# GitHub Actions

Make them work for you!

*26 September 2024*

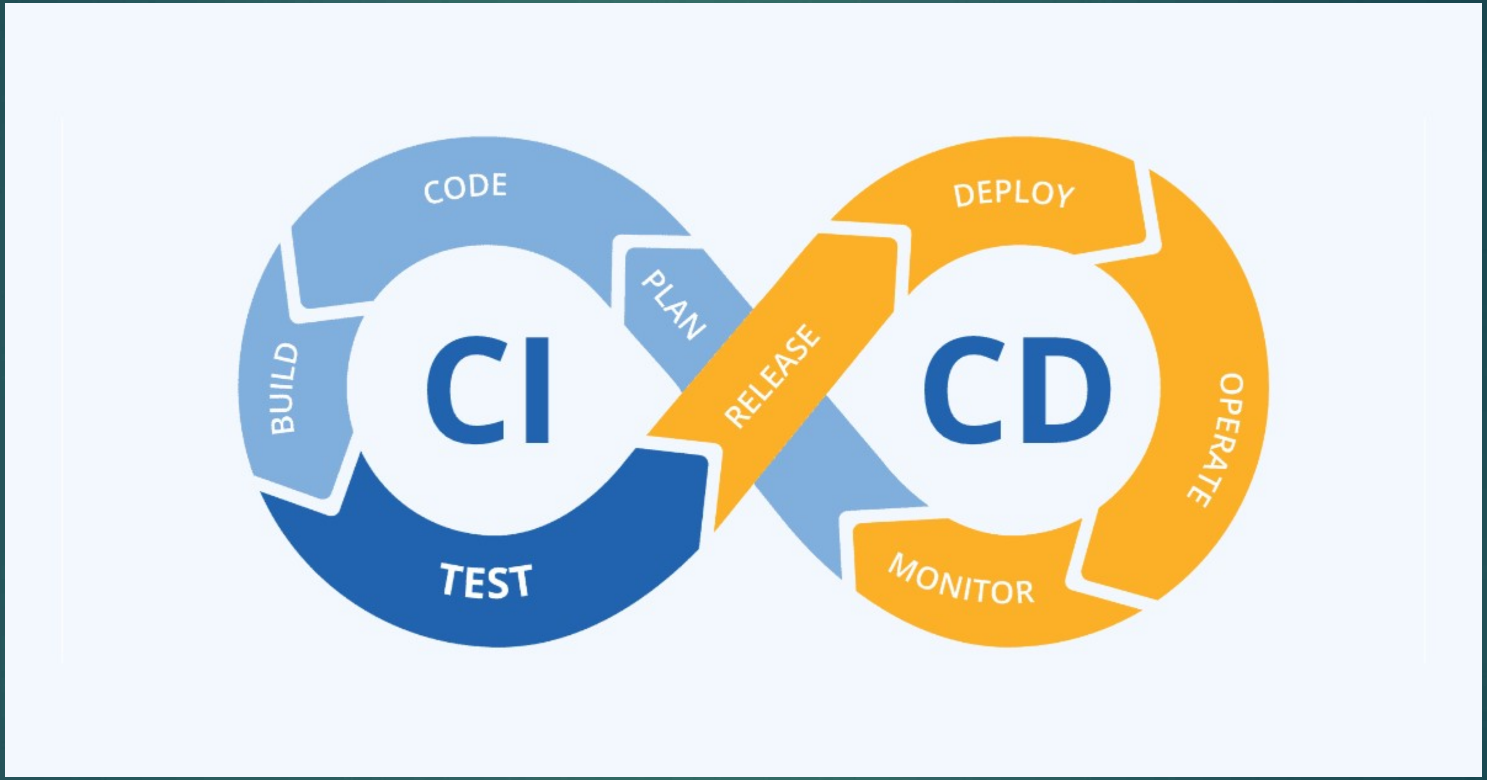
*Eleftherios Chrysochoidis*

## Agenda

- Introduction to CI/CD
- GitHub Actions – Overview
- GitHub Actions – Basic Use Cases
- GitHub Actions – Advanced Use Cases
- Resources & QA

## Introduction to CI/CD

## Introduction to CI/CD



**Continuous integration (CI)** refers to the practice of automatically and frequently integrating code changes into a shared source code repository

**Continuous deployment (CD)** refers to the integration, testing, and delivery of code changes.

## Benefits of CI/CD

- Developer focus on code
- Enhance code quality via testing
- Deploy faster
- Reduce costs of delivery

# CI/CD Platforms

GitHub Actions

Azure Pipelines

Travis CI

GitLab CI

Jenkins

JenkinsX

Circle CI

Argo CD

## GitHub Actions – Overview

## Pricing

- 2,000 CI/CD minutes per month – for private repositories
  - Free for public repositories!
- 500 MB of Packages storage – for private repositories
  - Unlimited for public repositories!

## Prerequisites

- Basic Linux commands and knowledge
  - **echo**, **cp**, **mv**, **cat**, **chmod, ls, grep, touch, sed**
  - Environmental variables
    - e.g. **export VAR_NAME=value** and use like: **echo $VAR_NAME**
  - Pipes (|), Appenders (>, >>)
  - Output types and redirection (STDOUT, STDERR, 2>&1)
- Yaml syntax/structure
- GitHub API (for more advanced usage)

## Core Concepts

- **Workflows**: Automated process that runs one or more jobs. It is defined in a YAML file located in the **.github/workflows/** directory.
- **Jobs**: Set of steps that are executed on the same runner.
- **Steps**: Individual tasks in a job. Can be shell commands or actions.
- **Actions**: Individual commands or scripts that can be reused across workflows.
- **Runners**: A server that runs the workflows. GitHub provides both Linux, Windows, and macOS runners (self-hosted is also supported)

## GitHub Events

Events that trigger workflows.

Configure workflows to run when specific activity on GitHub happens, or at a scheduled time.

- push
- pull_request
- workflow_dispatch
- release
- schedule

More: https://docs.github.com/en/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows

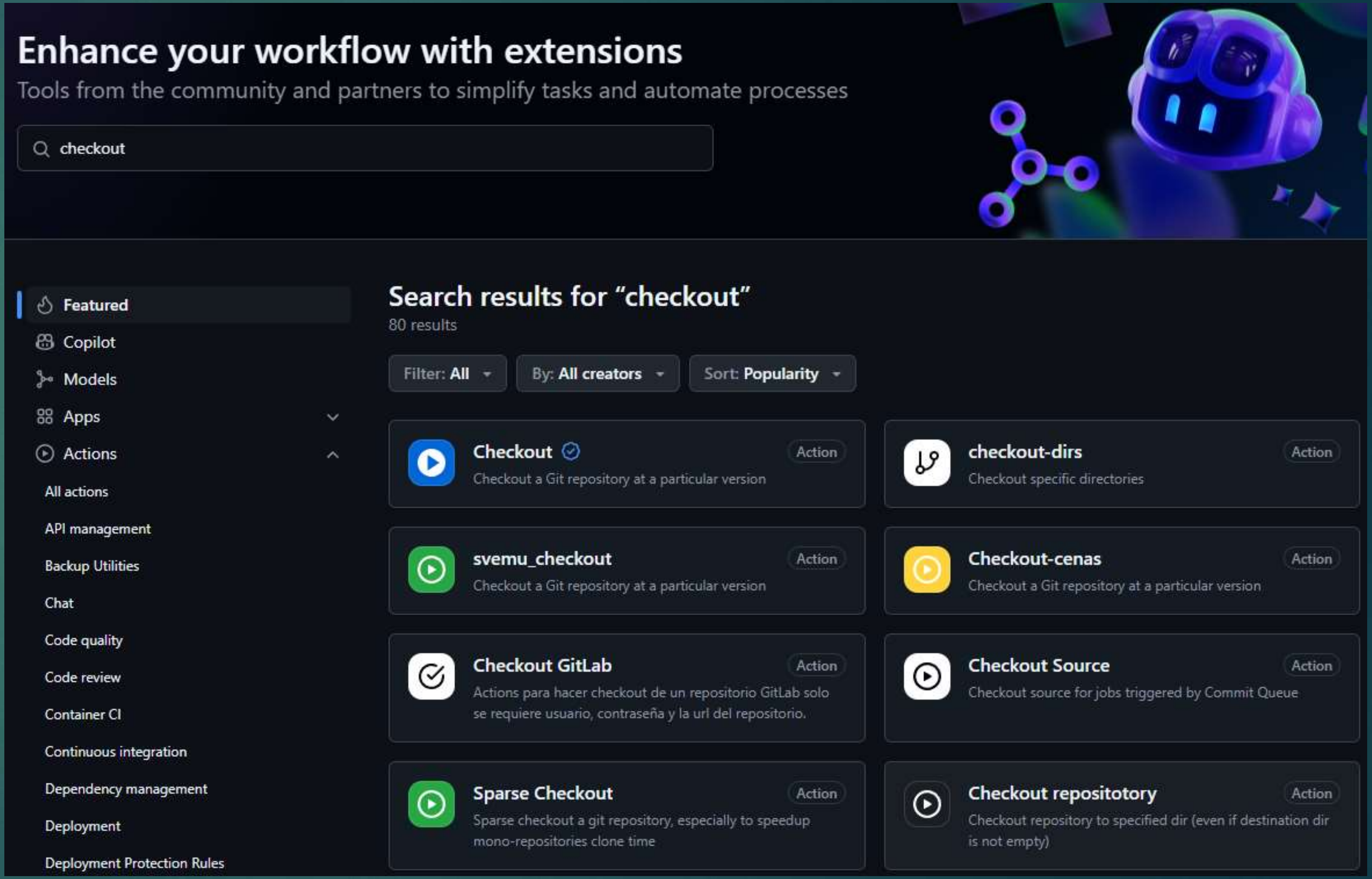
# Example Workflow

```yaml
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{ github.event_name }} event."
      - run: echo "🐧 This job is now running on a ${{ runner.os }} server hosted by GitHub!"
      - run: echo "🔎 Name of branch is ${{ github.ref }} and repo is ${{ github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v4
      - run: echo "💡 The ${{ github.repository }} repository has been cloned to the runner."
      - run: echo "🖥️ The workflow is now ready to test your code on the runner."
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🍏 This job's status is ${{ job.status }}."
```

## Marketplace

https://github.com/marketplace

- Tools from the community and partners to simplify tasks and automate processes
- Enhance your workflow with extensions



Enhance your workflow with extensions

Tools from the community and partners to simplify tasks and automate processes

checkout

Search results for "checkout"

80 results

Filter: All · By: All creators · Sort: Popularity

| | |
|---|---|
| **Checkout** (Action) — Checkout a Git repository at a particular version | **checkout-dirs** (Action) — Checkout specific directories |
| **svemu_checkout** (Action) — Checkout a Git repository at a particular version | **Checkout-cenas** (Action) — Checkout a Git repository at a particular version |
| **Checkout GitLab** (Action) — Actions para hacer checkout de un repositorio GitLab solo se requiere usuario, contraseña y la url del repositorio. | **Checkout Source** (Action) — Checkout source for jobs triggered by Commit Queue |
| **Sparse Checkout** (Action) — Sparse checkout a git repository, especially to speedup mono-repositories clone time | **Checkout repositotory** (Action) — Checkout repository to specified dir (even if destination dir is not empty) |

## GitHub Actions – Basic Use Cases

## Simple Workflow

https://docs.github.com/en/actions/writing-workflows/quickstart

```yaml
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{ github.event_name }} event."
      - run: echo "🐧 This job is now running on a ${{ runner.os }} server hosted by GitHub!"
      - run: echo "🔎 Name of branch is ${{ github.ref }} and repo is ${{ github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v4
      - run: echo "💡 The ${{ github.repository }} repository has been cloned to the runner."
      - run: echo "🖥️ The workflow is now ready to test your code on the runner."
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🍏 This job's status is ${{ job.status }}."
```

## Maven Build Workflow

- New Spring Boot project
- Build workflow for maven projects
- Run on push to master
- Using workflow templates - GitHub Docs

## Maven Build Workflow

- New Spring Boot project
- Build workflow for maven projects
- Run on push to master
- Using workflow templates - GitHub Docs

```yaml
name: Java CI with Maven

on:
  push:
    branches: [ "master" ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4.1.7
    - name: Set up JDK 17
      uses: actions/setup-java@v4.4.0
      with:
        java-version: '17'
        distribution: 'temurin'
        # cache: maven
    - name: Build with Maven
      run: mvn -B package --file pom.xml
```

## Test and Collect Artifacts

- Run tests
- Collect surefire reports

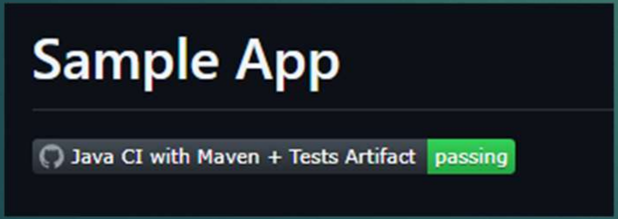
## Test and Collect Artifacts

- Run tests
- Collect surefire reports

```yaml
- name: Archive Test Reports
  # Always run even if the build fails
  if: always()
  uses: actions/upload-artifact@v4.4.0
  with:
    name: test-reports
    path: target/surefire-reports/
```

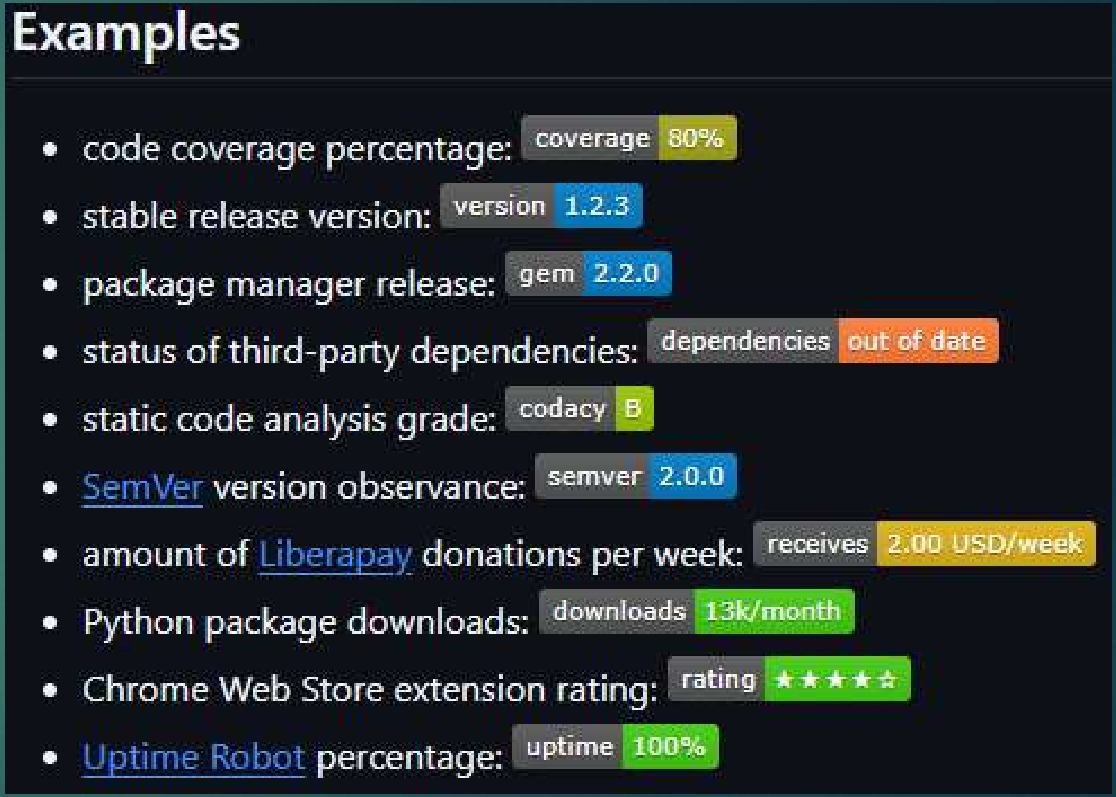
## Creating Build Badges

- Markdown with link to an .svg
- Live updates
- Very easy to create from actions



Sample App

Java CI with Maven + Tests Artifact passing



Examples

- code coverage percentage: coverage 80%
- stable release version: version 1.2.3
- package manager release: gem 2.2.0
- status of third-party dependencies: dependencies out of date
- static code analysis grade: codacy B
- SemVer version observance: semver 2.0.0
- amount of Liberapay donations per week: receives 2.00 USD/week
- Python package downloads: downloads 13k/month
- Chrome Web Store extension rating: rating ★★★★☆
- Uptime Robot percentage: uptime 100%

Make your own badges! - https://img.shields.io/badges/static-badge

## Caching Build Dependencies

- pom.xml (Java)
- *.gradle* (Java)
- package-lock.json (npm)
- yarn-lock.json (yarn)
- requirements.txt (python)
- go.sum (Go)

## Caching Build Dependencies

- pom.xml (Java)
- *.gradle* (Java)
- package-lock.json (npm)
- yarn-lock.json (yarn)
- requirements.txt (python)
- go.sum (Go)

```yaml
# Caching Mechanism
- name: Cache local Maven repository
  uses: actions/cache@v4.0.2
  with:
    path: ~/.m2/repository
    key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
    restore-keys: |
      ${{ runner.os }}-maven-
```

## Caching Build Dependencies

- pom.xml (Java)
- *.gradle* (Java)
- package-lock.json (npm)
- yarn-lock.json (yarn)
- requirements.txt (python)
- go.sum (Go)

```yaml
# Caching Mechanism
- name: Cache local Maven repository
  uses: actions/cache@v4.0.2
  with:
    path: ~/.m2/repository
    key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
    restore-keys: |
      ${{ runner.os }}-maven-

# Caching directly on JDK Setup
- name: Set up JDK 17
  uses: actions/setup-java@v4.4.0
  with:
    java-version: '17'
    distribution: 'temurin'
    cache: maven
```

## Commit with Skip CI

- What if we want to skip pipelines ❔

- If a commit message contains any of these identifiers, CI will be skipped
  - [skip ci]
  - [ci skip]
  - [no ci]
  - [skip actions]
  - [actions skip]

- e.g. git commit -m “Minor typo fix in README.md [skip ci]”

## Security Scanning / Dependency Updates

Dependabot automatically checks for vulnerabilities in project dependencies

- 3 Features (all configurable)
  - Alerts for Vulnerabilities
  - PR for dependency updates when vulnerability is found
  - PR for dependency updates to stay up-to-date

- https://github.com/dependabot/demo

## Security Scanning / Dependency Updates

Dependabot automatically checks for vulnerabilities in project dependencies

- 3 Features (all configurable)
  - Alerts for Vulnerabilities
  - PR for dependency updates when vulnerability is found
  - PR for dependency updates to stay up-to-date

```yaml
# Weekly check for updates to GitHub Actions
version: 2
updates:
  - package-ecosystem: "github-actions"
    directory: "/"
    schedule:
      interval: "weekly"
```

- https://github.com/dependabot/demo

- Can be used to maintain Actions!

## GitHub Actions – Advanced Use Cases

## Using GitHub Secrets

- Integrate an H2 database and provide secrets via GitHub
- Set GitHub Secrets (DB_USERNAME, DB_PASSWORD)

# Using GitHub Secrets

- Integrate an H2 database and provide secrets via GitHub
- Set GitHub Secrets (DB_USERNAME, DB_PASSWORD)

```yaml
- name: Set Database Secrets
  run: |
    sed -i 's/##db.username##/${{ secrets.DB_USERNAME }}/g' src/main/resources/application.yaml
    sed -i 's/##db.password##/${{ secrets.DB_PASSWORD }}/g' src/main/resources/application.yaml
```

## Coverage Report on Pull Requests

- Setup JaCoCo
- Collect artifact
- Publish Comment
- Fail PR if coverage is under limit

## Coverage Report on Pull Requests

- Setup JaCoCo
- Collect artifact
- Publish Comment
- Fail PR if coverage is under limit

```xml
<!-- Jacoco Maven Plugin for Code Coverage -->
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.12</version>
</plugin>
```

## Coverage Report on Pull Requests

- Setup JaCoCo
- Collect artifact
- Publish Comment
- Fail PR if coverage is under limit

```yaml
# Persist the JaCoCo code coverage report
- name: Archive Code Coverage Report
  if: always()
  uses: actions/upload-artifact@v4.4.0
  with:
    name: coverage-report
    path: target/site/jacoco/
```

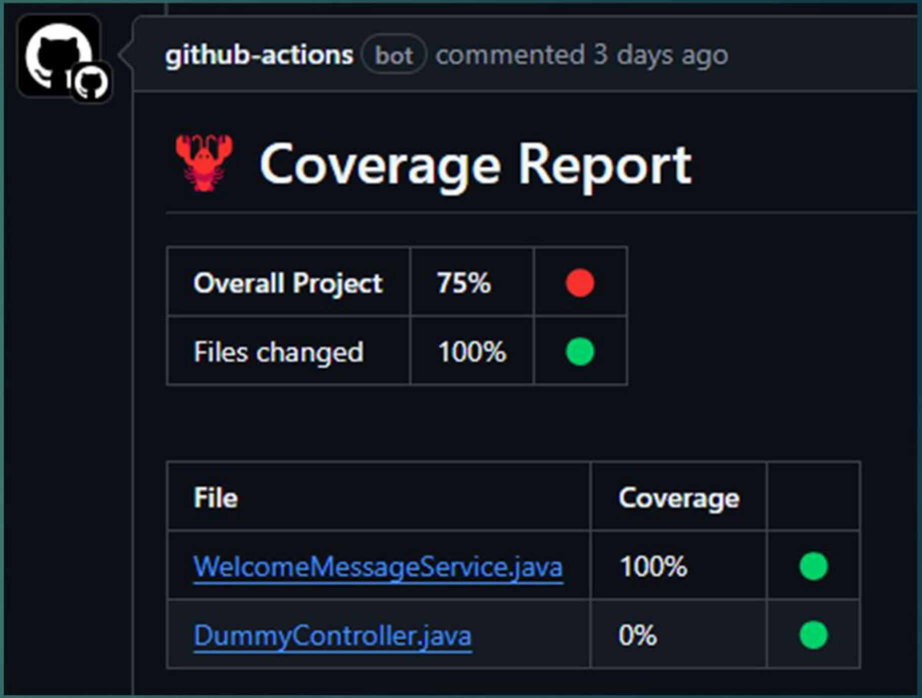
## Coverage Report on Pull Requests

- Setup JaCoCo
- Collect artifact
- Publish Comment (permissions)
- Fail PR if coverage is under limit

```yaml
- name: Add coverage to PR
  id: jacoco
  uses: madrapps/jacoco-report@v1.7.1
  with:
    paths: |
      target/site/jacoco/jacoco.xml
    token: ${{ secrets.GITHUB_TOKEN }}
    min-coverage-overall: 70
    min-coverage-changed-files: 90
    title: '# :lobster: Coverage Report'
    pass-emoji: ':green_circle:'
    fail-emoji: ':red_circle:'
```

# Coverage Report on Pull Requests

- Setup JaCoCo
- Collect artifact
- Publish Comment (permissions)
- Fail PR if coverage is under limit



**github-actions** bot commented 3 days ago

## Coverage Report

| | | |
|---|---|---|
| **Overall Project** | **75%** | 🔴 |
| Files changed | 100% | 🟢 |

| **File** | **Coverage** | |
|---|---|---|
| WelcomeMessageService.java | 100% | 🟢 |
| DummyController.java | 0% | 🟢 |

# Coverage Report on Pull Requests

- Setup JaCoCo
- Collect artifact
- Publish Comment (permissions)
- Fail PR if coverage is under limit

```
- name: Fail PR if overall coverage is less than 70%
  if: ${{ steps.jacoco.outputs.coverage-overall < 70.0 }}
  uses: actions/github-script@v7.0.1
  with:
    script: |
      core.setFailed('Coverage is less than 70%!')
```

# Multiple Environments - Matrix

- Run Workflow on different environment – in Parallel
- Great option to check compatibility
- Example: JDK versions (17 and 19)
- Example: OS (windows and ubuntu)

## Multiple Environments - Matrix

- Run Workflow on different environment
  - in Parallel
- Great option to check compatibility
- Example: JDK versions (17 and 19)
- Example: OS (windows and ubuntu)

```yaml
strategy:
  matrix:
    java-version: [17, 19]
    os: [windows-latest, ubuntu-latest]
runs-on: ${{ matrix.os }}

steps:
# Other steps...
- name: Set up JDK ${{ matrix.java-version }}
  uses: actions/setup-java@v4
  with:
    java-version: ${{ matrix.java-version }}
    distribution: 'temurin'
    cache: maven
```

## Notifications

- Email is already setup (on failures)
- Slack (via WebHook - paid only)
- Teams (TEAMS_WEBHOOK_URL )
- Discord (via DISCORD_WEBHOOK_URL)
- Telegram (Bot via BotFather and TELEGRAM_CHAT_ID)

## Notifications

- Email is already setup (on failures)
- Slack (via WebHook - paid only)
- Teams (TEAMS_WEBHOOK_URL )
- Discord (via DISCORD_WEBHOOK_URL)
- Telegram (Bot via BotFather and TELEGRAM_CHAT_ID)

```
# Notification to Slack
- name: Send custom JSON data to Slack workflow
  id: slack
  uses: slackapi/slack-github-action@v1.27.0
  with:
    payload-file-path: "./payload-slack.json"
  env:
    SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

## Deploy Container – GitHub Registry

### Prerequisites

- ghcr is free!

  [Working with the Container registry - GitHub Docs]

- packages: write

- vim ~/.docker/config.json

```
{
  "auths": {
    "ghcr.io": {
      "auth": "TOKEN_FROM_GITHUB"
    }
  }
}
```

```
# Log in to GitHub Container Registry
- name: Log in to GitHub Container Registry
  run: echo "${{ secrets.GITHUB_TOKEN }}" | docker login
ghcr.io -u ${{ github.actor }} --password-stdin

# Build Image
- name: Build Docker image
  run: docker build -t ghcr.io/lefterisxris/gh_actions:latest .

# Push Docker images to GitHub Container Registry
- name: Push Docker image to GHCR
  run: docker push ghcr.io/lefterisxris/gh_actions:latest
```

## Deploy – SSH

```yaml
# SSH into a Server and deploy
- name: Deploy to Server over SSH
  env:
    SSH_PRIVATE_KEY: ${{ secrets.SERVER_DEPLOY_KEY }}
    SERVER_HOST: ${{ secrets.SERVER_HOST }}
    SERVER_USER: ${{ secrets.SERVER_USER }}
    PKI_SSH_PASSPHRASE: ${{ secrets.PKI_SSH_PASSPHRASE }}
  run: |
    mkdir -p ~/.ssh
    echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
    chmod 600 ~/.ssh/id_rsa
    eval $(ssh-agent -s)
    echo "$PKI_SSH_PASSPHRASE" | ssh-add ~/.ssh/id_rsa
    ssh-keyscan $SERVER_HOST >> ~/.ssh/known_hosts

    ssh $SERVER_USER@$SERVER_HOST << 'EOF'
      cd /home/user/project_with_docker_compose_file
      docker compose down 2>&1
      docker compose pull 2>&1
      docker compose up -d 2>&1
    EOF

# scp app.jar $SERVER_USER@$SERVER_HOST:$TARGET_DIR/app.jar
```

## Create Draft Release

```yaml
draftReleaseJob:
  name: Create Draft Release
  runs-on: ubuntu-latest
  needs: build
  permissions: write-all
  if: github.event_name == 'push' &&
startsWith(github.event.head_commit.message,
'Merge pull request')

  steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Download build artifacts
      uses: actions/download-artifact@v4.1.8
      with:
        name: app-executables
        path: ./downloads
```

```yaml
- name: Create Draft Release
      uses: softprops/action-gh-release@v2.0.8
      with:
        files: ./downloads/**
        name: "Draft release after so much good work!"
        body: |
          # The anticipated Release is here! :lobster:

          ## Changelog:
          - Feature 1: Added support for X
          - Feature 2: Improved performance for Y
          - Bugfix: Resolved issue with Z

          Thank you for your contributions!
        draft: true
        tag_name: ${{ github.sha }}
```

## Microservices

- Supported as containers
- Normal usage after declaring it

```
services:
  mysql:
    image: mysql:8.0
    ports:
      - 3306:3306
    env:
      MYSQL_ROOT_PASSWORD: ${{
secrets.DB_ROOR_PWD }}
      MYSQL_DATABASE: ${{ secrets.DB_NAME }}
      MYSQL_USER: ${{ secrets.DB_USERNAME }}
      MYSQL_PASSWORD: ${{ secrets.DB_PASSWORD }}

# And use like
SPRING_DATASOURCE_URL:
jdbc:mysql://localhost:3306/testdb
```

# Custom Actions

- .github/actions/word-search/action.yml
- .github/actions/word-search/word-search.sh
- Demo: Custom Actions - check for a word

```yaml
name: "Word Search Action"
description: "Searches for a..."
author: "Your Name"

inputs:
  word:
    required: true
    default: "todo"

runs:
  using: "composite"
  steps:
    - run: ./word-search.sh
      shell: bash
      working-directory: ${{ github.workspace }}
      with:
        word: ${{ inputs.word }}
```

## Custom Actions

- .github/actions/word-search/action.yml
- .github/actions/word-search/word-search.sh
- Demo: Custom Actions - check for a word

```bash
#!/bin/bash

WORD_TO_SEARCH="${1:-todo}"

# Recursively search for the word in all files
FOUND=$(grep -r -l "$WORD_TO_SEARCH" .)

if [ -n "$FOUND" ]; then
  echo "Word \"$WORD_TO_SEARCH\" found in the following files:"
  echo "$FOUND"
  echo "Failing the build."
  exit 1  # Fail the action
else
  echo "Word \"$WORD_TO_SEARCH\" not found in any file. Build is successful."
  exit 0  # Success
fi
```

## Useful Resources

## Useful Resources

- Documentation - GitHub Actions documentation - GitHub Docs
- Marketplace - Marketplace · Tools to improve your workflow
- Checking free balance – Profile > Settings > Billing
- Setting limit - to avoid being charged
- Testing locally
  - nektos/act: Run your GitHub Actions locally 🚀

# QA Time!