**GitHub-Actions**

GitHub Actions is a powerful, integrated CI/CD service provided by GitHub that allows developers to automate their build, test, and deployment pipeline directly within their GitHub repositories.

GitHub Actions makes it possible for developers to create workflows that automatically execute upon specific GitHub events, such as a push to a repository or a pull request creation. These workflows are defined in YAML files and can run on GitHub-hosted runners (virtual machines) or self-hosted runners that developer will manage.

These workflows will be created inside developer’s repository inside “.github/workflows”.

***Key-Features of GitHub Actions :-***

* Automated Workflow – Developers can automate their development process from code integration, testing, to deployment with workflows that can be triggered on specific GitHub events.
* Hosted Runners – Developers can run their workflows on virtual machines hosted by GitHub with various OS like Linux/Windows/MacOS. Developers can use their own machine as well to run their workflow i.e self-hosted runners.
* Matrix Builds – Developers can run their applications across different version of language and environments by defining a matrix strategy in the workflow.
* Docker Container Support – Developers can build and push their docker images or run actions inside docker containers as well.
* MarketPlace – Developers can access a marketplace full of actions developed by the community and GitHub to automate every step of the development process.

***Uses of GitHub Actions :-***

* Continuous Integration (CI) – Developers can automatically run, build and test their code each time a change is pushed to their repository.
* Continuous Deployment (CD) – Developers can automate their deployement of code to the Prod environment each time CI workflow succeeds, ensuring that application is always up-to-date.
* Automating Workflows – Beyond CI/CD, it can also be used so many other workflows related to the development process including managing issues, automating project boards, welcome new developer etc.
* Scheduled Task – it can also be used to run scheduled tasks in a cron syntax useful for routine maintainance and data fetching tasks etc.
* Custom Automation – Developers can create their own custom workflow such as sending notification on Slack, greeting new contributors or automating code reviews etc.
* Security and Compliance – Developers can integrate security and compliance checks into CI/CD pipeline to automatically scan for vulnerabilities in dependencies or enforce coding standards.

***Benifits of GitHub Actions :-***

* Integration with GitHub – Being an integral part of GitHub, actions allows for seamless automation within the GitHub ecosystem without the need of external services like Jenkins etc.
* Flexibility – GitHub Actions supports a wide range of languages and frameworks, and workflows can be customized to meet the specific needs of a project.
* Community and Marketplace – GitHub Actions provides a open marketplace where there are so many already created actions that Developers can directly use in their workflow rather then to create every action each time.

***Terminologies of GitHub Actions :-***

* Workflow – A workflow is an automated process that developers set up in their repository. It consist of one or more jobs and can be triggered by GitHub events (e.g a pull or push request) or scheduled to run at specific times. Workflows are defined in ‘.yml’ or ‘.yaml’ files and stored in the “.github/workflows” directory of the repository.

The name of the workflow is specified with the ‘name’ field and the trigger event is specified with the ‘on’ field.

name: CI Workflow

on: [push, pull\_request]

* Job – A job is a set of steps that execute on the same runner. By default, jobs run in parallel but can be configured to run sequentially as well. Each job runs in a fresh instance of the virtual environment specified by “runs-on” or a self-hosted runner.

Jobs are specified with a key under the ‘jobs’ field.

jobs:

test:

runs-on : ubuntu-latest

build:

runs-on : ubuntu-latest

* Step – A step is an individual task that can run commands or actions. A job consist of one or more steps. Steps in a job are executed sequentially.

steps:

* name: Checkout code

uses: actions/checkout@v2

* name: Run a script

run: echo “Hello, world!!”

* Action – Actions are standalone commands that are combined into steps to create a job. Actions can be reused and can be community-maintained in the GitHub Marketplace or defined in Developer’s repository. An action can be referenced in a step with the ‘uses’ field, specifying the action name and version.
* uses: actions/checkout@v2
* Runner – A runner is a server that has the GitHub Actions runner application installed. It listens for available jobs, runs one job at a time, and reports the progress, logs, and results back to GitHub. Runners can be GitHub-hosted or self-hosted.

runs-on: ubuntu-latest

* Artifacts – Artifacts are files created during jobs, which can be saved and shared with other jobs in the workflow, or downloaded once the workflow has finished. This is useful for passing output between jobs or storing logs and other files that are generated by the workflow.

Artifacts aren’t directly defined in the workflow yaml, we use actions to upload or download artifacts.

* uses: actions/upload-artifact@v2

with:

name: My Artifact

path: path/to/artifact

* Cache – Caching dependencies and other frequently used files can speed up workflow execution time. GitHub Actions provides caching capabilities to reuse files downloaded in previous runs of the workflow, reducing execution time.

Caching is implemented with the ‘actions/cache’ action. This action allows us to specify paths to cache and a key for cache restoration.

* uses: actions/cache@v2

with:

path: |

path/to/dependencies

key: ${{ runner.os }}-deps-${{ hashFiles(‘\*\*/lockfile’) }}

* Events – An event is a specific activity that triggers a workflow. GitHub Actions supports a wide range of events, such as ‘push’, ‘pull\_request’, ‘shcedule’ (for cron jobs), and ‘repository\_dispatch’, among others. Workflow can be configured to run on one or more of these events. Events are specified under the ‘on’ field.

on:

push:

branches: [ main ]

pull\_request:

branches: [ main ]

* Secrets – Secrets are encrypted environment variables that Developers create in a repository or organization. They can store sensitive information, such as tokens, keys, and passwords, required for workflow to authenticate and gain access to external services.

Secrets are referenced in steps using the ‘secrets’ context but are not defined in the YAML file. We define them in the repo or organization setting.

env:

ACCESS\_TOKEN: ${{ secrets.ACCESS\_TOKEN }}

* Marketplace – The GitHub Marketplace is a central location where we can find and share actions to use in workflows. These actions are developed by GitHub and the GitHub community and cover a wide range of automation scenarios.

MATRIX –

jobs:

build\_and\_test:

strategy:

matrix:

version: [ “14”, “16”, “18” ]

os: [ “ubuntu-latest”, “windows-latest” ]

runs-on: ${{ matrix.os }}

steps:

* name: Setup node.js

uses: actions/setup-node@v3

with:

node-version: ${{ matrix.version}}

* name: Install Dependencies and Test

run: npm install && npm run test