Name:Nikhil Sharma

Class:23BCC-1

UID:23BCC70030

Set Up CI/CD Pipeline using GitHub Actions

Aim:

The aim of this experiment is to automate the build, test, and deployment process of a project using GitHub Actions. This ensures that code changes are continuously integrated, verified, and deployed, reducing manual effort and improving software quality and delivery speed.

Theory:

1. **CI/CD Concepts**
    - **CI (Continuous Integration):** Automatically integrates code changes from multiple contributors into a shared repository, followed by automated build and testing.
    - **CD (Continuous Deployment/Delivery):** Automatically deploys code to production (or staging) after passing tests and build steps.
    - Benefits: Early bug detection, faster release cycles, and reliable deployments.
2. **GitHub Actions**
    - GitHub Actions is a CI/CD platform integrated with GitHub that allows you to automate workflows triggered by events like push, pull request, or schedule.
    - Workflows are defined in YAML files stored in `.github/workflows/`.
    - Each workflow contains:
        - **Triggers (on)**: Events that start the workflow (e.g., `push`, `pull_request`).
        - **Jobs**: A set of tasks executed on virtual environments (e.g., `ubuntu-latest`).
        - **Steps**: Individual commands or actions in a job (e.g., install dependencies, run tests, deploy).
3. **Common Steps in CI/CD with GitHub Actions**
    - Checkout the repository.

    - Set up environment (Node, Python, Java, etc.).
    - Install dependencies.
    - Run tests.
    - Build the application.
    - Deploy to cloud services (e.g., Vercel, AWS, Heroku).

Learning Outcomes:

1. Understand the concepts of CI/CD and its advantages in software development.
2. Configure and automate workflows using GitHub Actions.
3. Set up automated build, test, and deployment pipelines for any project.
4. Reduce manual effort in project deployment and ensure consistent software quality.
5. Gain practical knowledge of YAML workflow syntax and GitHub ecosystem integration.

Materials Required:

- GitHub account
- Project repository on GitHub
- Node.js / Python / Java (depending on project)
- Internet connection
- Cloud deployment platform (optional, e.g., Vercel, Heroku)

Procedure / Steps:

1. **Create or Open a GitHub Repository**
    - Go to GitHub and create a new repository or use an existing one.
    - Clone it to your local system if needed.
2. **Create a Workflow Directory**

   ```
   mkdir -p .github/workflows
   ```

3. **Create a Workflow File**
    - Example: `.github/workflows/ci-cd.yml`
4. **Define Workflow Trigger**

   ```
   name: CI/CD Pipeline
   on:
     push:
   ```

```
    branches: [main]
  pull_request:
    branches: [main]
```

5. **Define Jobs and Steps**

```
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v3

      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install Dependencies
        run: npm install

      - name: Run Tests
        run: npm test

      - name: Build Application
        run: npm run build

      - name: Deploy to Vercel
        uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.VERCEL_ORG_ID }}
          vercel-project-id: ${{ secrets.VERCEL_PROJECT_ID }}
          working-directory: ./
          prod: true
```

6. **Add Secrets for Deployment**
   - Go to **GitHub → Settings → Secrets → Actions**.
   - Add necessary secrets like `VERCEL_TOKEN`, `VERCEL_ORG_ID`, `VERCEL_PROJECT_ID`.

7. **Push Workflow to GitHub**

```
git add .github/workflows/ci-cd.yml
git commit -m "Add CI/CD pipeline"
git push origin main
```

8. **Verify CI/CD Execution**

- Go to GitHub repository → **Actions** tab.
- You should see the workflow running on push or pull request.
- Check logs for each step to verify build, test, and deployment.
