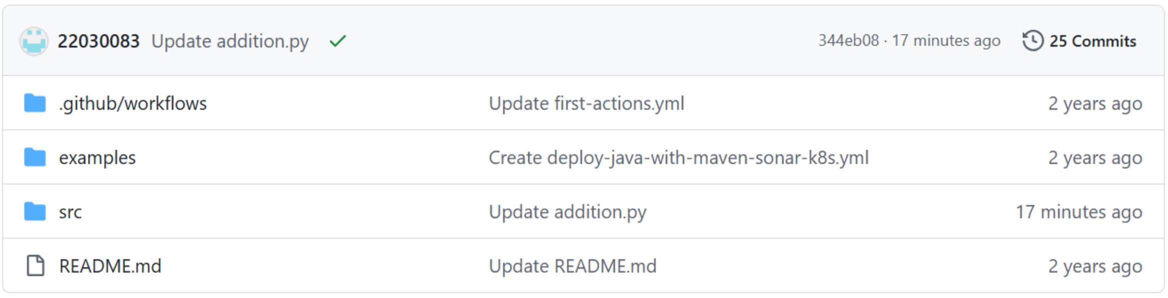
22030083 Update addition.py ✓ 344eb08 · 17 minutes ago 25 Commits

.github/workflows	Update first-actions.yml	2 years ago
examples	Create deploy-java-with-maven-sonar-k8s.yml	2 years ago
src	Update addition.py	17 minutes ago
README.md	Update README.md	2 years ago

🔹 **GitHub Actions Basics**

- **Workflow folder**:
  .github/workflows/first-actions.yml
  → This file defines your CI/CD pipeline (like Jenkinsfile in Jenkins).
- Example workflow (your code refined):

name: My First GitHub Actionson: [push] # Triggers when you push to repojobs: build: runs-on: ubuntu-latest # Runner provided by GitHub strategy: matrix: python-version: [3.8, 3.9] # Runs for multiple versions steps: - uses: actions/checkout@v3 # Checkout repo code - name: Set up Python uses: actions/setup-python@v2 # Official GitHub action with: python-version: ${{ matrix.python-version }} - name: Install dependencies run: | python -m pip install --upgrade pip pip install pytest - name: Run tests run: | cd src python -m pytest addition.py

✅ Every time you push code → this workflow runs.
✅ You can create **multiple workflow files** inside .github/workflows/.

🔹 **Jenkins vs GitHub Actions**

Feature	**Jenkins**	**GitHub Actions**
Where it runs?	Needs your own server (VM, Docker, on-prem).	Runs on GitHub's hosted runners by default.
Who manages server?	You/team (install, update, plugins, scaling).	GitHub manages runners (auto-scale, auto-update).
File used	Jenkinsfile	.yml inside .github/workflows/
Plugins	Must install manually.	Most integrations are built-in as actions.
Secrets	Stored in Jenkins credentials store.	Stored in GitHub Settings → Secrets & Variables.

Feature	Jenkins	GitHub Actions
Cost	Free (self-hosted).	Free for public repos; limits for private repos.

👉 **Analogy**:

- Jenkins = Your own **kitchen** (you set up stove, ingredients).
- GitHub Actions = **Restaurant** (they give you kitchen & staff).

🔹 setup-python@v2 explained

- This is an **official GitHub Action** maintained by GitHub.
- Purpose: Install and configure specific versions of Python.
- with: python-version → lets you run tests on multiple Python versions.
- Without it, runner only has a default Python (not your required version).

🔹 Secrets Management in DevOps

Every CI/CD tool has a way to store **credentials securely**:

- **Jenkins** → Credentials Store (encrypted in Jenkins server).
- **GitHub Actions** → Settings → Secrets and Variables → Actions.
- **GitLab CI/CD** → Variables in Project Settings.
- **Docker** → docker secrets.
- **Kubernetes** → kubeconfig secrets in kubectl or via SealedSecrets.
- **Vault** → HashiCorp Vault (popular standalone tool).

🔹 Migrating CI/CD (GitHub → GitLab → Jenkins)

- Not “plug & play.” You must **rewrite pipeline files** because:
    - GitHub uses .yml workflows.
    - Jenkins uses Jenkinsfile (Groovy).
    - GitLab uses .gitlab-ci.yml.
- But **concepts are same**: stages, jobs, runners, secrets.

🔹 **GitHub Hosted vs Self-hosted Runner**

1. GitHub Hosted Runner

- runs-on: ubuntu-latest → GitHub provides a temporary VM.
- Good for **public projects** and small private projects.
- Limitation: You don't control where your code runs.
  ❌ Not suitable for **sensitive projects** (like banking).
  ❌ Limited CPU/RAM.

2. Self-hosted Runner

- You bring your own machine (EC2, on-prem server, VM, Docker).
- Steps:
  1. Launch EC2.
  2. Install GitHub runner agent.
  3. Register runner with repo/org (Settings → Actions → Runners).
  4. Change workflow:

runs-on: self-hosted

- All jobs now run **on your EC2** instead of GitHub servers.

👉 Why use Self-hosted?

1. Private company, sensitive data.
2. Heavy compute (AI/ML, banking apps).
3. Need control over environment (custom software, firewalls).

🔹 **Jenkins Agent Communication**

- Jenkins **master** (controller) and **agents** (workers).
- Communicate via **SSH protocol**.
- Configure inbound/outbound rules: open **HTTP (8080) & HTTPS (443)**.

🔹 **Final Rule of Thumb**

- **Public project** → GitHub Actions (free, secure).
- **Private, sensitive project** → Jenkins or self-hosted GitHub runner.
- **Secrets** → Always use built-in secret managers (never hardcode).

👉 So if someone asks you:
"Where are you running your CI/CD pipeline?"
You should answer:

- If public repo → "On GitHub hosted runners (ubuntu-latest)."
- If private/sensitive project → "On self-hosted runners (EC2) or Jenkins agents."

🔹 **1. GitHub Hosted Runner (default, simple case)**

👉 **Runs on GitHub's own server (ubuntu-latest, windows-latest, macos-latest).**

Steps:

1. Create a repo on GitHub.
2. Inside repo → create folder:
   .github/workflows/
3. Add workflow file → ci.yml:

name: CI with GitHub Hosted Runneron: [push]jobs: build: runs-on: ubuntu-latest # Hosted runner (GitHub provides it) steps: - uses: actions/checkout@v3 # Checkout repo code - name: Run a sample script run: | echo "Hello from GitHub hosted runner!" uname -a

4. Push code →
   - Go to **GitHub Repo → Actions tab**
   - You'll see your workflow executed on a GitHub-provided VM.
   - Logs will show **Ubuntu environment details**.

✅ That's it. Nothing else to install. GitHub manages everything.

🔹 **2. Self-hosted Runner (your own EC2, VM, or machine)**

👉 Here, you run jobs on **your own machine** instead of GitHub's.

Step 1: Launch a Server

- Example: AWS EC2 Ubuntu instance (t2.micro for test, bigger if heavy workloads).
- Allow inbound rules: **SSH (22)**, **HTTP (80)**, **HTTPS (443)**.

Step 2: Register a Self-hosted Runner

7. Go to your GitHub Repo → Settings → Actions → Runners → New self-hosted runner.
8. Select **OS = Linux**. GitHub will show you commands like:

# 1. Download runner packagecurl -o actions-runner-linux-x64-2.317.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.317.0/actions-runner-linux-x64-2.317.0.tar.gz# 2. Extract ittar xzf ./actions-runner-linux-x64-2.317.0.tar.gz# 3. Create folder and move insidemkdir actions-runner && cd actions-runner# 4. Configure runner (GitHub gives you a token)./config.sh --url https://github.com/USERNAME/REPO --token <TOKEN>

3. Start the runner:

./run.sh

👉 Now your EC2 is linked with GitHub repo as a **self-hosted runner**.

Step 3: Use Self-hosted Runner in Workflow

Create .github/workflows/ci.yml:

name: CI with Self-hosted Runneron: [push]jobs: build: runs-on: self-hosted # Runs on your EC2 steps: - uses: actions/checkout@v3 - name: Run on EC2 run: | echo "Hello from my self-hosted runner (EC2)!" hostname lsb_release -a

---

Step 4: Run & Verify

4. Push code to GitHub.
5. Workflow triggers.
6. Go to **Actions tab** → Logs → You'll see **your EC2 machine name** in output.

Step 5 (Optional): Run as Service

To make sure runner restarts after EC2 reboot:

sudo ./svc.sh installsudo ./svc.sh start

🔹 **Difference in Practice**

- **GitHub Hosted Runner** → No setup, just runs-on: ubuntu-latest.
- **Self-hosted Runner** → You configure your own server, register it, and use runs-on: self-hosted.

👉 Summary:

- Use **GitHub Hosted** for **public/open-source projects** (free, auto-managed).
- Use **Self-hosted** for **private/sensitive projects** or heavy compute workloads.