Lab 7

Flask App CI/CD Pipeline with AWS CodePipeline

Objective:

The objective of this lab is to set up a continuous integration and deployment (CI/CD) pipeline for a Flask app using AWS CodePipeline. This will automate the process of deploying changes to the Flask app in a reliable and consistent manner.

Prerequisites:

- Knowledge of AWS IAM, S3, EC2
- Knowledge of Flask application
- AWS CLI configured

Steps

1. Create Flask Application

1. app.py

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)

```

2. index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to Flask App </title>
    <style>
        body {
            display: flex;
            align-items: center;
            justify-content: center;
            height: 80vh;
            margin: 0;
            background-color: #a0a0a0; /* Grey Background */
        }

        h1 {
            color: #045677; /* Dark Grey Text Color */
        }
    </style>
</head>
<body>
    <h1>Welcome to Flask App via CodePipeline </h1>
</body>
</html>
```

2. AWS Code Commit

1. Create AWS CodeCommit repository
2. Create IAM user with AWSCodeCommitPowerUser Policy
3. In Security credentials of IAM User, Click on "Generate HTTPS Git credentials for AWS CodeCommit" and save them
4. Copy HTTPS link from CodeCommit Repository created
5. On Local machine, clone the repository using the HTTPS link and use the generated Git Credentials
6. Copy the flask application to the CodeCommit repository on local machine
7. Push it to AWS CodeCommit using git
    1. git status
    2. git add .
    3. git commit -m "Added flask app"
    4. git push origin master
    5. Use The Generated Git Credentials to Authenticate
8. Check AWS CodeCommit repository

3. AWS Code Build

1. Create Build Project
2. Select source as AWS CodeCommit
3. Select Managed Image for Environment Image
4. Select EC2 for Compute
5. Select Ubuntu for Operating System
6. Select New-service-role and it will create it automatically

7. On Local machine create a buildspec.yml file and push it to CodeCommit Repo

```yaml
version: 0.2

phases:
  build:
    commands:
      - mkdir /flask-app
      - cp -r * /flask-app
artifacts:
  files: '**/*'

```

8. On Local machine create an appspec.yml file and push it to CodeCommit Repo

flask-lab-test > ! appspec.yml

```yaml
version: 0.0
os: linux
files:
  - source: /
    destination: /home/ubuntu/flask-app
hooks:
  AfterInstall:
    - location: scripts/install.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start.sh
      timeout: 300
      runas: root

```

9. On Local machine create file scripts/install.sh and push it to CodeCommit Repo

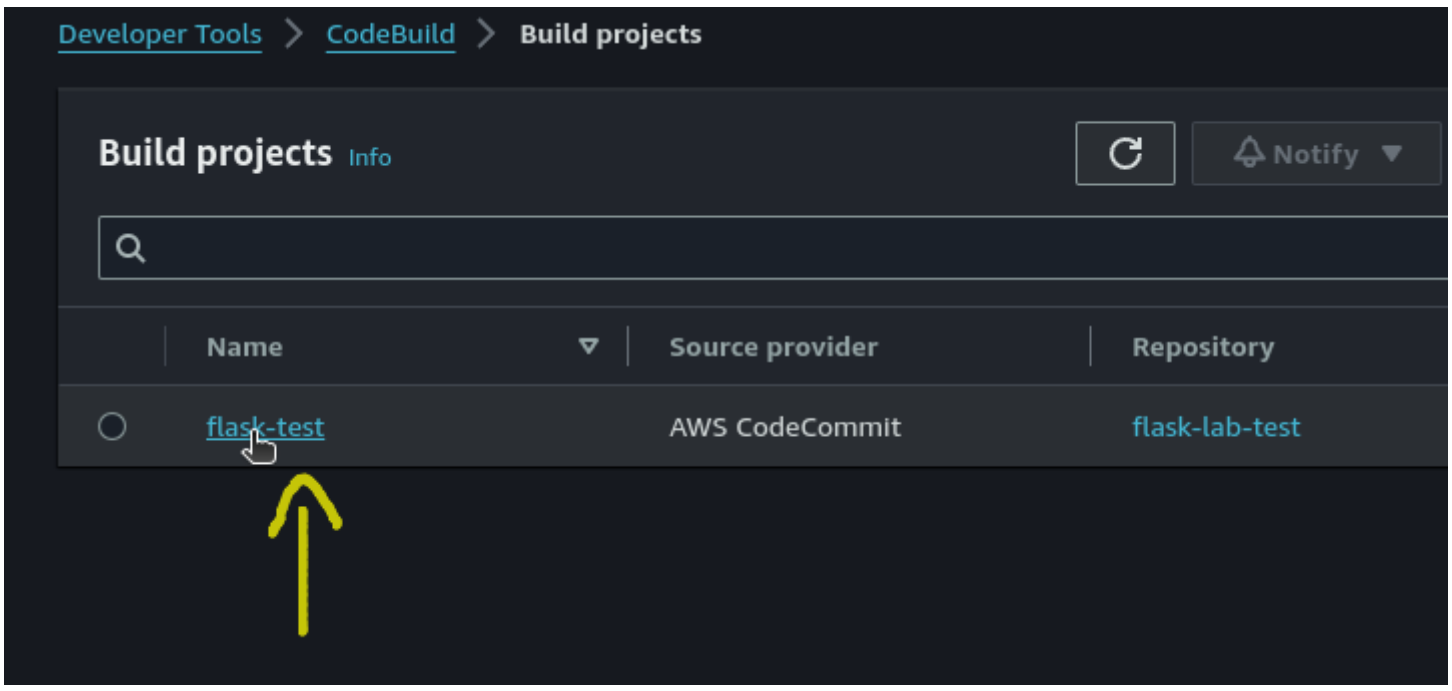
```
#!/bin/bash

# Update the package lists
sudo apt update

# Install prerequisites
sudo apt install -y software-properties-common

# Add the deadsnakes PPA to get Python 3.9
sudo add-apt-repository --yes ppa:deadsnakes/ppa

# Update the package lists again
sudo apt update

# Install Python 3.9
sudo apt install -y python3.9

# Verify the installation
python3.9 --version

# Install pip
sudo apt update
sudo apt install python3.9-distutils -y
sudo wget https://bootstrap.pypa.io/get-pip.py
sudo python3.9 get-pip.py
pip3.9 --version

# Install Flask while ignoring installed blinker
pip3.9 install --ignore-installed Flask
```

10. On Local machine create file scripts/start.sh and push it to CodeCommit Repo

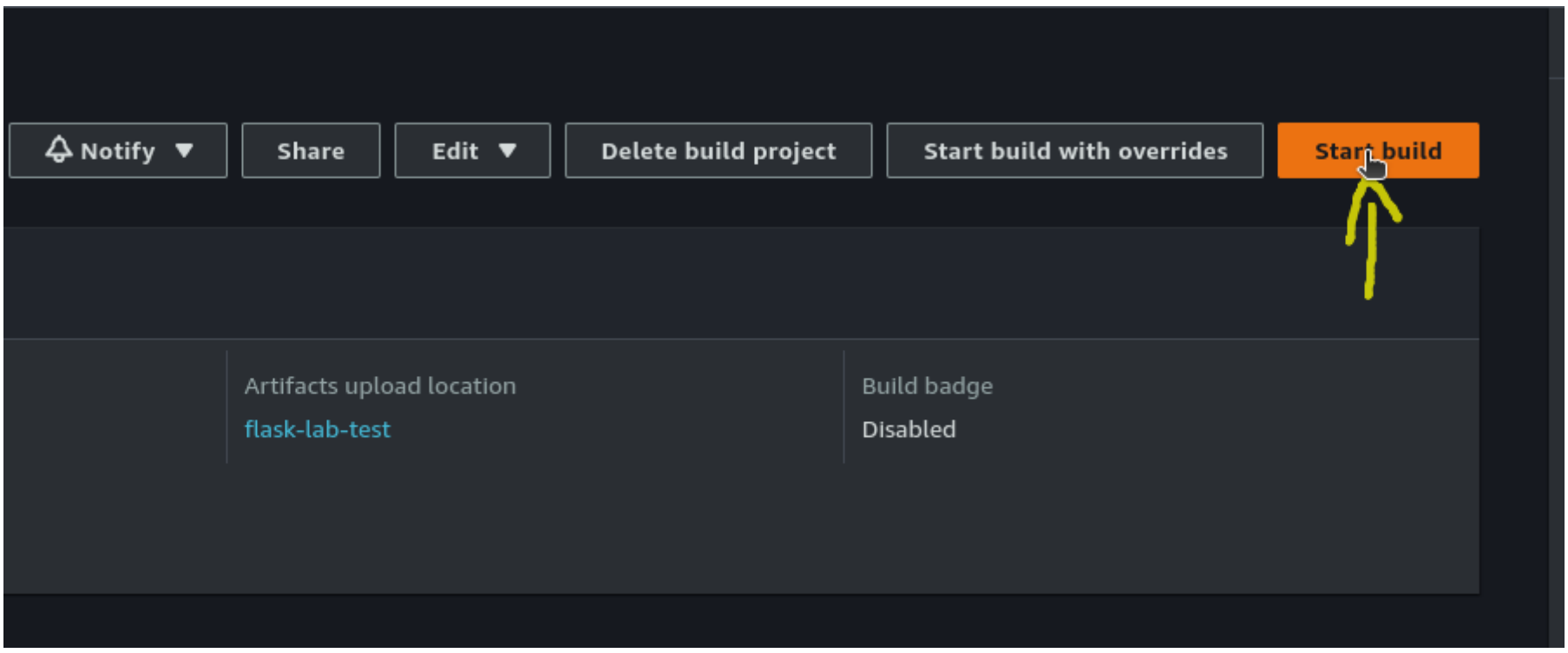
```
#!/bin/bash
# Run Flask app in the background
nohup python3.9 /home/ubuntu/flask-app/app.py > flask.out 2>&1 &
```

11. In Build Project leave the buildspec name as it will automatically select it from Repo
12. Select Artifacts as S3
13. Create bucket in S3
14. Add the bucket name in AWS Code Build Artifacts section
15. Enter a name of zip file e.g artifacts.zip
16. Select Zip in Artifacts packaging
17. Disable Artifact encryption
18. Un-check logs
19. Create build project

20. Click on the project



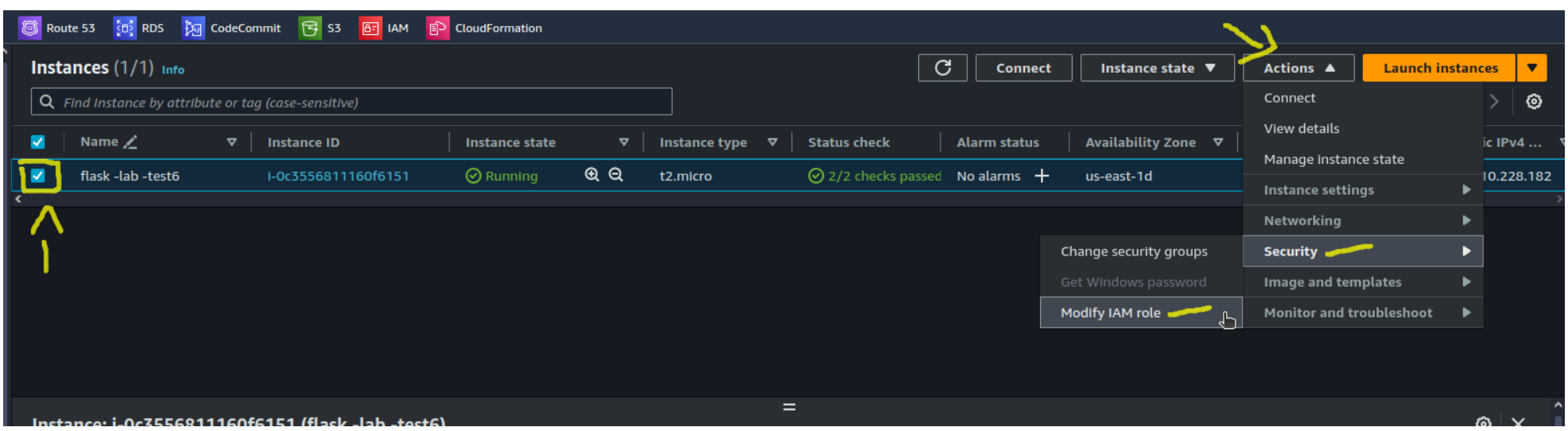
21. Click on Start Build



22. After build is successful
23. Check The S3 bucket for artifact.zip file

3. AWS Code Deploy

1. Create AWS EC2 Instance for Code Deploy
2. Create a role in IAM for EC2 with following Policies
    1. EC2Fullaccess
    2. S3FullAccess
    3. AWSCodeDeployFullAccess
3. To Attach it to the EC2 Instance:
    1. Select EC2 Instance
    2. Click On Actions
    3. Click on Security
    4. Click on Modify IAM role



4. In IAM, Create a role for AWS CodeDeploy with the following policies

    1. AmazonEC2FullAccess
    2. AmazonS3FullAccess
    3. AWSCodeDeployFullAccess
    4. AWSCodeDeployRole
    5. AmazonEC2RoleforAWSCodeDeploy
    6. AmazonEC2RoleforAWSCodeDeployLimited
5. Go to AWS CodeDeploy
6. Click On Application > Create Application
7. Enter name and select compute platform as EC2/On-premises
8. Go in the application created
9. Create deployment Group
    1. Enter Name
    2. Enter the service role name create for CodeDeploy
    3. Select In-place for Deployment type
    4. Use Amazon EC2 Instances for Environment Configuration
    5. Put tags of the EC2 instance created for code deploy
    6. Select Never for Agent configuration with AWS Systems Manager
    7. Uncheck Enable Load balancing
    8. leave rest as default and create
10. Now we will setup agent on EC2 so our CodeDeploy can use it
    1. Login to ec2 via ssh
    2. Create a .sh file i.g agent.sh
    3. Copy the following in the agent.sh file
        1. sudo apt-get update
        2. sudo apt-get install ruby-full ruby-webrick wget -y
        3. cd /tmp
        4. wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/releases/codedeploy-agent_1.3.2-1902_all.deb
        5. mkdir codedeploy-agent_1.3.2-1902_ubuntu22
        6. dpkg-deb -R codedeploy-agent_1.3.2-1902_all.deb codedeploy-agent_1.3.2-1902_ubuntu22
        7. sed 's/Depends:.*/Depends:ruby3.0/' -i ./codedeploy-agent_1.3.2-1902_ubuntu22/DEBIAN/control
        8. dpkg-deb -b codedeploy-agent_1.3.2-1902_ubuntu22/
        9. sudo dpkg -i codedeploy-agent_1.3.2-1902_ubuntu22.deb
        10. systemctl list-units --type=service | grep codedeploy
        11. sudo service codedeploy-agent status
    4. run command: bash agent.sh
    5. This will install the agent automatically
11. Go in the deployment group created and click on Create Deployment
    1. Enter name of Deployment Group
    2. Select S3 for Revision type
    3. Copy the S3 URL of the artifact.zip file and paste it in Revision location
    4. Leave rest as default and Create
12. The deployment will automatically run after a few seconds and will use the appspec.yml file to run.
13. After deployment is successfull check the flask application in browser using instance-public-ip:5000
14. To stop the flask application:
    1. Log in to instance
    2. sudo apt install net-tools
    3. sudo netstat -tulpn | grep :5000
    4. sudo kill pid-of-process

```
ubuntu@ip-172-31-18-20:~$
ubuntu@ip-172-31-18-20:~$ sudo netstat -tulpn | grep :5000
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN      6803/python3.9
ubuntu@ip-172-31-18-20:~$
```

4. AWS Code Pipeline

1. Go to AWS CodePipeline
2. Click on Create Pipeline
    1. Enter Name for pipeline
    2. Leave rest as default and click on Next
    3. Choose AWS CodeCommit for Source Provider
    4. Enter the Repo name and branch name
    5. select AWS CodePipeline for Check detection options
    6. Leave rest as default and click on Next
    7. Choose AWS CodeBuild for Build Provider
    8. Enter the build project name
    9. Click on Next
    10. Select AWS CodeDeploy for Deploy provider
    11. Enter Application and deployment group name
    12. Click on Next
    13. review the pipeline and click on create pipeline
    14. The pipeline will automatically run
    15. After pipeline is successfull check the flask application in browser using instance-public-ip:5000