CI/CD tool. GitHub Actions.

Build Automation vs. Deployment Automation

- Build Automation: Focuses on compiling source code, running initial tests (like unit tests), and packaging the application into deployable artifacts (e.g., JAR files, Docker images).
- Deployment Automation: Takes those build artifacts and distributes them to target environments, configuring the necessary infrastructure and services

Resource: https://www.netdata.cloud/academy/deployment-automation/

Benefits of Deployment Automation

- Faster release cycles
- Reduced human errors
- Better reliability & consistency
- Improved efficiency and collaboration
- Easier rollbacks

Resource: https://www.netdata.cloud/academy/deployment-automation/

Automated Deployment Process

1. Commit to version control
2. Automated build & testing
3. Artifact stored in repository
4. Deploy to staging
5. Optional approval
6. Deploy to production
7. Monitoring & feedback

Resource: https://www.netdata.cloud/academy/deployment-automation/

Tools Used in Deployment Automation

Version Control: GitHub, GitLab, Bitbucket

CI/CD Platforms:

- Jenkins (Highly extensible, open-source)
- GitLab CI/CD (Integrated with GitLab repositories)
- GitHub Actions (Integrated with GitHub repositories)
- Bitbucket Pipelines (Integrated with Bitbucket repositories)
- CircleCI, Travis CI, Azure DevOps Pipelines, AWS CodePipeline

Deployment Tools:

- Ansible (Agentless, uses YAML)
- Chef (Agent-based, uses Ruby)
- Puppet (Agent-based, uses its own declarative language)
- SaltStack (Agent-based, uses Python)

Resource: https://www.netdata.cloud/academy/deployment-automation/

Tools Used in Deployment Automation

Containerization & Orchestration

- Docker (Creating container images)
- Kubernetes (Orchestrating container deployments, scaling, and management)

Artifact Repositories

- JFrog Artifactory
- Sonatype Nexus Repository
- Docker Hub / other container registries

Infrastructure as Code (IaC) Tools

- Terraform
- AWS CloudFormation
- Azure Resource Manager (ARM) Templates
- Pulumi

Automated Testing Tools

- Validate code at different stages (JUnit, NUnit, pytest, Selenium, Cypress, Postman, etc.)

Resource: https://www.netdata.cloud/academy/deployment-automation/

Best Practices

- Start small
- Version control everything
- Automate Testing Extensively
- Standardize environments
- Implement Automated Rollbacks
- Monitor aggressively
- Treat Infrastructure as Code (IaC)
- Secure The Pipeline
- Foster Collaboration
- Document The Process

Resource: https://www.netdata.cloud/academy/deployment-automation/

Metrics for Success

- Deployment Frequency
- Lead Time for Changes
- Change Failure Rate
- Mean Time to Recovery (MTTR)

Resource: https://www.netdata.cloud/academy/deployment-automation/

Common Challenges

- Lack of automated testing
- Toolchain complexity
- Environment differences
- Database migration issues
- Legacy systems
- Cultural Resistance
- Security
- Deployment automation

History of Continuous Integration Tools

Tinderbox

Hudson

Bamboo
(Atlassian)

Jenkins
(fork of Hudson)

GitLab CI/CD

GitHub Actions

2000

2005

2010

2015

2020

CruiseControl

TeamCity
(JetBrains)

CircleCI

Travis CI

AWS CodeBuild

GITHUB ACTIONS

+19%

ARGO

+16%

JENKINS

+40%

GITLAB

+20%

AZURE PIPELINES

+3%

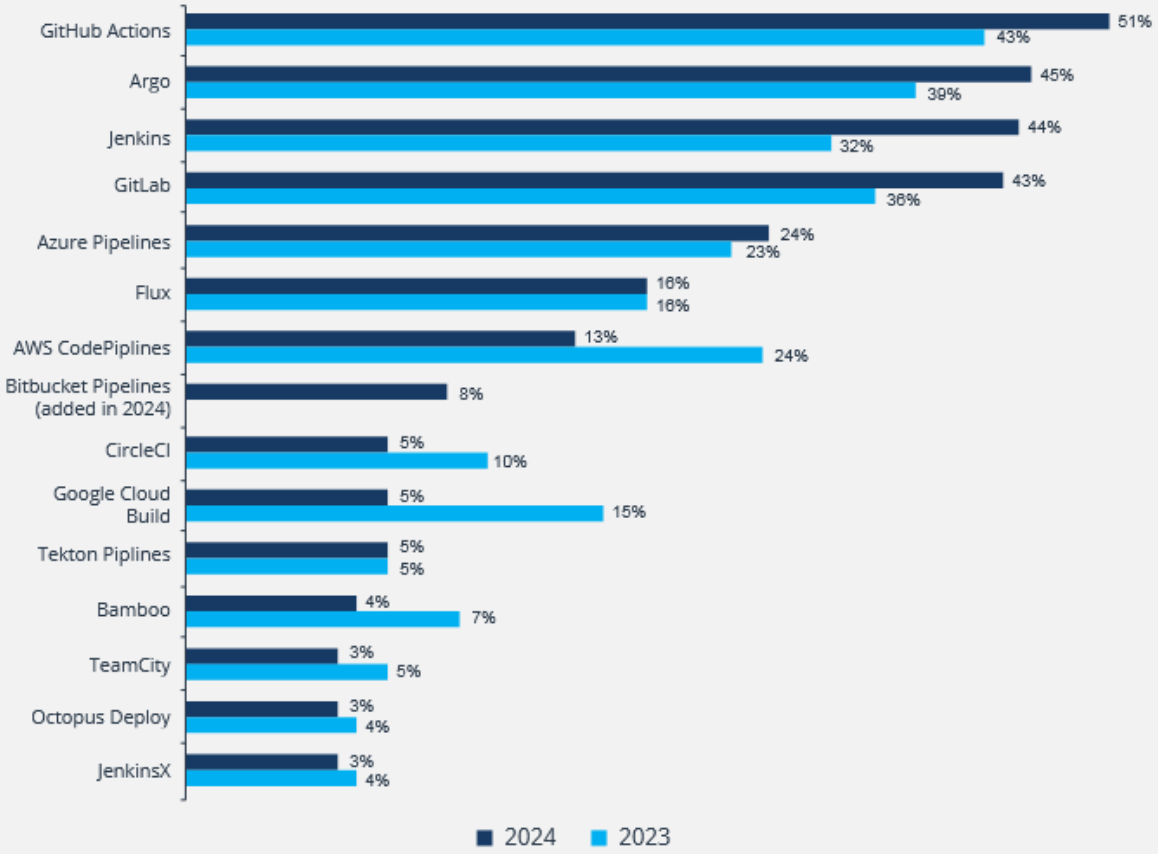
FLUX

+3%

FIGURE 25

TOOLS IN USE TO MANAGE CI/CD PIPELINES

What tools does your organization currently use to manage its CI/CD pipeline? (select all that apply) (top 15 products/projects shown)



2024 CNCF Annual Survey Q57 sample size = 596, Valid Cases = 596, Total mentions = 1774, shown to those who are using or testing Ci/CD tools in Q56
2023 CNCF Annual Survey, q95, sample size = 819, valid cases = 819, total mentions = 2,786

https://www.cncf.io/wp-content/uploads/2025/04/cncf_annual_survey24_031225a.pdf

What is GitHub Actions?

- GitHub Actions is a CI/CD tool for the GitHub flow.
- GitHub Actions brings automation directly into the software development lifecycle on GitHub via event-driven triggers.
- GitHub Actions is free and available for use on any public repository and self-hosted runner.
- Coding language: C, C++, C#, Java, JavaScript, PHP, Python, Ruby, Scala, and TypeScript.

The most common use cases for GitHub Actions

- Build, test, and deploy within the GitHub flow
- Automate repetitive tasks
- Manage users easily at scale
- Easily add preferred tools and services to your project
- Keep track of your projects

Workflows

- A workflow is a configurable automated process that will run one or more jobs.
- Workflows are defined in the .github/workflows directory in a repository.
- A repository can have multiple workflows, each of which can perform a different set of tasks such as:
  - Building and testing pull requests
  - Deploying your application every time a release is created
  - Adding a label whenever a new issue is opened

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

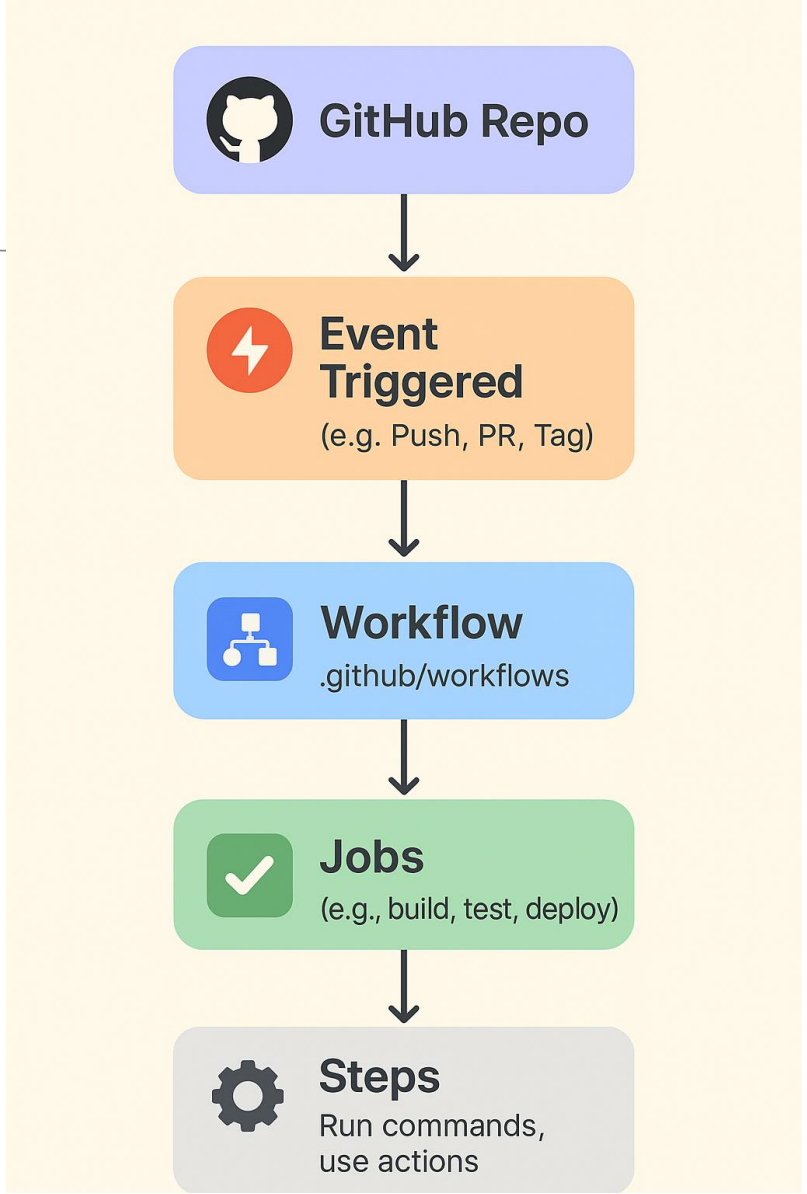
Common Workflow Types

- Validate
  - Lint
  - Test
  - Static analysis
- Build
  - Executable
  - Container Image
- Deploy
  - Push based deploys
  - Update tags for git-ops deploys

- Repo Automations
  - Release automations (e.g., release-please)
  - Stale Issues/PRs
  - Dependency upgrades (e.g., renovate)

Core concepts

- **Events**: Events are defined triggers that kick off a workflow.
- **Jobs**: Jobs are a set of steps that execute on the same runner.
- **Steps**: Steps are individual tasks that run commands in a job.
- **Actions**: An action is a command that's executed on a runner.
- **Runners**: A runner is a GitHub Actions server.



Examples of triggers/events

- A new commit is pushed to a branch
- Someone opens/edits a pull request
- Someone closes an issue
- A new release is made
- Another GitHub Action finishes running
- Manual triggering

Examples of tasks

- Run regression tests
- Calculate code coverage
- Profile performance
- Update a website
- Prepare a new release
- Build the manual
- Check code formatting

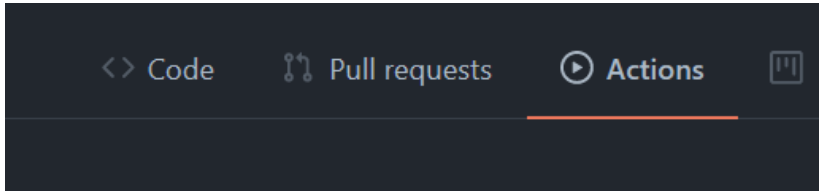
Example workflows

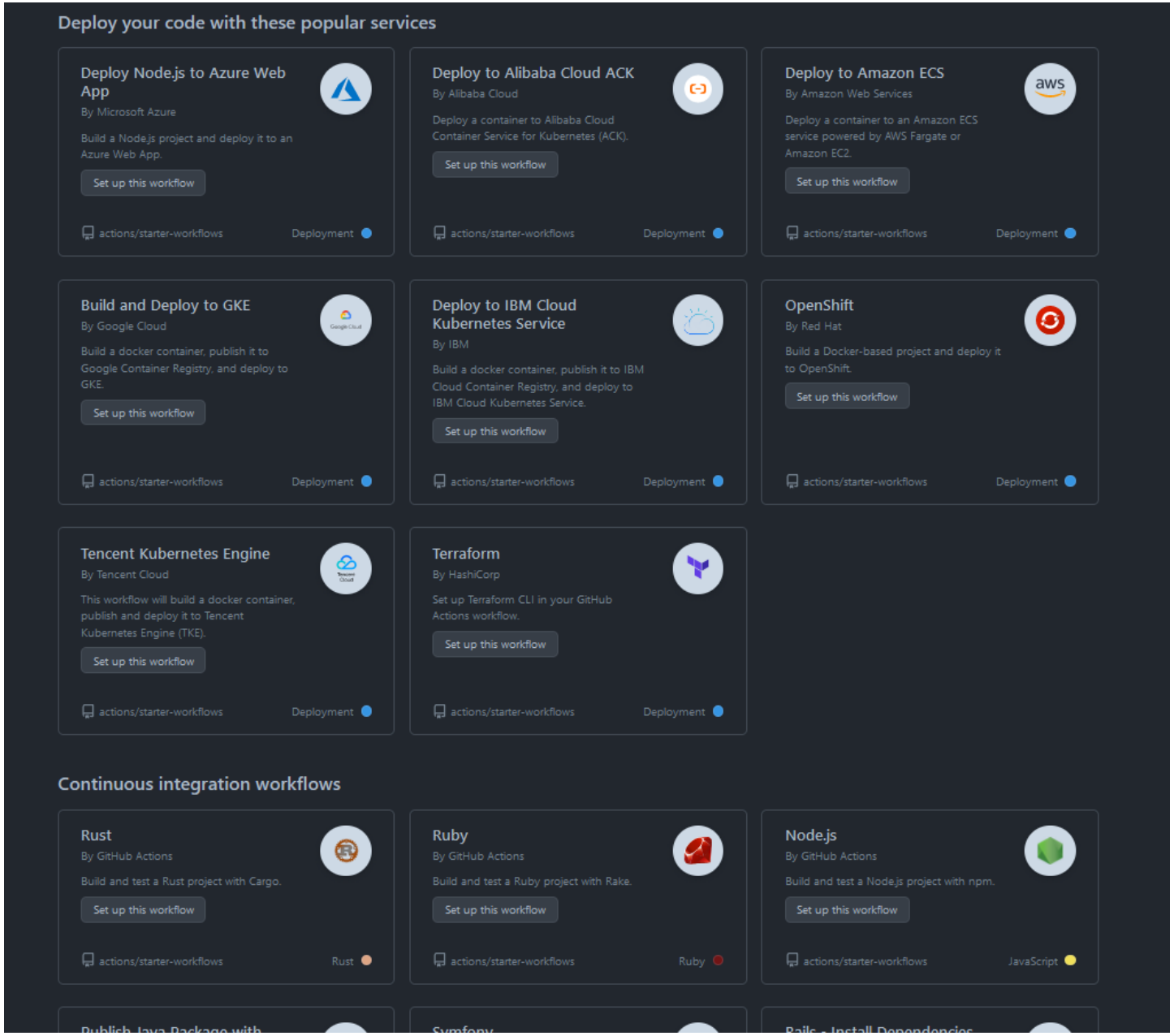
A “workflow” is a unit consisting of configurations and tasks.

Examples:

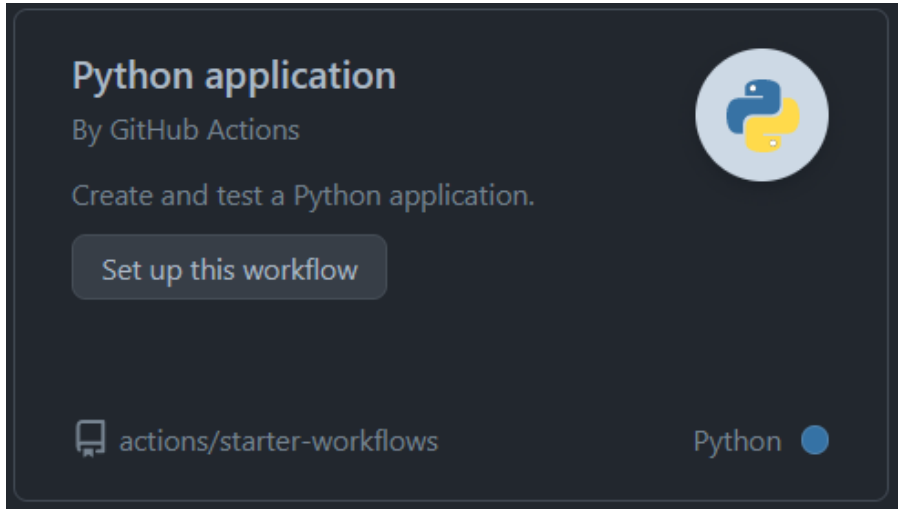
- When a pull request is made: run the regression tests, and see if new code has test coverage.
- Every morning at 8am, prepare a release of your software.
- Every time there’s a new release, download it, check that it works, and update a website.

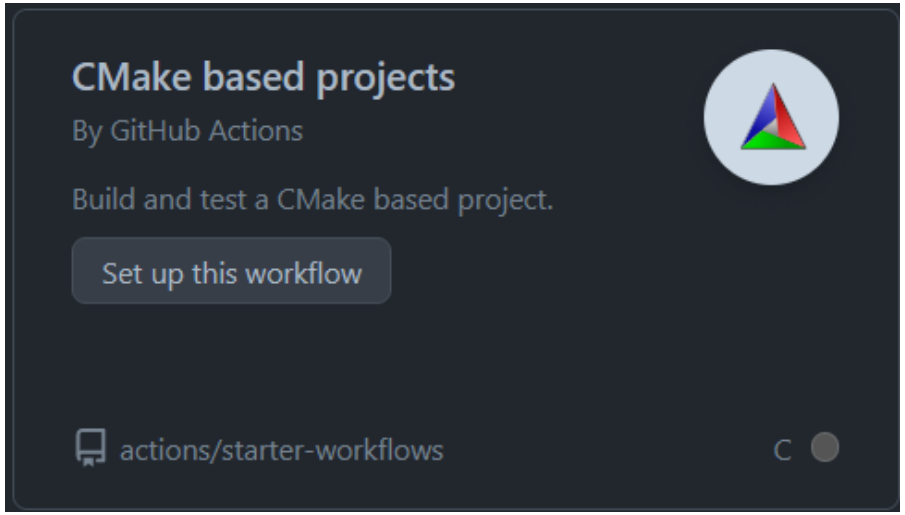
Example

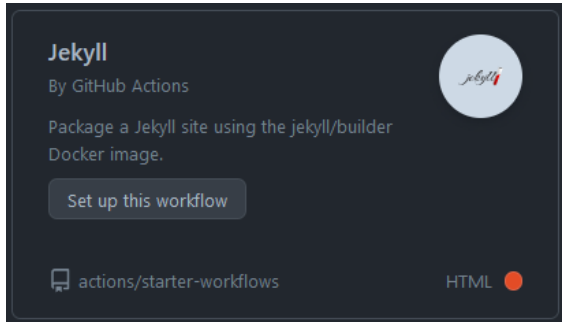

Code
Pull requests
Actions

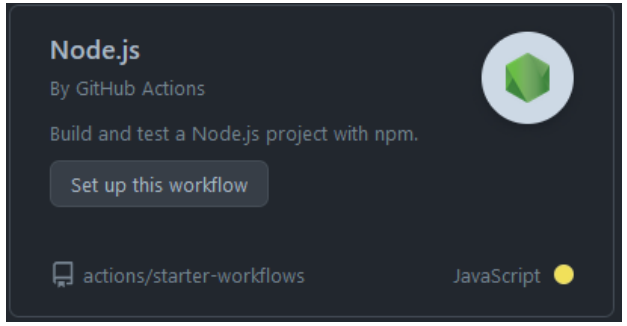

Deploy your code with these popular services
Deploy Node.js to Azure Web App
By Microsoft Azure
Build a Node.js project and deploy it to an Azure Web App.
Set up this workflow
actions/starter-workflows
Deployment
Deploy to Alibaba Cloud ACK
By Alibaba Cloud
Deploy a container to Alibaba Cloud Container Service for Kubernetes (ACK).
Set up this workflow
actions/starter-workflows
Deployment
Deploy to Amazon ECS
By Amazon Web Services
Deploy a container to an Amazon ECS service powered by AWS Fargate or Amazon EC2.
Set up this workflow
actions/starter-workflows
Deployment
Build and Deploy to GKE
By Google Cloud
Build a docker container, publish it to Google Container Registry, and deploy to GKE.
Set up this workflow
actions/starter-workflows
Deployment
Deploy to IBM Cloud Kubernetes Service
By IBM
Build a docker container, publish it to IBM Cloud Container Registry, and deploy to IBM Cloud Kubernetes Service.
Set up this workflow
actions/starter-workflows
Deployment
OpenShift
By Red Hat
Build a Docker-based project and deploy it to OpenShift.
Set up this workflow
actions/starter-workflows
Deployment
Tencent Kubernetes Engine
By Tencent Cloud
This workflow will build a docker container, publish and deploy it to Tencent Kubernetes Engine (TKE).
Set up this workflow
actions/starter-workflows
Deployment
Terraform
By HashiCorp
Set up Terraform CLI in your GitHub Actions workflow.
Set up this workflow
actions/starter-workflows
Deployment
Continuous integration workflows
Rust
By GitHub Actions
Build and test a Rust project with Cargo.
Set up this workflow
actions/starter-workflows
Rust
Ruby
By GitHub Actions
Build and test a Ruby project with Rake.
Set up this workflow
actions/starter-workflows
Ruby
Node.js
By GitHub Actions
Build and test a Node.js project with npm.
Set up this workflow
actions/starter-workflows
JavaScript

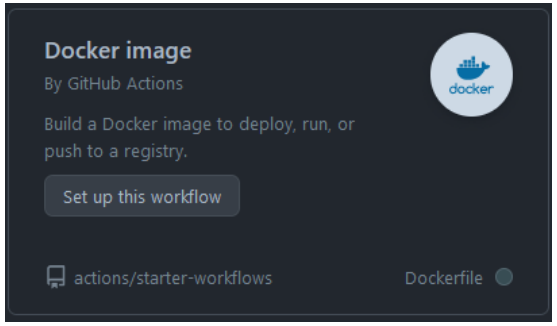
CI Workflow Examples



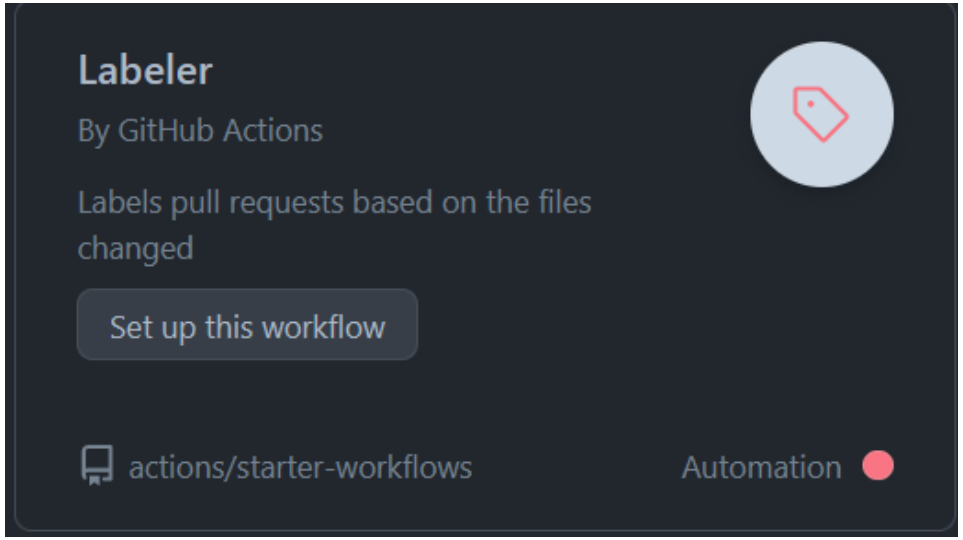


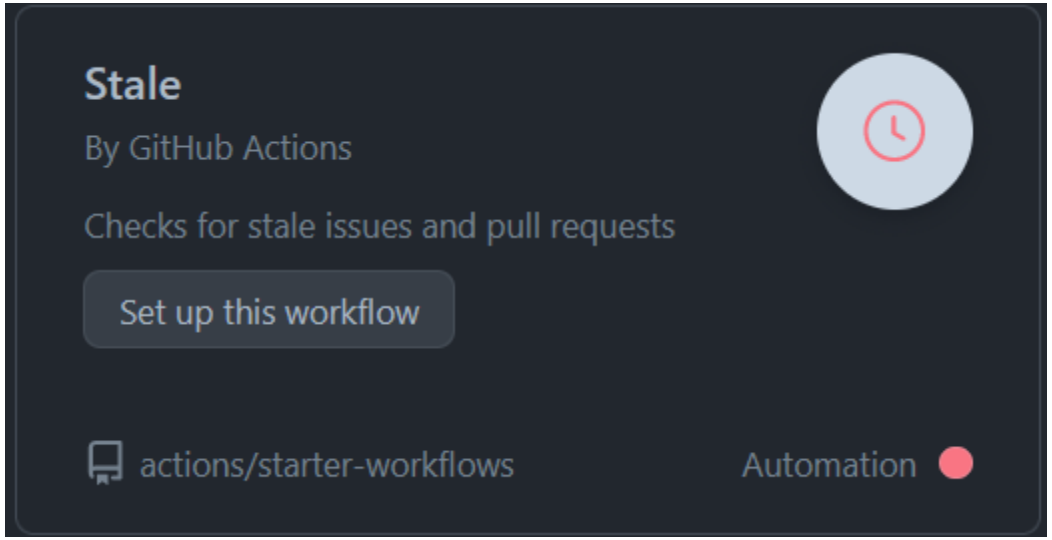






Github Task Automation examples





. . . or anything else!

How to add an workflow

Insert a **.yml file** at {repository-name}/.github/workflows/

Define the following

- Name
- Workflow trigger
- What jobs to run

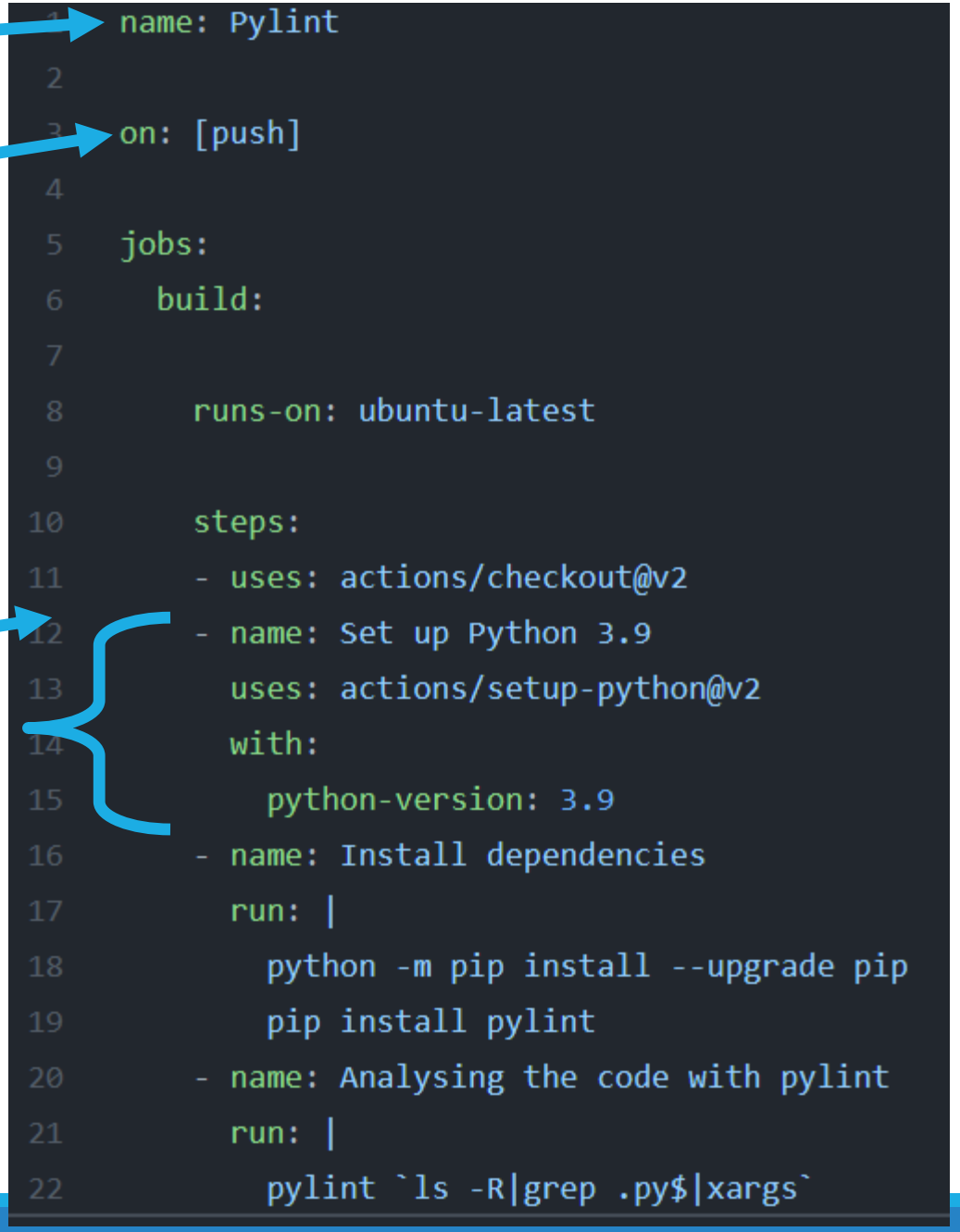
Name

Trigger

“Workflow file”

Jobs

“Action”



```yaml
name: Pylint

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.9
      uses: actions/setup-python@v2
      with:
        python-version: 3.9
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install pylint
    - name: Analysing the code with pylint
      run: |
        pylint `ls -R|grep .py$|xargs`
```

Specifying the Java version and architecture

YAML

```yaml
steps:
  - uses: actions/checkout@v5
  - name: Set up JDK 11 for x64
    uses: actions/setup-java@v4
    with:
      java-version: '11'
      distribution: 'temurin'
      architecture: x64
```

Building and testing your code

YAML

```yaml
steps:
  - uses: actions/checkout@v5
  - uses: actions/setup-java@v4
    with:
      java-version: '17'
      distribution: 'temurin'
  - name: Run the Maven verify phase
    run: mvn --batch-mode --update-snapshots verify
```

Caching dependencies

YAML

```yaml
steps:
  - uses: actions/checkout@v5
  - name: Set up JDK 17
    uses: actions/setup-java@v4
    with:
      java-version: '17'
      distribution: 'temurin'
      cache: maven
  - name: Build with Maven
    run: mvn --batch-mode --update-snapshots verify
```

Packaging workflow data as artifacts

YAML

```yaml
steps:
  - uses: actions/checkout@v5
  - uses: actions/setup-java@v4
    with:
      java-version: '17'
      distribution: 'temurin'
  - run: mvn --batch-mode --update-snapshots verify
  - run: mkdir staging && cp target/*.jar staging
  - uses: actions/upload-artifact@v4
    with:
      name: Package
      path: staging
```