Deploying the Swiggy Application Using GitHub Actions as a CI/CD Tool

Setting Up GitHub Actions Workflow

Step 1: Create a `.github` Directory

In your repository, create a directory named `.github` in the root folder. Inside `.github`, create a folder named `workflows`.

mkdir -p .github/workflows

Step 2: Create a Workflow File

Inside the `workflows` folder, create a YAML file (e.g., `deploy.yml`). This file will define the CI/CD workflow.

```
# .github/workflows/deploy.yml
name: Deploy Swiggy Application

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
```

```
steps:
  # Step 1: Checkout the code
  - name: Checkout Code
    uses: actions/checkout@v3

  # Step 2: Set up Node.js (if applicable)
  - name: Set up Node.js
    uses: actions/setup-node@v3
    with:
      node-version: '16'

  # Step 3: Install dependencies
  - name: Install Dependencies
    run: npm install

  # Step 4: Run tests
  - name: Run Tests
    run: npm test

  # Step 5: Build the application
  - name: Build Application
    run: npm run build

  # Step 6: Deploy to Hosting Platform
  - name: Deploy to Hosting Platform
    env:
      HOSTING_API_KEY: ${{ secrets.HOSTING_API_KEY }}
    run: |
      # Example deployment command
      echo "Deploying application..."
      # Add deployment script here (e.g., AWS CLI, Azure CLI, etc.)
```

Explanation of the Workflow File

- `on`: *Specifies the events that trigger the workflow. Here, the workflow is triggered on* `push` *and* `pull_request` *to the* `main` *branch.*
- `jobs`: *Defines the tasks to be performed.*
- `steps`: *Contains individual tasks such as checking out code, setting up Node.js, running tests, and deploying the application.*
- `secrets.HOSTING_API_KEY`: *A secure way to store sensitive information like API keys.*

4. Setting Up Secrets in GitHub

To deploy the application, you may need credentials such as API keys or SSH keys. These can be securely stored as secrets in GitHub.

Steps to Add Secrets:

1. *Go to your GitHub repository.*
2. *Navigate to* ***Settings > Secrets and variables > Actions****.*
3. *Click* ***New repository secret****.*
4. *Add secrets like* `HOSTING_API_KEY` *with their respective values.*

5. Deploying the Application

Step 1: Push Code to GitHub

Commit your changes and push the code to the `main` *branch.*

```
git add .
git commit -m "Set up GitHub Actions workflow"
```

git push origin main

Step 2: Trigger the Workflow

Once the code is pushed, GitHub Actions will automatically trigger the workflow.

Step 3: Monitor Workflow Execution

1. *Go to the **Actions** tab in your repository.*
2. *Select the running workflow.*
3. *Monitor the logs to ensure all steps complete successfully.*

Step 4: Verify Deployment

Check the hosting platform to ensure the Swiggy application is deployed and functioning correctly.

6. Best Practices

1. ***Use Environment Variables**: Store sensitive data securely using GitHub secrets.*
2. ***Run Tests**: Always include testing steps to catch issues early.*
3. ***Monitor Workflows**: Regularly monitor CI/CD pipelines to address failures.*
4. ***Version Control**: Use branches to test new features before merging into the main branch.*
5. ***Scalable Hosting**: Choose a hosting platform that supports scalability for increased traffic.*

7. Common Issues and Troubleshooting

Issue 1: Workflow Fails

- ***Cause:*** *Missing dependencies or incorrect configuration.*
- ***Solution:*** *Check the logs under the Actions tab to identify and fix errors.*

Issue 2: Deployment Fails

- ***Cause:*** *Incorrect API keys or hosting configuration.*
- ***Solution:*** *Verify secrets and deployment commands.*

Issue 3: Application Errors Post Deployment

- ***Cause:*** *Bugs in the code or incorrect environment setup.*
- ***Solution:*** *Debug the application logs and resolve issues.*