ScopeTreasure Courses

GIT & GITHUB COURSES · LESSON G.9 · CODE SCOPE SERIES

GitHub Actions — Auto Kaam Ka Taaruf

A robot on every push: what CI is, your first workflow YAML file, reading the green check and the red X, and what teams automate next.

Introduction

Every professional team asks the same question on every change: "does it still work?" — and no human wants to answer it by hand, fifty pushes a day. **GitHub Actions** is GitHub's built-in robot: you write a small **workflow** file, and on every push GitHub starts a fresh machine, runs your steps (install, test, build), and stamps the commit with a **green check** or a **red X**. That stamp is called **CI — Continuous Integration** — and this lesson writes your first workflow and reads its results like a professional. *(Roman Urdu: GitHub Actions aik robot hai — har push par khud jaagta hai, aap ke steps chalata hai, aur pass ya fail ki khabar commit par laga deta hai.)*

Key points

- **A workflow is a YAML file in `.github/workflows/`:** the folder and file live IN your repo, so the automation is versioned with the code. Any `.yml` file there is a live workflow.
- **Three parts: event → jobs → steps.** `on:` names the trigger (here: every `push`). A **job** runs on a fresh virtual machine (`runs-on: ubuntu-latest`). **Steps** are commands (`run:`) or ready-made actions (`uses:`) executed top to bottom.
  (Roman Urdu: Waqia (push) → kaam (job) → chhoti chhoti hidayat (steps) — bas yehi poora khaka hai.)
- **`actions/checkout` is almost always step one:** it downloads YOUR repo onto the robot's machine. Without it, the machine starts empty and has nothing to test.
- **The result decorates the commit and the PR:** green check = every step passed; red X = something failed, and the run's log shows the exact step and error. Teams set repos to refuse merging while the check is red — that is a **required status check**.
- **Minutes are the currency:** public repos get Actions free; private repos have a monthly free allowance. One more reason steps stay lean.

Code examples

Example 1 — Your first workflow file

```
name: Pehla CI

on: [push]

jobs:
  salaam:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Robot ka salaam
        run: echo "Assalam o Alaikum — main CI robot hoon, push mil gaya!"
      - name: Repo ki files ginna
        run: ls -la
```

What it does: saved as `.github/workflows/ci.yml` and pushed, this workflow wakes on every push. `checkout` brings your code onto a fresh Ubuntu machine, the first `run` prints its greeting into the log, and `ls -la` lists your repo's files — proof the machine really holds your project.

What you will see: within seconds of the push, the repo's **Actions tab** shows a run named "Pehla CI" with a spinning yellow dot that turns into a green check. Open the run → open the "salaam" job → every step unfolds, and the greeting plus the file listing are printed in the log, line by line.

Example 2 — A workflow that actually guards a project: run the tests

```
name: Tests

on:
  push:
    branches: [main]
  pull_request:

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm install
      - run: npm test
```

What it does: this is the shape real projects use. It triggers on pushes to `main` AND on every pull request — so a PR shows its verdict before anyone merges. `setup-node` installs Node 20 on the robot, `npm install` builds the project's dependencies, and `npm test` runs the suite; if one test fails, that step turns red and the whole run fails.

What you will see: on the pull request page, a checks box appears: "Tests / test — Successful in 42s" with a green tick, or a red X with a **Details** link that opens the log exactly at the failing test. That box is what reviewers look at before reading a single line of your diff (G.5).

Example 3 — Reading a failure in the run log

```
Run npm test
  npm test
  ...
  FAIL  tests/notes.test.js
  ● Notes › deleting a note removes it from the list

    expect(received).toBe(expected) // Object.is equality

    Expected: 2
    Error: Process completed with exit code 1.
```

What it does: this is the tail of a FAILED run's log (from the `npm test` step). It names the suite (`tests/notes.test.js`), the exact test ("deleting a note removes it from the list"), and what the test expected versus received, before GitHub stamps the run failed. Your job as the developer: reproduce it locally, fix, push — the same PR's checks simply re-run and turn green.

What you will see: notice there is no mystery and no blame — the robot reports WHERE it stopped and WHAT the failing command printed. Ninety percent of CI debugging is reading the first red line in the log, not the last.

Quick recap

- Workflow = YAML in `.github/workflows/`: `on:` trigger → `jobs` on fresh machines → `steps` (`uses` an action, or `run` a command). `checkout` first, almost always.
- Results land as a green check / red X on the commit and inside the PR — CI is the automated half of code review.
- A failing run is a gift with a log attached: read the first red step, fix locally, push again.

Common mistakes

1. **YAML spacing errors** — YAML is indentation-strict and tabs are forbidden; a workflow that "does nothing" is usually a file GitHub rejected silently. The fix: spaces only, keep the structure copied from a working example, and check the Actions tab — GitHub lists invalid workflows with their error.

2. **Forgetting `actions/checkout`** — the run fails at your first real command because the machine is empty. The fix: checkout is step one in practically every workflow; make it a reflex.
3. **Treating the red X as the robot's fault** — beginners re-run until green ("shayad theek ho jaye"). The fix: the same code fails the same way; read the log, reproduce locally, fix the code — re-running is only for proving the fix.

Practice

1. Add Example 1's workflow to one of your GitHub repos (path must be exactly `.github/workflows/ci.yml`), push, and watch the run appear in the Actions tab. Open the log and find your greeting line.
2. Break it on purpose: add a step `run: exit 1` after the greeting, push, and read the failed run — which step is red, and what does the log say? Then remove the step, push, and watch the same workflow turn green.

Practice quiz

1. Where does a GitHub Actions workflow live, and what are its three parts?

- On your computer only, in any folder
- A YAML file in .github/workflows/ inside your repo, so it is versioned with the code; its parts are event, then jobs, then steps
- On GitHub's settings page only
- In a .yml file anywhere, with no structure

Answer: A YAML file in .github/workflows/ inside your repo, so it is versioned with the code; its parts are event, then jobs, then steps

2. Why is actions/checkout almost always step one of a workflow?

- It checks your code for errors
- It downloads your repo onto the robot's fresh machine; without it the machine starts empty and has nothing to test
- It publishes the repo
- It is required by law

Answer: It downloads your repo onto the robot's fresh machine; without it the machine starts empty and has nothing to test

3. What does a red X on a commit or PR mean in Actions?

- The robot is broken, re-run until green
- Something failed, and the run's log shows the exact step and error; read the first red step, fix locally, push again
- The commit was deleted

- Nothing, colours are decoration

Answer: Something failed, and the run's log shows the exact step and error; read the first red step, fix locally, push again

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/github-actions-intro · More free tools and notes at https://scopetreasure.com