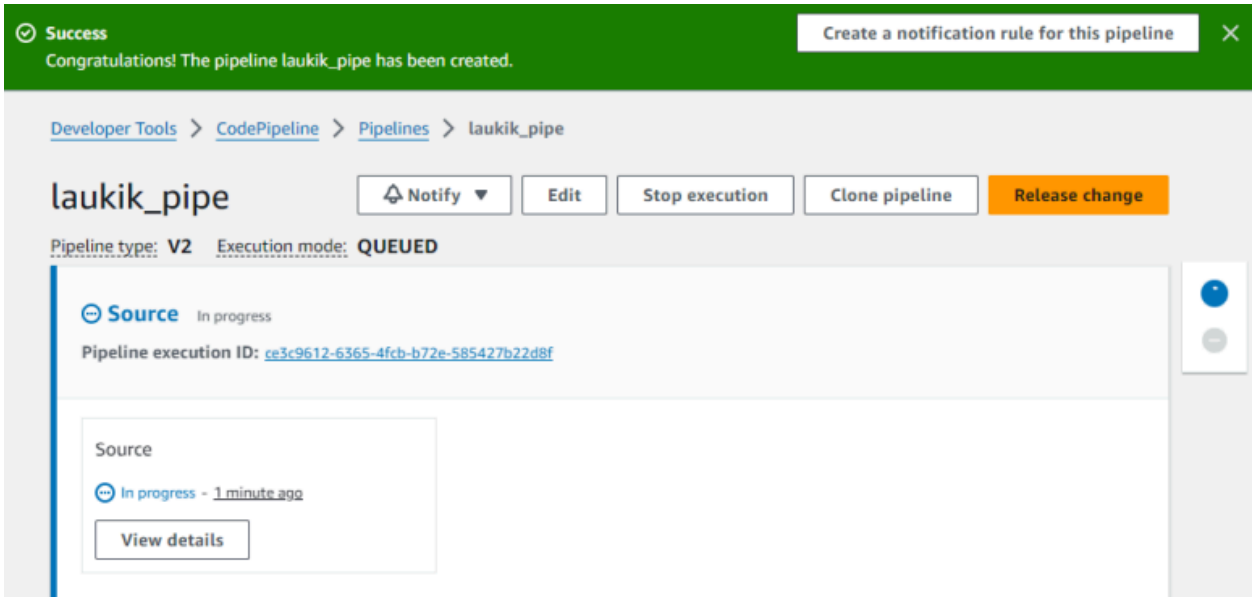


Step 5: Deployment stage:
We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation



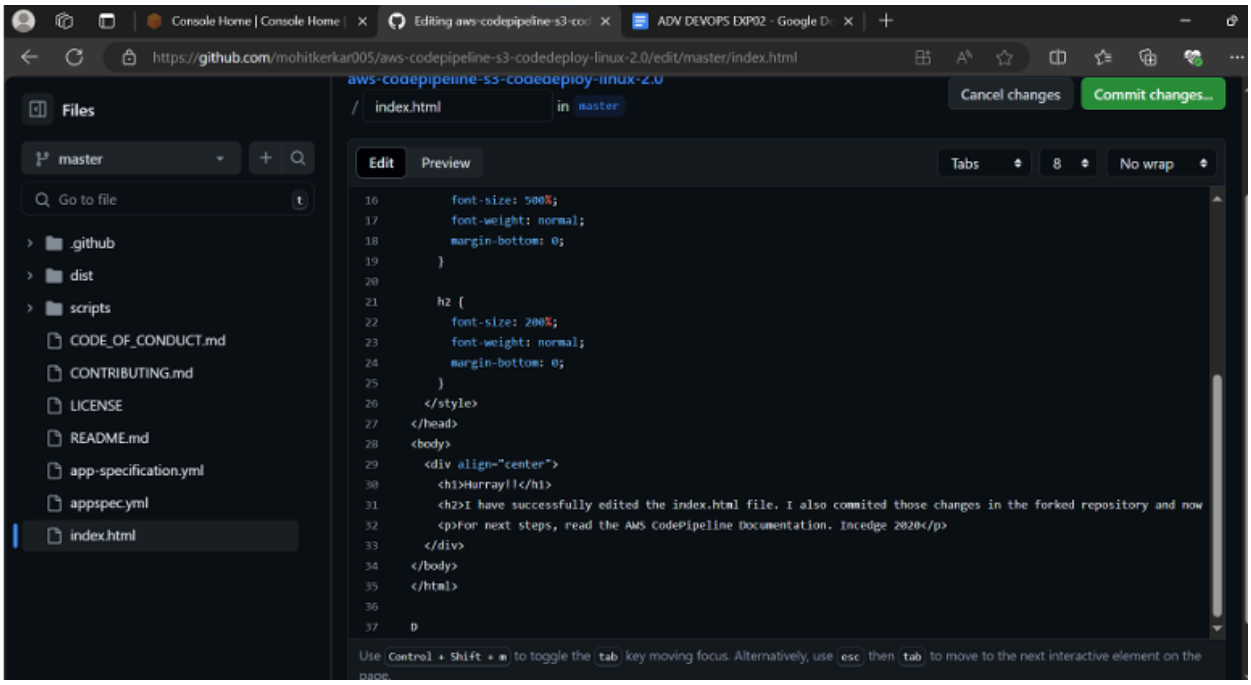


Step 6: Post deployment stage: When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created



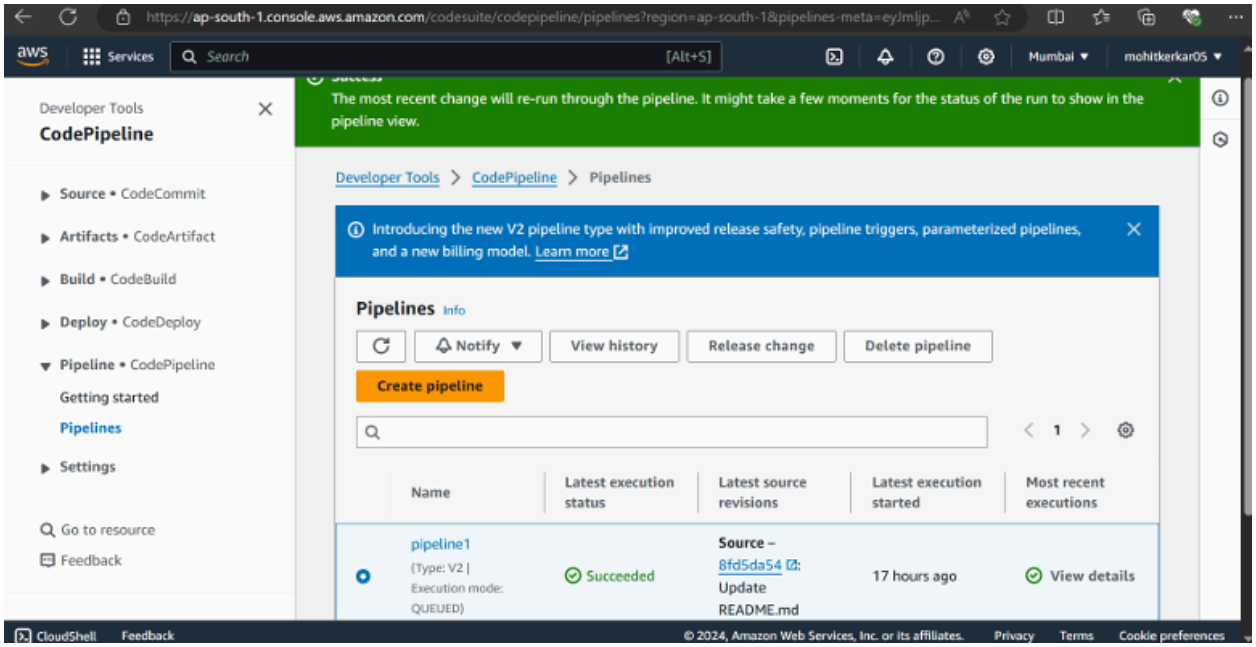
Step 7: Committing changes to your github code

Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.

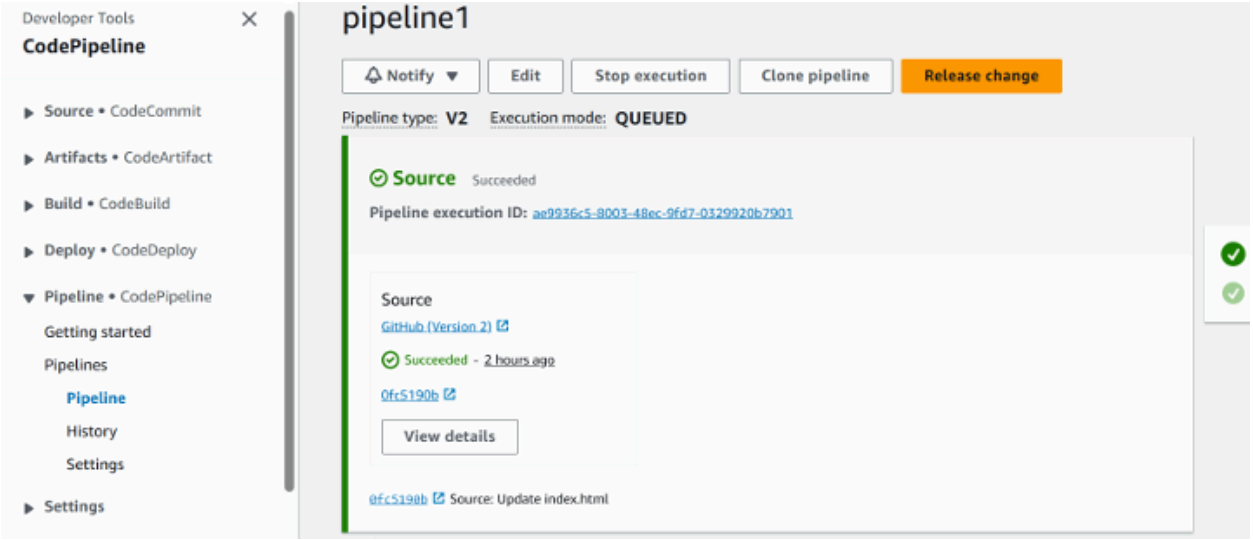


Step 8: Apply the newly made changes in index.html onto our pipeline

Come back to the Codepipeline section and select the pipeline through which we successfully created and deployed our application. Click on the release change option to apply the latest changes/commits from our github repository to our pipeline



Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. “Update index.html” was the latest commit message in the github repository



Step 9: Open the Domain of our Elastic Beanstalk environment
Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application
The text in this image is clearly distinguishable from the earlier website’s text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed