**📖 Table of Contents**

1. What is GitHub Actions?
2. Core Concepts
3. Workflow File Structure
4. Commonly Used Triggers (Events)
5. Jobs and Runners
6. Steps and Actions
7. Environment Variables & Secrets
8. Matrix Strategy
9. Reusable Workflows
10. Caching & Artifacts
11. Deployment Integration
12. Best Practices
13. Sample Workflow Examples

**1. 🚀 What is GitHub Actions?**

GitHub Actions is a **CI/CD platform** that enables you to **automate your build, test, and deployment pipeline** directly in your GitHub repositories using YAML syntax.

**Key Benefits:**

* Native integration with GitHub.
* Built-in runners (Linux, Windows, macOS).
* Supports Docker containers.
* Reusable workflows and composite actions.

**2. 🧠 Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| Workflow | Automation process defined in YAML and triggered by events. |
| Event | A GitHub activity (e.g., push, pull\_request) that triggers a workflow. |
| Job | A set of steps executed on the same runner. |
| Step | Individual task within a job (can be shell commands or action invocations). |
| Runner | Machine that runs workflows (hosted or self-hosted). |
| Action | Reusable unit of code (custom or marketplace) used in steps. |

**3. 📁 Workflow File Structure**

Workflow files are stored in: .github/workflows/<your-workflow>.yml

**Example:**

name: CI Pipeline

on:

push:

branches: [ main ]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Run build

run: npm run build

**4. ⏱️ Commonly Used Triggers (Events)**

| **Event** | **Description** |
| --- | --- |
| push | Triggered on code push to the repository |
| pull\_request | Triggered when a PR is opened, synchronized, etc. |
| schedule | Triggered at scheduled intervals (cron format) |
| workflow\_dispatch | Manually triggered workflows |
| release | Triggered on release events |

**5. 🧱 Jobs and Runners**

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Echo Hello

run: echo "Hello, world!"

* runs-on: Defines the type of runner.
* Job dependencies can be managed using needs:.

**6. 🔄 Steps and Actions**

**Using an Action:**

- uses: actions/checkout@v3

**Running a Command:**

- name: Install Dependencies

run: npm install

**7. 🔐 Environment Variables & Secrets**

**Using Environment Variables:**

env:

NODE\_ENV: production

**Using GitHub Secrets:**

- name: Use Secret

run: echo "${{ secrets.MY\_SECRET }}"

**8. 🎯 Matrix Strategy**

Run tests in multiple environments:

strategy:

matrix:

node-version: [14, 16, 18]

**9. ♻️ Reusable Workflows** Define a reusable workflow in .github/workflows/deploy.yml:

on:

workflow\_call:

inputs:

environment:

required: true

type: string

Call it in another workflow:

jobs:

call-deploy:

uses: ./.github/workflows/deploy.yml

with:

environment: production

**10. 📦 Caching & Artifacts**

**Caching Dependencies:**

- uses: actions/cache@v3

with:

path: ~/.npm

key: ${{ runner.os }}-node-${{ hashFiles('\*\*/package-lock.json') }}

**Upload/Download Artifacts:**

- uses: actions/upload-artifact@v3

with:

name: build-output

path: dist/

**11. 🚀 Deployment Integration**

You can deploy to environments like AWS, Azure, Docker Hub, or Firebase.

**Azure Example:**

- name: Azure Login

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

**12. ✅ Best Practices**

* Use actions/checkout in all workflows.
* Store secrets securely in GitHub Settings.
* Reuse workflows to reduce duplication.
* Use caching to speed up builds.
* Keep workflows modular and readable.

**13. 🧪 Sample CI Workflow**

name: Node.js CI

on: [push, pull\_request]

jobs:

build:

runs-on: ubuntu-latest

strategy:

matrix:

node-version: [14, 16]

steps:

- uses: actions/checkout@v3

- name: Use Node.js ${{ matrix.node-version }}

uses: actions/setup-node@v4

with:

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

- run: npm ci

- run: npm test