

DevOps Engineer Assessment

(CI/CD Pipeline)

Problem:

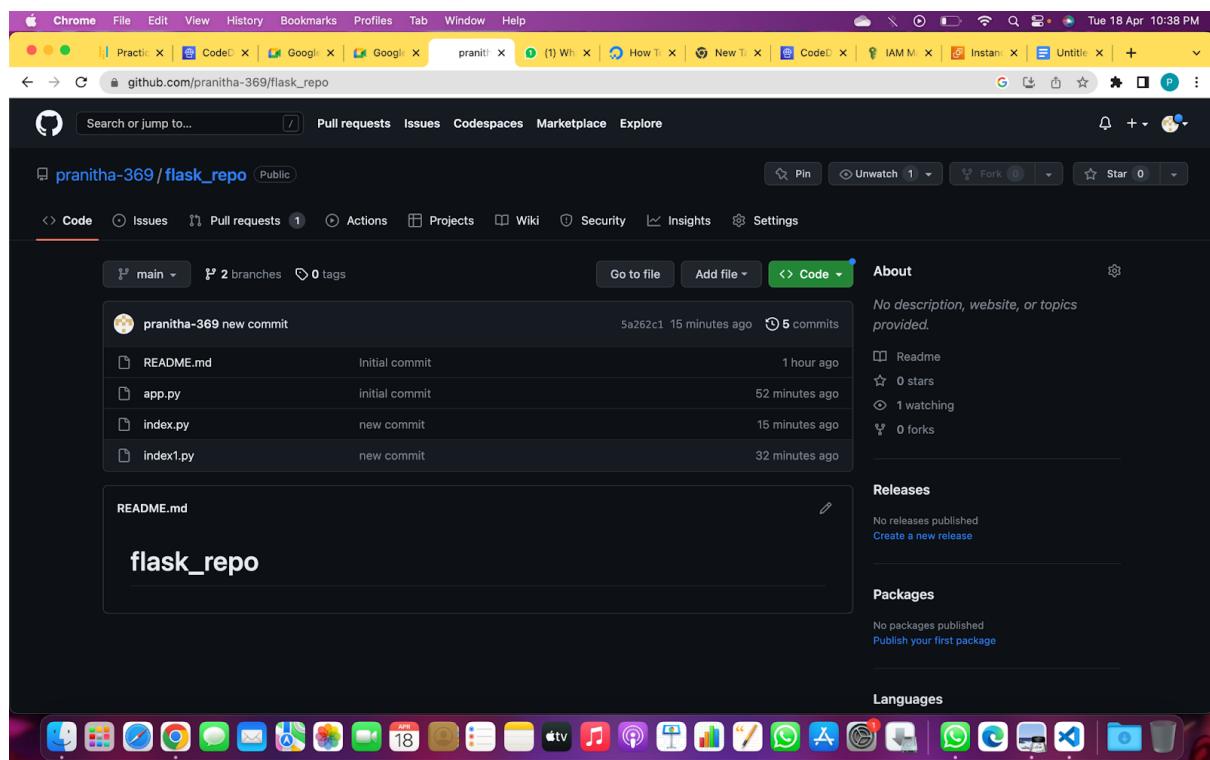
1) A server-side application built with Python Flask is on GitHub. This project is being worked on by a group of people. Anyone can commit directly to the master/main branch,

resulting in massive conflict and high error rates.

2) The same application is running on an Amazon EC2 instance. Every time a new push is received, we must manually pull the code from GitHub and redeploy it. This results in an exorbitant time waste because even a minor change requires us to go through the entire process again.

Objective:

- On GitHub, no one should be allowed to push directly to the master/main branch. A push to the master/main branch should only happen through a pull request. The request should be reviewed and approved by another collaborator.
- Once the master/main branch gets updated, It should be deployed on the EC2 instance Automatically.



Create a flask repository

In this flask repository

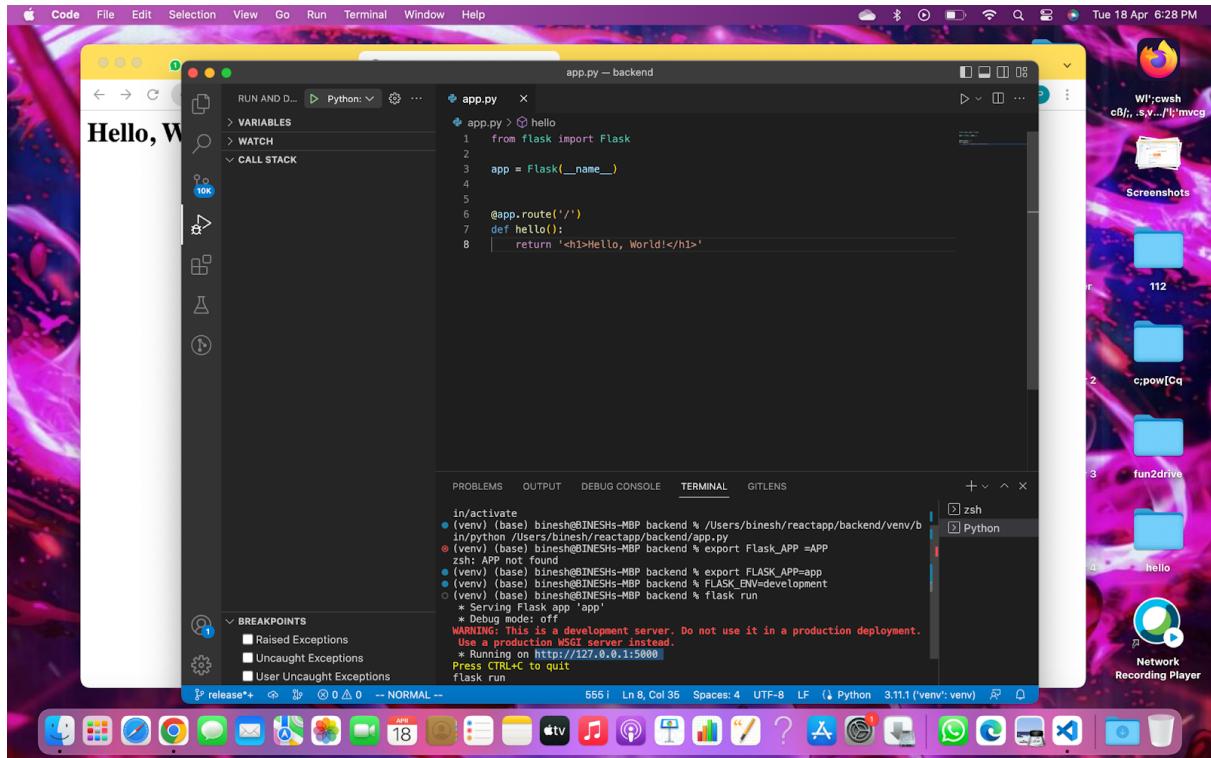
Add a new file

And commit the changes to the git repository

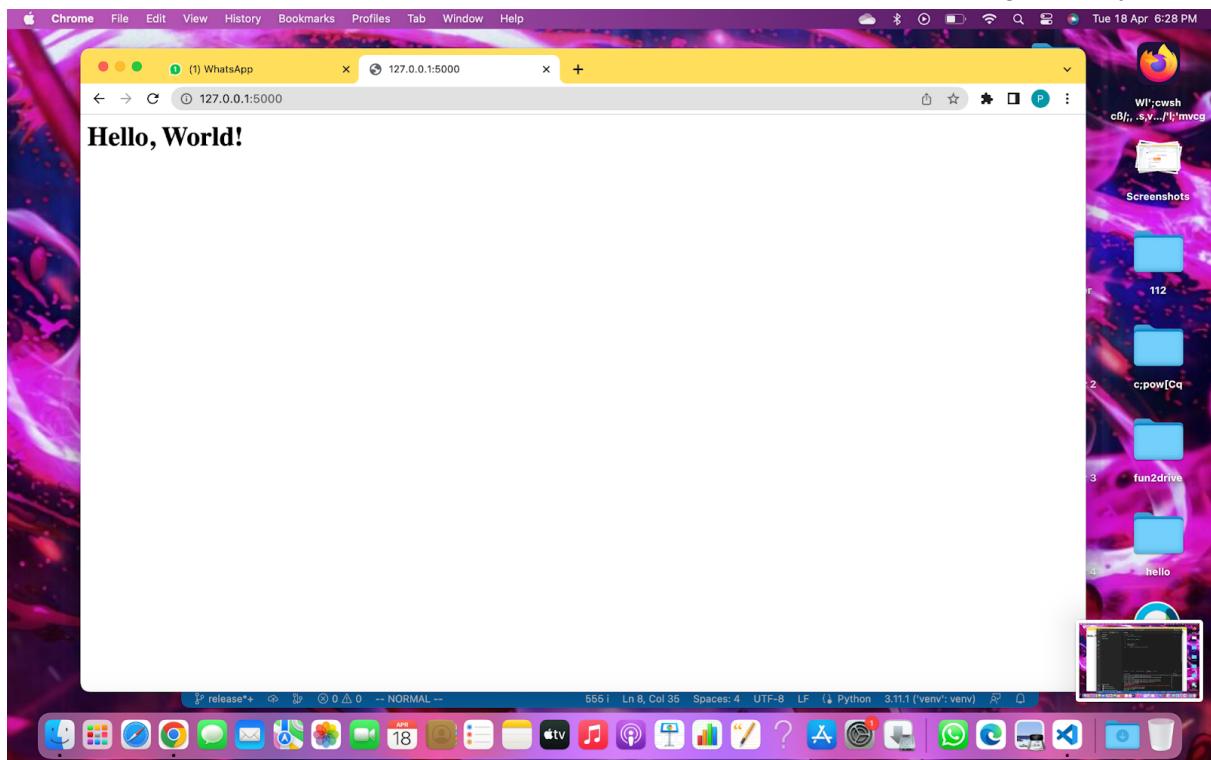
Git init

Git add file name

Git commit to save the file



Run a sample application in visual studio code and check whether it is running properly



HOW TO DISABLE DIRECT PUSH REQUEST TO MASTER AND MAIN GITHUB

First create a repository on the github

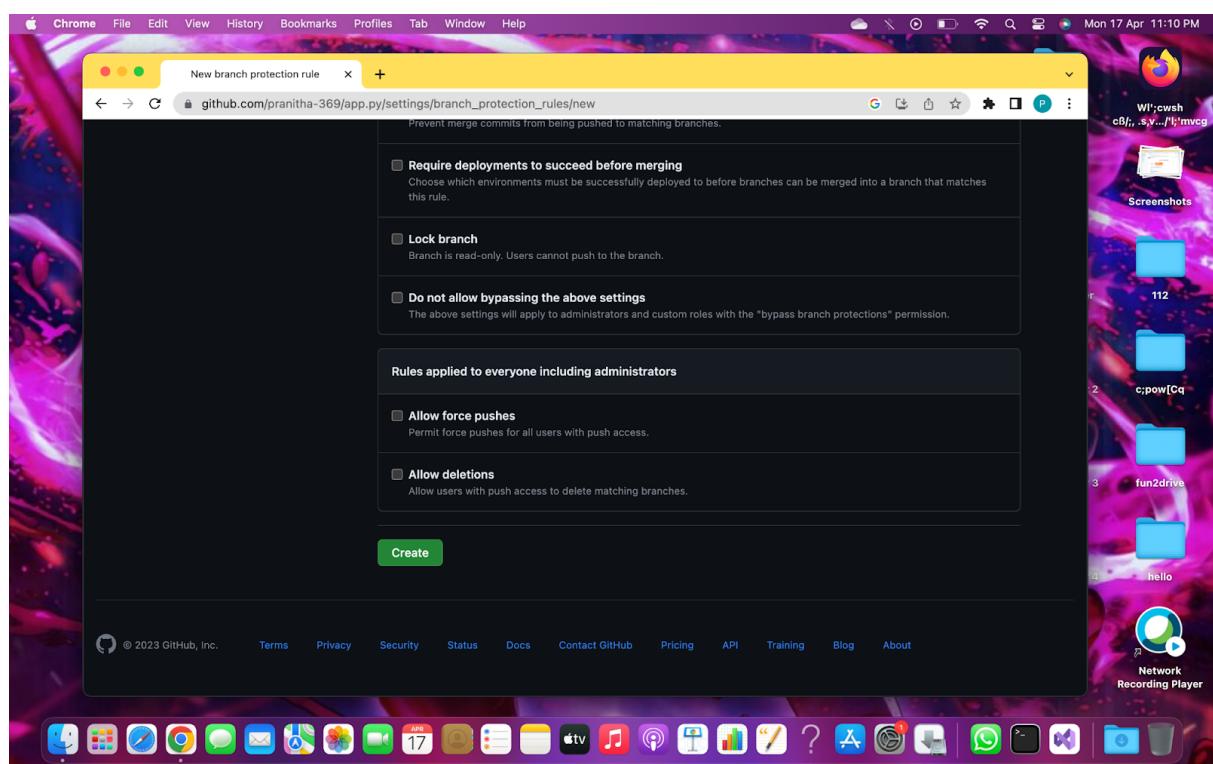
In the name of

In the setting

Go to the branches and add protection rule

it should be require a pull request before merging

On GitHub, no one should be allowed to push directly to the master/main branch. A push to the master/main branch should only happen through a pull request. The request should be reviewed and approved by another collaborator.



```
from flask import Flask
app = Flask(__name__)
@app.route('/')
@app.route('/index')
def hello():
    return <h1>Hello, World!</h1>
@app.route('/about')
def about():
    return <h3>This is a Flask web application!</h3>
```

Change some code in github and commit the code in github
Create a push request and push the code from main to dev
It will not automatically pull the code from main to master
Review is required atleast by one member or different member

new commit #1

pranitha-369 wants to merge 1 commit into `main` from `dev`

Conversation 0 Commits 1 Checks 0 Files changed 1

No description provided.

pranitha-369 commented 27 minutes ago

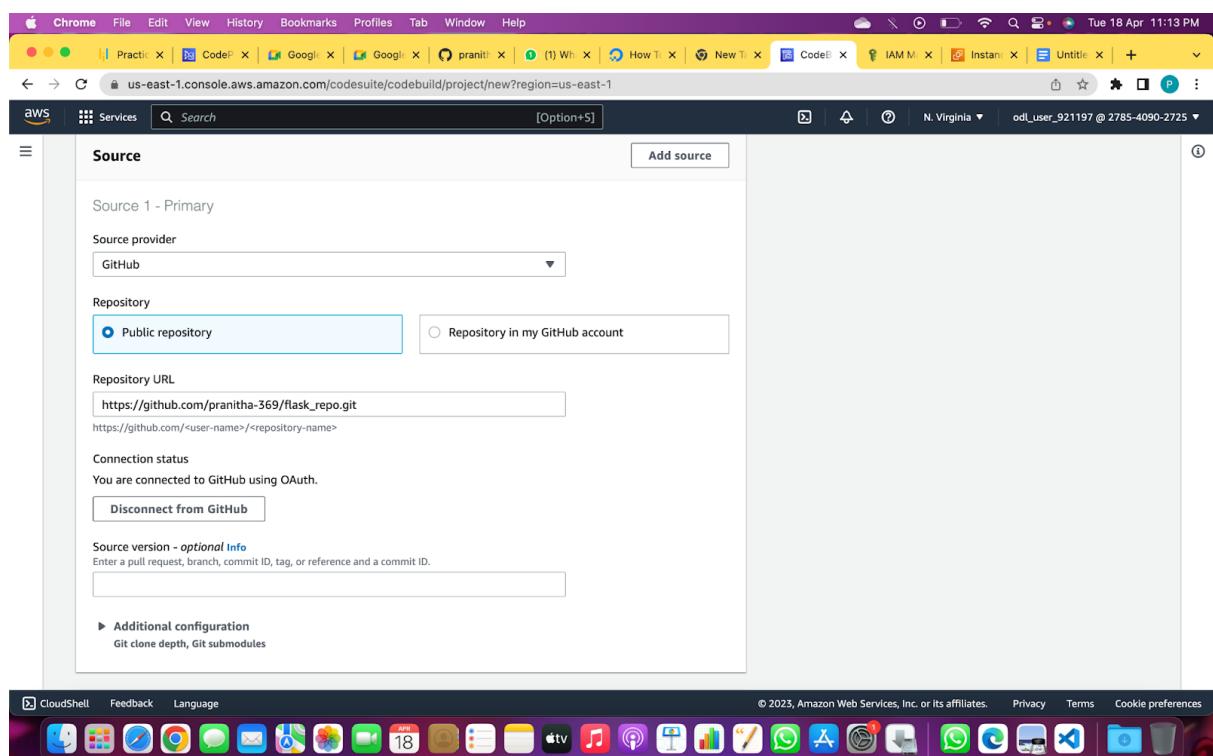
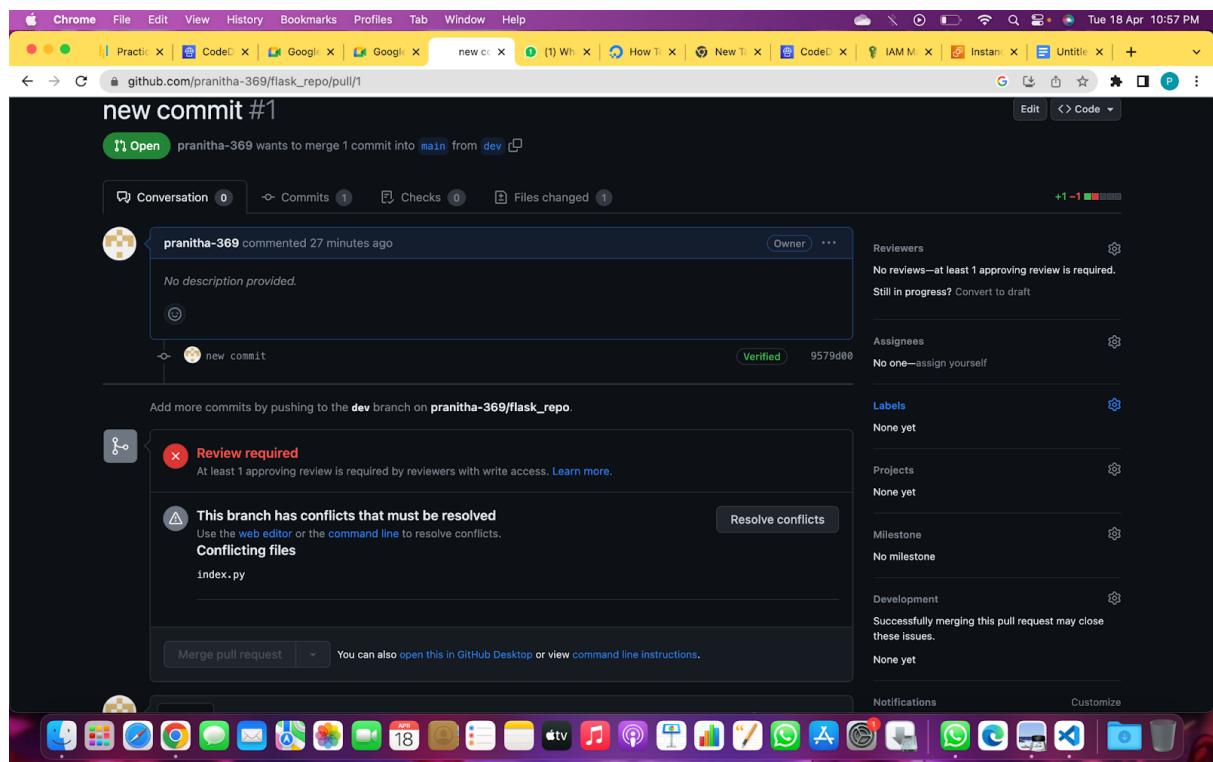
new commit

Verified 9579d80

Review required At least 1 approving review is required by reviewers with write access. Learn more.

This branch has conflicts that must be resolved Use the web editor or the command line to resolve conflicts. Resolve conflicts

In the cicd pipeline
In the build stage



Go to the cicd pipeline in the aws account

The image consists of three vertically stacked screenshots of the AWS CodeBuild project creation interface in a web browser.

Screenshot 1: Environment Configuration

This screen shows the configuration for the build environment:

- Environment image:** Selected "Managed image" (Use an image managed by AWS CodeBuild).
- Operating system:** Set to "Amazon Linux 2".
- Runtime(s):** Set to "Standard".
- Image:** Set to "aws/codebuild/amazonlinux2-x86_64-standard:3.0".
- Image version:** Set to "Always use the latest image for this runtime version".
- Environment type:** Set to "Linux".

A note indicates: "The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See Docker Images Provided by CodeBuild for details.".

Screenshot 2: Artifacts and Logs Configuration

This screen shows the configuration for artifacts and logs:

- No artifacts:** A note says "You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR."
- Additional configuration:** Options for "Cache" and "encryption key".
- Logs:**
 - CloudWatch:** "CloudWatch logs - optional" is checked. A note says "Checking this option will upload build output logs to CloudWatch.".
 - Group name:** An input field.
 - Stream name:** An input field.
- S3:** "S3 logs - optional" is unchecked. A note says "Checking this option will upload build output logs to S3.".

Screenshot 3: Final Step

This screen shows the final step of creating the build project:

- Create build project** button (orange).

Create a ec2 instance

EC2 Instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
flask_ec2	i-043cc203b643ccf41	Running	t2.micro	-	No alarms	us-east-1d	ec2-199-214-111-123
Flaskapp-env	i-0dd75e1ab341e1ee6	Running	t3.micro	2/2 checks passed	No alarms	us-east-1b	ec2-199-214-111-124

Create deployment group

Create a deployment group

The screenshot displays a Mac desktop with three windows open:

- AWS CodeDeploy - Create deployment group**: This window shows the configuration for creating a deployment group. It includes sections for "Application" (set to "flask_application"), "Deployment group name" (set to "deploy"), and "Service role".
- AWS CodeDeploy - Deployment Targets**: This window lists deployment targets. Under "Amazon EC2 instances", there is one entry: "1 unique matched instance" with the tag "flask_ec2".
- CloudShell - sai (8).pem**: A terminal window showing a standard macOS dock at the bottom.

Go to the code pipeline
 Choose code pipeline Name
 In the artifact setting choose default location
 Encryption is managed by aws
 Create
 In the source
 Choose source provider: github (version 1 or version 2)
 Choose a connection with github

Connect to github

Choose the github repo name

Branch name

Choose the code build if created

Choose the already created code deploy

The screenshot shows the AWS CodePipeline console with a successful pipeline run for the 'flaskapp' pipeline. The pipeline consists of two stages: 'Source' and 'Deploy'. The 'Source' stage is configured to pull from a GitHub repository and has succeeded. The 'Deploy' stage is configured to use AWS Elastic Beanstalk and has also succeeded. The pipeline execution ID is c25b2680-608d-4215-a2cc-0acca00ecb99. The status of each stage is indicated by a green circle with a checkmark. The pipeline is currently in a 'Succeeded' state.

This screenshot is identical to the one above, showing the AWS CodePipeline console with a successful pipeline run for the 'flaskapp' pipeline. The pipeline stages, execution ID, and overall status are all the same as in the first screenshot.