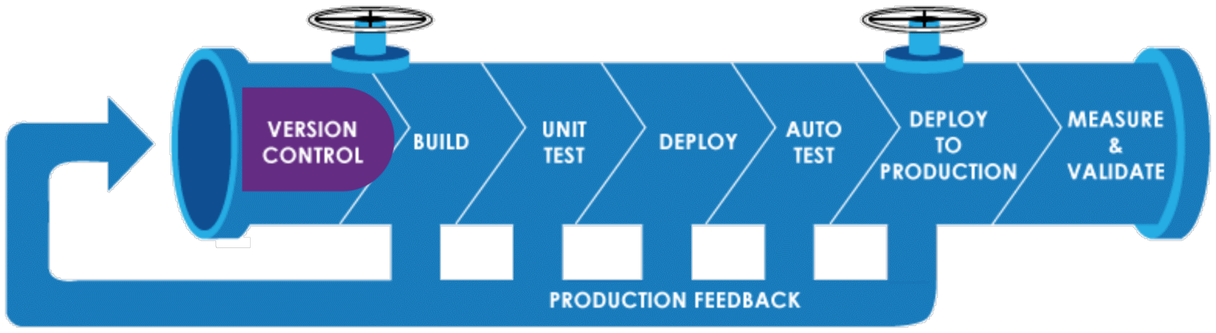
Step in to CI/CD: A Hands-On Guide to Building CI/CD Pipeline with GitHub Actions



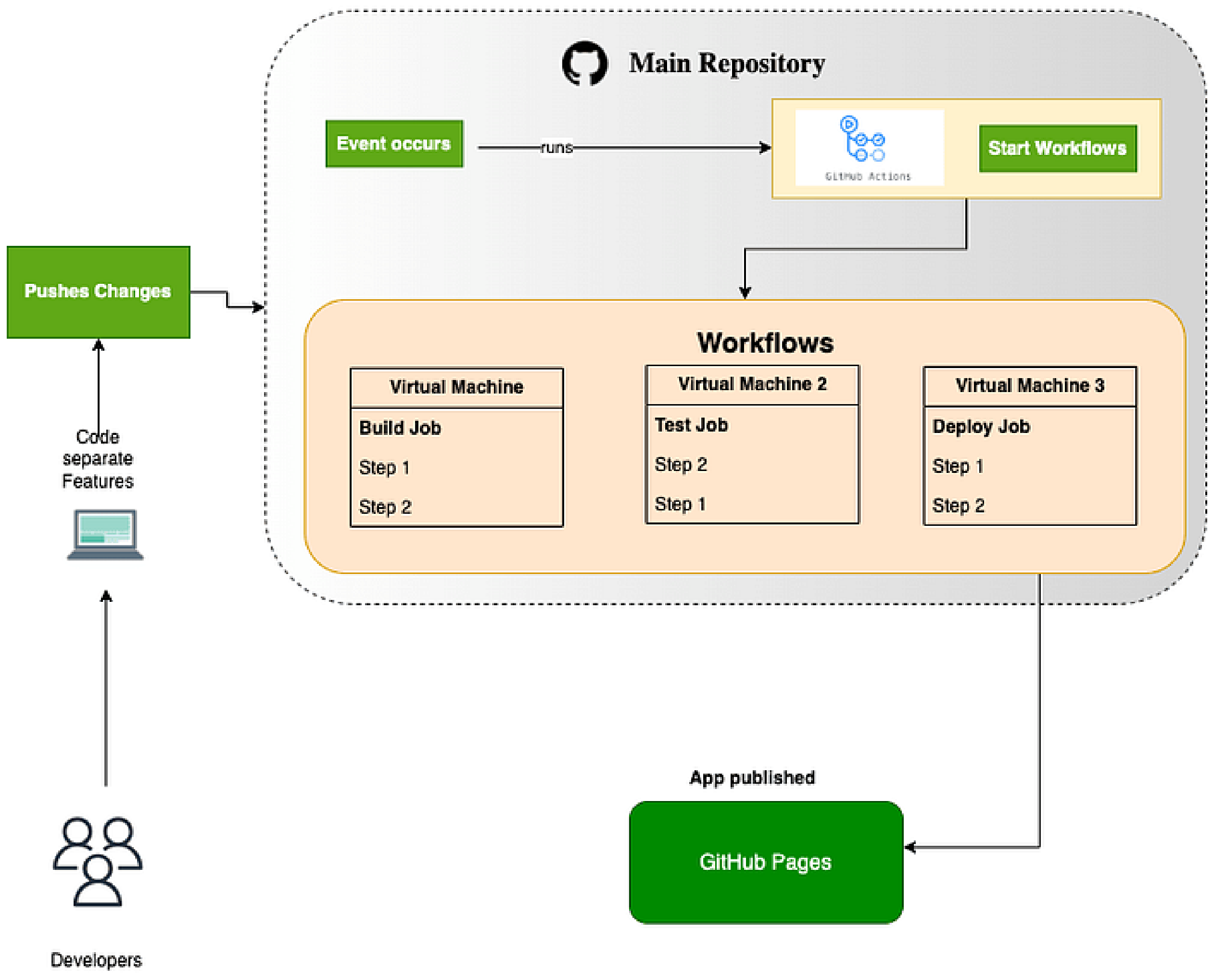
In this article, you will learn how to implement a CI/CD pipeline using github action. In order to implement that i will take a react Todo app as a software product and github pages as the deployment environment.
Before move in to the implementation, I'll briefly clarify terms of the CICD (I am not going to explain this in a in details here but in a nutshell)

Continuous Integration (CI)
Assume In a collaborative development environment, multiple developers work on individual features. Upon completing a feature, they seamlessly merge their code with the central repository of the project. This is doing Continuously.That's what we called "continuous integration"

Continuous Delivery (CD)
This phase involves systematically building, testing, and pushing code changes to a staging environment, ensuring a robust pre-production validation process.

Continuous Deployment (CD)
The ultimate step involves the automatic launch and distribution of the final product (artifacts) to the production environment / end-users.

GitHub Actions — The CI/CD Platform
Press enter or click to view image in full size



GitHub Actions stands out as a popular CI/CD platform, empowering developers to seamlessly build, test, and deploy applications directly within their GitHub repositories. The flow of CI/CD processes is facilitated through workflows — workflows are automated processes that execute one or more predefined tasks.

Workflows can be tailored to meet specific needs, such as building and testing every pull request or deploying merged pull requests to the production environment. These workflows are clearly defined within a YAML file in your local repository and subsequently pushed to the GitHub repository.

Okay. let’s go with practical scenario

Here are the steps.

1. Create your React application (in here i have already created a todo application using create-react-app).
2. Create a GitHub repository for the project (you can do by your self)
3. Set up CI/CD workflow with GitHub Actions in your project directory

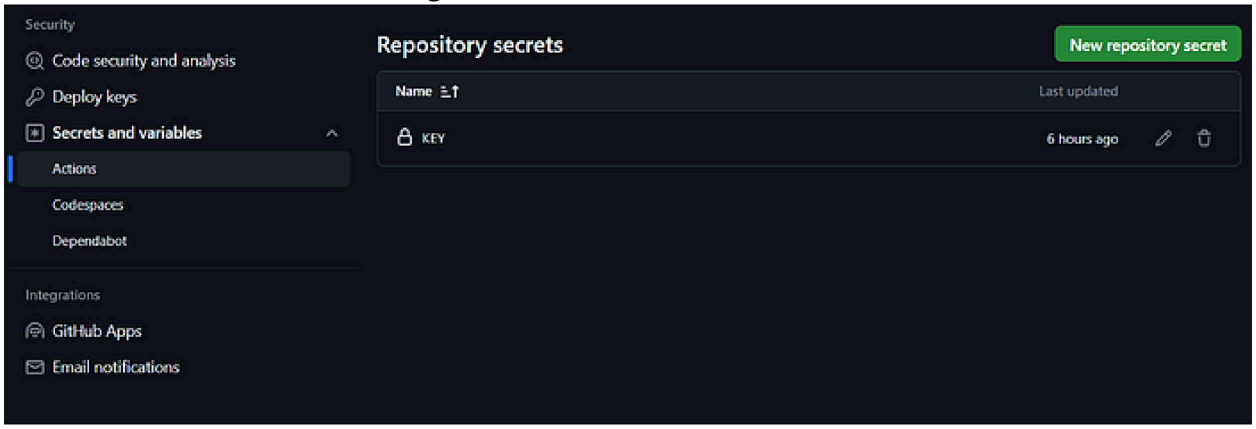
follow this,

- Create a access token in GitHub (*see here*)
- use the access token in our Actions secrets in the repository (This is for github action to access our repository)

To do this Follow this step

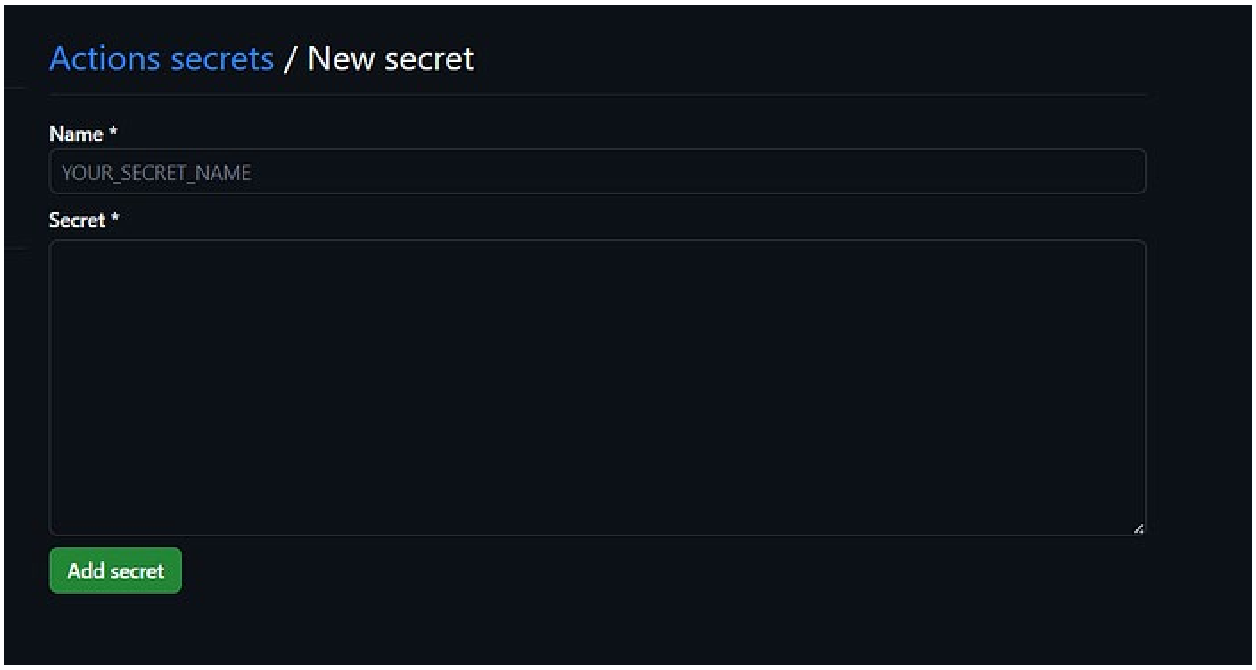
- Click on the Settings tab in your repository
- In the left sidebar, select Secrets and variables
- From the dropdown, select Actions

Press enter or click to view image in full size



- Click on the New repository secret button
- Enter the secret name in the Name field(This is going to be used in our workflow file)
- Paste your copied personal access token in the secret field and add.

Press enter or click to view image in full size



Now you’re complete..

Let's create the yml file!

- Navigate to the root of your project directory and create a directory named "**.github**"
- Inside the ".github", create a directory named "**workflows**" to store your workflow files.
- In the **.github/workflows** directory, create a new file called **main.yml** (you can name it your own name. it is entirely up to you) and add the following code. (*make sure modify your diretory name in bellow code I have used my directory name* as "**todo**")

```
name: CI/CD for react Todo

on:
  push:
    branches:
      - main

jobs:
  # Build Job
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3

      - name: Install Node
        uses: actions/setup-node@v3
        with:
          node-version: 18.x

      - name: Install Dependencies
        run: |
          cd todo
          npm install

      - name: Build Project
        run: |
          cd todo
          npm run build

      - name: Upload artifact to enable deployment
        uses: actions/upload-artifact@v3
        with:
          name: poduction file
          path: ./todo/build
```

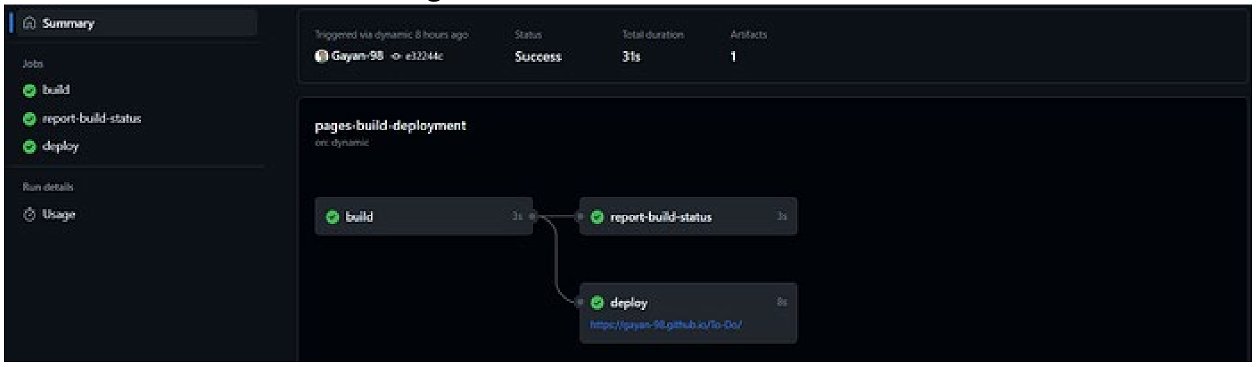
```
 # Deploy Job
 deploy:

   needs: build

   runs-on: ubuntu-latest
   steps:
    - name: Download artifact
      uses: actions/download-artifact@v3
      with:
        name: poduction file
        path: ./todo/build

    - name: Deploy to GitHub Pages
      uses: peaceiris/actions-gh-pages@v3
      with:
        github_token: ${{ secrets.KEY }} #this is the key of the access token
        publish_dir: ./todo/build
```

Update the package.json
Add the homepage of the app to the package.json file
"homepage": "https://{your github id }.github.io/{repo name}/",
if you are done with all the things up to now, you can commit them, and push to your GitHub repository. The GitHub Actions automatically takes care of each step. Simply navigate to the 'Actions' tab within your repository to see the automated workflow in action. Furthermore hands-on experience of the pipeline's automation, try making additional code changes, commit them, and push to the repository. Then your updates will seamlessly render on GitHub Pages without requiring any manual intervention.
Press enter or click to view image in full size



***here* the source code:** https://github.com/Gayan-98/To-Do