Github Actions

GitHub Actions

Presented by

Appvengers

Oct 2025

Content

GitHub Actions

What is GitHub Actions?

GitHub Actions is a continuous integration and continuous deployment (CI/CD) platform built directly into GitHub. It enables developers to automate software workflows, including building, testing, and deploying code, all within their GitHub repositories. Launched in November 2019, GitHub Actions has become one of the most popular automation tools in modern software development, offering seamless integration with the GitHub ecosystem.

When to use GitHub Actions?

There are multitude of reasons to use GitHub Actions, and it’s really up for a Developer on when to use it. If you really feel like you need to cut some time and stop doing the things that are quite repetitive and manually exhausting, that’s when you know you really need to

AUTOMATE THINGS

Why is it useful for Angular Projects

GitHub Actions can be very useful as it provides a lot of benefits for a developer when working for any modern frontend or full-stack framework such as Angular.

- Automated CI/CD Pipelines
- Effortless Testing
- Consistent Builds
- Automated Dependency Updates (tools like ‘Dependabot’)
- Deployment Made Easy

Workflow Components

YAML (.yml)

Workflows are defined in YAML files inside

.github/workflows/<workflow-name>.yml

Components

- Events/Trigger
- Jobs
- Runners
- Steps
- Actions
- Commands
- Environment variables
- Dependencies & Outputs

Workflows

A workflow is a configurable automated process that will run one or more jobs.

Defined by **YAML** file (.yaml / .yml) and is stored in the .github/workflows directory in a repository

Events

An event is a specific activity in a repository that **triggers a workflow run.**

There are different triggers to trigger an event: on **push, pull_request, workflow_dispatch** etc.

You can also select what branch will trigger the event.

Jobs

A job is a set of steps in a workflow that is executed on the same runner.

Container of **steps**: something that you want to happen if an event is triggered.

Actions

An action is a pre-defined, reusable set of jobs or code that performs specific tasks within a workflow.

Runners

A runner is a server that runs your workflows when they're triggered.

Github hosted-runners:

1. **Ubuntu Linux** (Most common used in workflows)
2. **Microsoft Windows**
3. **macOs**

Serves as a **virtual machine** that executes commands inside a job.

Workflow Syntax

name

is the name of your workflow.

```
GitHub Workflow - YAML GitHub Wo
# Name of the Workflow.
name: Push Workflow

```

on

To define which events can cause the workflow to run.

You can define single or multiple events that can trigger a workflow, or set a time schedule.

```
on:
  push: # Trigger Event
```

Event Examples:

1. **push**
2. **pull_request**
3. **workflow_dispatch**
4. **etc.**

on.schedule

define a time schedule for your workflows.

```
on:
  schedule:
    - cron:  '59 9 * * *' # Command Run On Notice
```

jobs

A job is a **set of steps** in a workflow that is executed on the same runner.

```yaml
jobs:
  push-workflow: # Personal Job Identifier
    name: First Job # Displayed in the Github Actions UI.
    runs-on: ubuntu-latest # VM where commands will be executed.

    steps: # Processes/Steps that the job will execute.
      - name: Print Greeting
        run: echo "Hello Github Actions!" # Print "Hello Github A
```

jobs.<job_id>.env

Sets **variables** for steps to use in the runner environment.

A map of **variables** that are available to all steps in the job.

```yaml
jobs:
  display-variable:
    runs-on: ubuntu-latest
    env:
      FIRST_NAME: John

    steps:
      - name: "display first name variable"
        run: echo "Hello, $FIRST_NAME!"
```

jobs.<job_id>.name

Is used to set a name for the job, which is displayed in the GitHub UI.

```
jobs:
  push-workflow: # Per
    name: First Job #
    runs-on: ubuntu-la

    steps: # Processes
      - name: Print Gr
        run: echo "Hel
```

jobs.<job_id>.runs-on

Is used to define the type of **machine** to run the job on.

- **Ubuntu Linux**
- **Microsoft Windows**
- **macOs**

```
s:
ush-workflow: # Personal
 name: First Job # Displa
 runs-on: ubuntu-latest #

 steps: # Processes/Steps
   - name: Print Greeting
     run: echo "Hello Git
```

jobs.<job_id>.steps

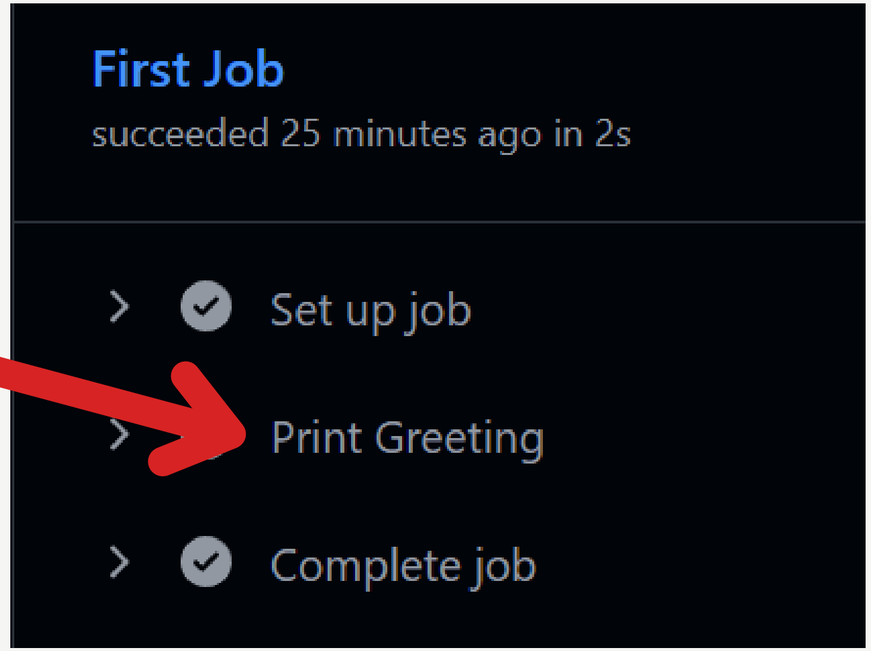
A job contains a sequence of tasks called **steps**.

```
steps: # Processes/Steps that the job will execute.
  - name: Print Greeting
    run: echo "Hello Github Actions!" # Print "Hello
```

jobs.<job_id>.steps[*].name

A name for your step to display on GitHub.

```yaml
steps: # Processes/Steps
  - name: Print Greeting
```



jobs.<job_id>.steps[*].uses

Used to select an action that is already defined and can be reused.

```
steps:
- uses: actions/checkout@v4
```

actions/checkout@v4 - checks out your repository code.

jobs.<job_id>.steps[*].run

Executes command-line programs using the operating system's shell.

```
steps: # Processes/Steps that the job
  - name: Print Greeting
    run: echo "Hello Github Actions!"
```

DEMONSTRATION

Automating Unit Test

DEMONSTRATION

1. Create or prepare an angular project **(ng new [name])**
2. **Go to package.json** and add this line of code inside the **scripts:**
   a. “test:ci”: “ng test --no-watch --no-progress --browsers=ChromeHeadless”
3. Connect your angular project to your remote repo
4. Commit and push all files to your repository
5. Open a terminal and type **“npm run test:ci”**

```
PS C:\Users\Justine\Desktop\repos\demo> npm run test:ci

 > demo@0.0.0 test:ci
 > ng test --no-watch --no-progress --browsers=ChromeHeadless

 Initial chunk files   | Names              |  Raw size
 chunk-OVIQD2LZ.js     | -                  |   2.49 MB |
 polyfills.js          | polyfills          | 943.14 kB |
 spec-app.spec.js      | spec-app.spec      | 230.71 kB |
 test_main.js          | test_main          |  21.73 kB |
 jasmine-cleanup-0.js  | jasmine-cleanup-0  | 519 bytes |
 styles.css            | styles             |  96 bytes |

                       | Initial total      |   3.69 MB

 Application bundle generation complete. [3.454 seconds] - 2025-10-08T15:10:01.644Z

 08 10 2025 23:10:01.845:INFO [karma-server]: Karma v6.4.4 server started at http://localhost:9876/
 08 10 2025 23:10:01.847:INFO [launcher]: Launching browsers ChromeHeadless with concurrency unlimited
 08 10 2025 23:10:01.861:INFO [launcher]: Starting browser ChromeHeadless
 08 10 2025 23:10:03.356:INFO [Chrome Headless 141.0.0.0 (Windows 10)]: Connected on socket eo5vNwnS57R2xSy3AAAB with id 31
 115876
 Chrome Headless 141.0.0.0 (Windows 10): Executed 2 of 2 SUCCESS (0.139 secs / 0.126 secs)
 TOTAL: 2 SUCCESS
PS C:\Users\Justine\Desktop\repos\demo>
```

DEMONSTRATION: Integrate test:ci to Github Actions

6. Create folder **“.github/workflows”**
7. Inside the “.github/workflows” folder, create a new file **“ci.yml”**
8. Copy this **code** and paste it to your own **“ci.yml”.**
9. Commit and push to your own repository.
10. Check workflow status through **actions** in your remote repository.

DEMONSTRATION

Prerequisites

1. Git Repository
2. .yml file stored in .github/workflow directory
3. Docker and act [for local workflow]

Running GitHub Actions Locally

winget install nektos.act

```
James Michael in ~
❯ winget install nektos.act
Found an existing package already installed. Trying to upgrade the installed package...
No available upgrade found.
No newer package versions are available from the configured sources.

James Michael in ~ took 1s
❯
```

```
James Michael in ~
❯ act --version
act version 0.2.81
```

DEMONSTRATION

Apply ESLint in Pull Requests

Practical Use Cases

1. Running tests on every pull request.
2. Checking code quality (lint, prettier).
3. Running scheduled tasks (like cron jobs).

Advantages of Github Actions

Easy integration with GitHub

Github Actions is built-in in Github.

Protection for main branch

You can set branch rule protection and set status checks before merging.

Automation

GitHub Actions can automate your workflow, allowing you to build, test, and deploy your code right from GitHub.

Customization

GitHub Actions are highly customizable. You can create your own actions or use actions from the GitHub Marketplace to build workflows that meet your specific needs.

Disadvantages of Github Actions

Learning curve with YAML syntax

Learning YAML can be overwhelming.

Requires basic knowledge in Github

Learning the foundation first before proceeding to automation might be required to fully understand how GitHub Actions integrates and automates repository workflows.

Limited free minutes for private repos.

There are usage limits for GitHub Actions workflows. Usage charges apply to repositories that go beyond the amount of free minutes and storage for a repository.

Summary

GitHub Actions is a powerful CI/CD tool that automates repetitive tasks such as checking code logic, syntax, and even deployment. It supports multiple workflows depending on the developer’s needs, such as security checks, automated testing, or continuous integration. This flexibility allows developers to maintain cleaner and more secure codebases while adding an extra layer of protection for the main branch.

Thank you

for listening

References

https://docs.github.com/en/actions/get-started/understand-github-actions

https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-syntax

https://www.youtube.com/watch?v=1vqJ1_AAcUg&t=369s