Experiment 15

Lab Objective:

This experiment aims to demonstrate the setup of a CI/CD pipeline using a cloud platform integrated with GitHub to automate the build, test, and deployment processes of a project.

Prerequisites:

● Basic knowledge of Git and GitHub.
● Access to a cloud platform like GitHub Actions, Jenkins, or CircleCI.
● A simple project (e.g., a basic web app or microservice).

Outcome:

1. Learn how to create a GitHub account and repository.
2. Understand how to integrate GitHub with a CI/CD pipeline on a cloud platform.
3. Experience configuring automated build, test, and deployment stages.

Description:

Task 1: Create a GitHub Account and Repository

● Go to GitHub and create a new account if you don't already have one.
● Once logged in, click on New to create a new repository.
● Name the repository and select the visibility (Public/Private).
● Optionally, add a README file and .gitignore file for your project.
● Clone the repository to your local machine using the following command

git clone https://github.com/<username>/<repository>.git

Task 2: Push Your Project Code to GitHub

● Create or add the code for your project to the repository's directory on your local machine.
● Add the changes to your Git staging area:

git add

● Commit the changes:

git commit -m "Initial project commit"

● Push the changes to your GitHub repository:

git push origin main

Task 3: Set Up the CI/CD Pipeline Using a Cloud Platform (GitHub Actions)

● Navigate to the Actions tab in your GitHub repository.
● Choose a pre-configured workflow or click New workflow to create your own CI/CD pipeline.
● Write a GitHub Actions workflow YAML file in the .github/workflows/ directory. For example, to build and test a Node.js application:

```
name: CI/CD Pipeline
on:
push:
branches:
- main
jobs:
build:
runs-on: ubuntu-latest
```

Experiment 15

steps:
- name: Checkout code
uses: actions/checkout@v2
- name: Set up Node.js
uses: actions/setup-node@v2
with:
node-version: '14'
- name: Install dependencies
run: npm install
- name: Run tests
run: npm test
- name: Build project
run: npm run build
● Commit the workflow file and push it to GitHub. This will trigger the CI/CD pipeline.
Task 4: Monitor the CI/CD Pipeline
● Go to the Actions tab in your GitHub repository to view the pipeline's progress.
● Ensure that the pipeline runs through the build and test stages successfully.
● Once the pipeline completes, verify that the changes are deployed if a deployment step is configured (e.g., deployment to a cloud platform like AWS, Heroku, or Google Cloud).
Note: Attach screenshots to your document that clearly showcase the steps and outcomes of your work for this lab

1. Introduction

Continuous Integration (CI) and Continuous Deployment (CD) are essential practices for modern software development. By automating the build, test, and deployment processes, CI/CD pipelines ensure that developers can deliver high-quality code quickly and with minimal manual intervention.

In this lab, we focus on GitHub Actions, a cloud-based CI/CD tool that is seamlessly integrated into GitHub. This experiment walks through the setup of an end-to-end pipeline that automates the process of building, testing, and deploying a basic web application whenever code is pushed to the GitHub repository. The use of GitHub Actions streamlines the process and reduces errors associated with manual intervention.

The experiment involves creating a GitHub account, setting up a repository, pushing code, and configuring an automated CI/CD pipeline through GitHub Actions. Additionally, we will monitor the pipeline, ensuring it functions correctly and deploying the application (optional).

2. Lab Objective

The primary goal of this experiment is to:
Learn how to create a GitHub account and repository.
Understand how to integrate GitHub with GitHub Actions to automate the CI/CD pipeline.
Experience the process of configuring automated build, test, and deployment stages for a basic project.

Gain hands-on experience with YAML-based workflow configuration for CI/CD automation.

Experiment 15

By the end of the experiment, you should be familiar with setting up a CI/CD pipeline using GitHub Actions, including integrating it with your project repository and monitoring pipeline execution.

3. Prerequisites

Before proceeding with the lab, the following prerequisites should be fulfilled:

Basic Knowledge of Git and GitHub: Understanding the fundamental Git commands (clone, commit, push) and how GitHub repositories work.

Access to GitHub: A GitHub account is required to create a repository and work with GitHub Actions.

A Simple Web Application: Any basic web application or microservice project, such as a Node.js or Python app, can be used for the CI/CD pipeline.

Git and Node.js (optional): Git should be installed locally to interact with repositories, and Node.js should be installed if you are using a Node.js application.

4. Expected Outcomes

Upon completing this lab, the following outcomes should be achieved:

GitHub Repository Creation: You will be able to create a GitHub account and repository, pushing your project code to the repository.

CI/CD Pipeline Setup: You will successfully set up and configure a GitHub Actions workflow to automate the build, test, and deployment processes for the project.

Automated Testing and Deployment: When you push changes to the GitHub repository, the GitHub Actions workflow will be triggered automatically, and the tests will be run. Upon success, the application will be deployed (if a deployment step is configured).

Monitoring the Pipeline: You will monitor the pipeline to ensure it runs smoothly through the build and test stages.

5. Lab Tasks
Task 1: Create a GitHub Account and Repository
Procedure:
Sign up/Login to GitHub:

Go to GitHub.

If you do not have an account, click on Sign up and follow the instructions. If you already have an account, simply log in.

Create a New Repository:

Experiment 15

Once logged in, click on the "+" icon in the top-right corner of the GitHub dashboard and select "New repository".

Choose a name for your repository (e.g., my-cicd-project).

Optionally, select the repository’s visibility (Public or Private).

Optionally initialize the repository with a README file and a .gitignore template for your project type.

Click on Create repository.

Clone the Repository Locally:

Copy the clone URL from the repository page.

Open a terminal on your local machine and clone the repository with the following command:

bash
Copy
Edit
git clone https://github.com/<username>/<repository>.git
cd <repository>
Task 2: Push Your Project Code to GitHub
Procedure:
Add Project Code to Repository:

Place your project code (e.g., a basic web application or Node.js app) in the repository’s directory on your local machine.

You can use an existing project or create a new one.

Stage and Commit Changes:

Stage all project files using the git add command:

bash
Copy
Edit
git add .
Commit the changes with an appropriate commit message:

bash
Copy
Edit
git commit -m "Initial project commit"
Push the Changes to GitHub:

Experiment 15

Push the committed code to the GitHub repository:

bash
Copy
Edit
git push origin main
Verification:

After pushing, check your GitHub repository to ensure the project files are visible.
Task 3: Set Up the CI/CD Pipeline Using GitHub Actions
Step 1: Navigate to the Actions Tab
Go to the Actions tab in your GitHub repository.

You can either select an existing template for the workflow or click "Set up a workflow yourself" to start creating your own CI/CD pipeline.

Step 2: Create the Workflow YAML File
Inside your project directory, create a new folder called .github/workflows/.

Create a new YAML file inside this folder, such as ci.yml, and add the following content to automate the build and test stages:

yaml
Copy
Edit

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'

    - name: Install dependencies
      run: npm install
```

Experiment 15

```
    - name: Run tests
      run: npm test

    - name: Build project
      run: npm run build
```

Note: Adjust the script as necessary to match your project structure and testing commands.

Step 3: Commit and Push the Workflow File
Add, commit, and push the new .github/workflows/ci.yml file to your GitHub repository:

bash
Copy
Edit

```
git add .github/workflows/ci.yml
git commit -m "Add GitHub Actions CI/CD workflow"
git push origin main
```

Task 4: Monitor the CI/CD Pipeline
Monitor the Pipeline Execution:

Navigate to the Actions tab of your GitHub repository.

You will see a list of workflow runs. Click on the latest run to view detailed logs and status.

Verify Pipeline Stages:

Check that the workflow runs through the build and test stages without errors.

Review the logs for each step to ensure all steps are completed successfully.

Verify Deployment (Optional):

If you have configured a deployment step, verify that the application is successfully deployed (e.g., to a cloud platform like Heroku, AWS, etc.).

6. Summary of Git Commands and GitHub Actions Workflow

Command / Action	Purpose
git clone	Clone a GitHub repository to your local machine.
git add .	Stage project files for committing.
git commit -m "<message>"	Commit changes to your local repository with a message.
git push origin main	Push changes from the local repository to GitHub.
GitHub Actions YAML Workflow	Automate the CI/CD pipeline (build, test, deploy).
actions/checkout@v2	Checkout the project code in the pipeline.
actions/setup-node@v2	Set up Node.js environment in the pipeline.

7. Conclusion

Experiment 15

This lab demonstrated the process of setting up a fully automated CI/CD pipeline using GitHub Actions. By leveraging this cloud-based tool, we were able to automate essential stages in the software development lifecycle, such as building, testing, and deploying code. The hands-on experience of configuring YAML workflows, managing GitHub repositories, and monitoring the execution of pipelines provided invaluable insight into how automation can enhance the efficiency and reliability of software delivery.

By completing this lab, you now have a clear understanding of how to integrate GitHub with a CI/CD pipeline, making it easier to manage and automate complex workflows.

8. References

GitHub Actions Documentation: https://docs.github.com/en/actions

GitHub CLI Reference: https://git-scm.com/docs

Node.js GitHub Action: https://github.com/actions/setup-node

Heroku GitHub Action (Optional Deployment): https://github.com/AkhileshNS/heroku-deploy

This detailed version of the report can be submitted as-is. Would you like me to help you format it into a Word document or PDF for easier submission?