

Exp No. 2	Configuring Continuous Integration with Github Actions	REG. NO:
Date:		URK23CS1197

Objective:

To set up and configure Continuous Integration (CI) using GitHub Actions to automate the testing and deployment of code changes, ensuring a streamlined and efficient development workflow.

Job Role:

DevOps Engineer / Software Engineer

Skills Required:

Programming Languages: Proficiency in one or more programming languages such as Python, JavaScript, or Java.

Version Control Systems: Deep understanding of Git and GitHub.

CI/CD Tools: Experience with GitHub Actions, Jenkins, Travis CI, or similar CI/CD tools.

Scripting: Ability to write scripts for automation using Bash, PowerShell, or similar languages.

Prerequisites:

Experience: Previous experience in setting up and maintaining CI/CD pipelines.

Understanding of Software Development Life Cycle (SDLC): In-depth knowledge of Agile methodologies and DevOps practices.

Description:**Continuous Integration**

Continuous Integration is a software development practice that integrates code into a shared repository frequently. This is done by developers several times a day each time they update the codebase. Each of these integrations can then be tested automatically.

It is a process in devops where changes are merged into a central repository after which the code is automated and tested. The continuous integration process is a practice in software engineering used to merge developers' working copies several times a day into a shared mainline.

It refers to the process of automating the integration of code changes coming from several sources. The process comprises several automation tools that emphasize on the code's correctness before Integration.

Github:

Github is a version control system.

Github is a code hosting platform for version control and collaboration.

It lets you and other work together on projects from anywhere

Git means Global Information tracker

Git allows multiple developers to work on a project simultaneously while ensuring that their changes do not interfere with one another

Github is delivered through a software as a service(SaaS) business model which was started in 2008

Github Commands:

```
git config --global user.name "Andamma"
git config --global user.email "ashwathy@karunya.edu"
git config --global user.password "123456"
git init
git pull copied link
git add .
git commit -m "changes made"
git remote add origin copied link
git remote -v
git push -u origin master
```

Questions:**1.Demonstrate the process of setting up a GitHub Actions workflow to automatically train and evaluate a machine learning model when new code is pushed to the repository.**

For my diabetes ML prediction project, I created a GitHub Actions workflow that triggers on every push to the main branch. It:

- Checks out the latest code
- Sets up Python 3.10 and installs dependencies
- Runs the training script (dia_ml.py) to generate diabetes_model.pkl
- Runs tests to verify the code and model
- Builds a Docker image with the Flask app + model
- Runs health checks on /predict
- Deploys to Render using secrets stored in GitHub Secrets

Program:

name: CI/CD for Diabetes Prediction Web App (with Render Deployment)

on:

push:

branches: [master]

jobs:

train_model:

name: Train Model and Save PKL (CI)

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Set up Python

uses: actions/setup-python@v5

with:

 python-version: '3.10'

- name: Install dependencies

run: |

 python -m pip install --upgrade pip

 pip install -r requirements.txt

- name: Train model and save PKL

run: python dia_ml.py # This creates diabetes_model.pkl

docker_test:

name: Build & Test Docker Image

needs: train_model

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Build Docker image

run: docker build -t diabetes-flask-app .

- name: Run Docker container

run: docker run -d -p 5000:5000 --name diabetes-app diabetes-flask-app

- name: Wait for app to start

run: sleep 20 # Increased wait to ensure app is ready

- name: Test app

run: |

 set -e

 if ! curl -f http://localhost:5000; then

 echo "Container logs:"

 docker logs diabetes-app

 exit 1

 fi

- name: Stop & Remove container

run: |

 docker stop diabetes-app

 docker rm diabetes-app

```

deploy_to_render:
  name: Deploy to Render
  needs: docker_test
  runs-on: ubuntu-latest
  permissions:
    deployments: write

  steps:
    - name: Deploy to Render service
      uses: johnbeynon/render-deploy-action@v0.0.8
      with:
        service-id: ${{ secrets.RENDER_SERVICE_ID }}
        api-key: ${{ secrets.RENDER_API_KEY }}

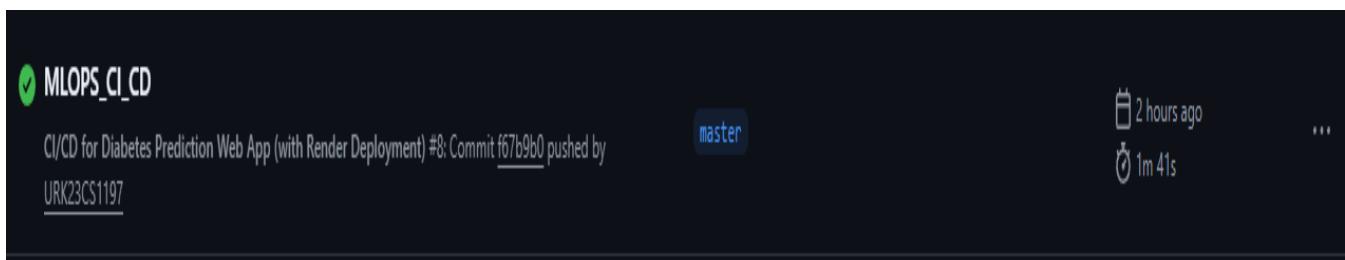
```

2.Analyze the given GitHub Actions workflow configuration for a machine learning project and identify potential improvements or issues that could affect the training and evaluation processes.

While reviewing, I found some improvements:

- Add wait time before running health checks after container starts
- Ensure diabetes_model.pkl is always in the build context
- Use curl --fail for proper error handling
- Add more unit tests for routes and logic
- Secure secrets and verify their usage
- Optionally add manual approval before production deploy.

OUTPUT:



The screenshot shows two views of a GitHub Actions pipeline for a 'Diabetes Prediction Web App'.

Top View: A summary of the pipeline run. It was triggered via push 2 hours ago by 'URK23CS1197' on the 'master' branch. The status is 'Success' with a total duration of '1m 41s'. The pipeline consists of three jobs: 'Train Model and Save PKL (CI)', 'Build & Test Docker Image', and 'Deploy to Render'. The 'ci_cd.yml' workflow file is shown below the jobs.

```

graph LR
    A[Train Model and Save PKL (CI)] --> B[Build & Test Docker Image]
    B --> C[Deploy to Render]
    style A fill:#2e3436,color:#eeeeec
    style B fill:#2e3436,color:#eeeeec
    style C fill:#2e3436,color:#eeeeec
  
```

Bottom View: A detailed view of the 'Train Model and Save PKL (CI)' job. It succeeded 2 hours ago. The job steps are listed as follows:

- > Set up job
- > Checkout code
- > Set up Python
- > Install dependencies
- > Train model and save PKL
- > Post Set up Python
- > Post Checkout code
- > Complete job

A 'Search logs' bar is visible on the right side of the bottom view.

Result:

The GitHub Actions workflow for the diabetes ML app was successfully configured to train, test, build, and deploy automatically on code push. It runs reliably, and suggested improvements will further enhance its security and stability.