# SWE 261P Software Testing and Analysis - Part 4 Report

## PDFsam Basic: Continuous Integration






**Repo Github Link:**
https://github.com/eric-song-dev/pdfsam

**Team Members:**


- Kingson Zhang: kxzhang@uci.edu
- Zian Xu: zianx11@uci.edu
- Zhenyu Song: zhenyus4@uci.edu


This report documents the setup and configuration of **Continuous Integration (CI)** for **PDFsam Basic** using GitHub Actions, including the CI pipeline configuration, build and test automation, and verification results.

# 📂 Quick Navigation

# 🎯 1. Continuous Integration: Definition and Purpose

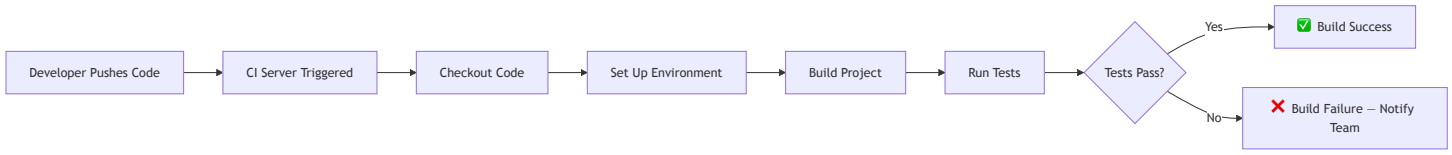
## 1.1 What Is Continuous Integration?

**Continuous Integration (CI)** is a software development practice where developers frequently integrate their code changes into a shared repository, ideally several times a day. Each integration is automatically verified by building the project and running automated tests, allowing teams to detect errors early and reduce integration problems.

## 1.2 Why Continuous Integration Matters

| Benefit | Description |
|---|---|
| **Early Bug Detection** | Automated builds and tests catch errors immediately after code is pushed, before they propagate to other team members |
| **Reduced Integration Risk** | Frequent integration of small changes avoids "integration hell" that occurs when merging large, divergent branches |
| **Consistent Build Environment** | CI ensures the project builds and passes tests in a clean, reproducible environment, not just on one developer's machine |
| **Faster Feedback Loop** | Developers receive immediate notification if their changes break the build or fail tests |
| **Quality Assurance** | Enforces that all tests pass before code is merged, maintaining a stable main branch |
| **Documentation of Build State** | CI history provides a clear record of when builds succeeded or failed, and what changes caused failures |

## 1.3 CI Workflow



# 🔧 2. CI Configuration

## 2.1 Workflow Files

The CI is split into **three independent workflows**, one per team member, each targeting a specific module:

| Workflow File | Owner | Target Module |
|---|---|---|
| ZhenyuCi.yml | Zhenyu Song | `pdfsam-model` |
| ZianCi.yml | Zian Xu | `pdfsam-persistence` |
| KingsonCi.yml | Kingson Zhang | `pdfsam-core` |

### ZhenyuCi.yml — PDFsam CI on Model

```yaml
name: PDFsam CI on Model

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build-and-test-on-model:
    runs-on: ubuntu-latest

    env:
      CI: "true"

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

      - name: Set up JDK 21
        uses: actions/setup-java@v4
        with:
          java-version: '21'
          distribution: 'temurin'
          cache: maven

      - name: Build and Test
        run: mvn clean test --batch-mode -pl pdfsam-model -am -Dmaven.antrun.skip=true
```

## ZianCi.yml — PDFsam CI on Persistence

```yaml
name: PDFsam CI on Persistence

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build-and-test-on-persistence:
    runs-on: ubuntu-latest

    env:
      CI: "true"

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

      - name: Set up JDK 21
        uses: actions/setup-java@v4
        with:
          java-version: '21'
          distribution: 'temurin'
          cache: maven

      - name: Build and Test
        run: mvn clean test --batch-mode -pl pdfsam-persistence -am -Dmaven.antrun.skip=true
```

## KingsonCi.yml — PDFsam CI on Core

```yaml
name: PDFsam CI on Core

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build-and-test-on-core:
    runs-on: ubuntu-latest

    env:
      CI: "true"

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

      - name: Set up JDK 21
        uses: actions/setup-java@v4
        with:
          java-version: '21'
          distribution: 'temurin'
          cache: maven

      - name: Build and Test
        run: mvn clean test --batch-mode -pl pdfsam-core -am -Dmaven.antrun.skip=true
```

## 2.2 Configuration Walkthrough

### Trigger Events

```yaml
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]
```

The workflow runs on:

- **Every push** to the `master` branch
- **Every pull request** targeting the `master` branch

This ensures that both direct commits and PR-based contributions are validated before merging.

### Runner Environment

```yaml
runs-on: ubuntu-latest
```

We use the `ubuntu-latest` GitHub-hosted runner, which provides a clean Linux environment for each build. This ensures reproducibility and avoids "works on my machine" issues.

### CI Environment Variable

```yaml
env:
  CI: "true"
```

Setting `CI=true` activates our existing Maven CI profile ( `no-headless-failing-tests` ) defined in `pom.xml` . This profile:

1. **Excludes `NoHeadless` tagged tests** — JavaFX GUI tests that cannot run in a headless CI environment
2. **Disables module path** for surefire — avoids Java module system complications in CI
3. **Adds `--enable-preview`** — required for Foreign Function & Memory API features
4. **Includes `javafx-monocle`** dependency — provides a headless JavaFX rendering backend for tests that use JavaFX components

The relevant Maven profile in `pom.xml` :

```
<profile>
    <activation>
        <property>
            <name>env.CI</name>
            <value>true</value>
        </property>
    </activation>
    <id>no-headless-failing-tests</id>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <configuration>
                    <excludedGroups>NoHeadless</excludedGroups>
                    <useModulePath>false</useModulePath>
                    <argLine>--enable-preview</argLine>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.pdfsam</groupId>
            <artifactId>javafx-monocle</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
</profile>
```

## Java Setup with Caching

```
- name: Set up JDK 21
  uses: actions/setup-java@v4
  with:
    java-version: '21'
    distribution: 'temurin'
    cache: maven
```

- **Temurin JDK 21**: Matches our development environment requirement (Java 21 with preview features)
- **Maven caching**: Caches `~/.m2/repository` to speed up subsequent builds by avoiding repeated dependency downloads

## Build and Test Execution

Each workflow targets a single module with its own `-pl` parameter:

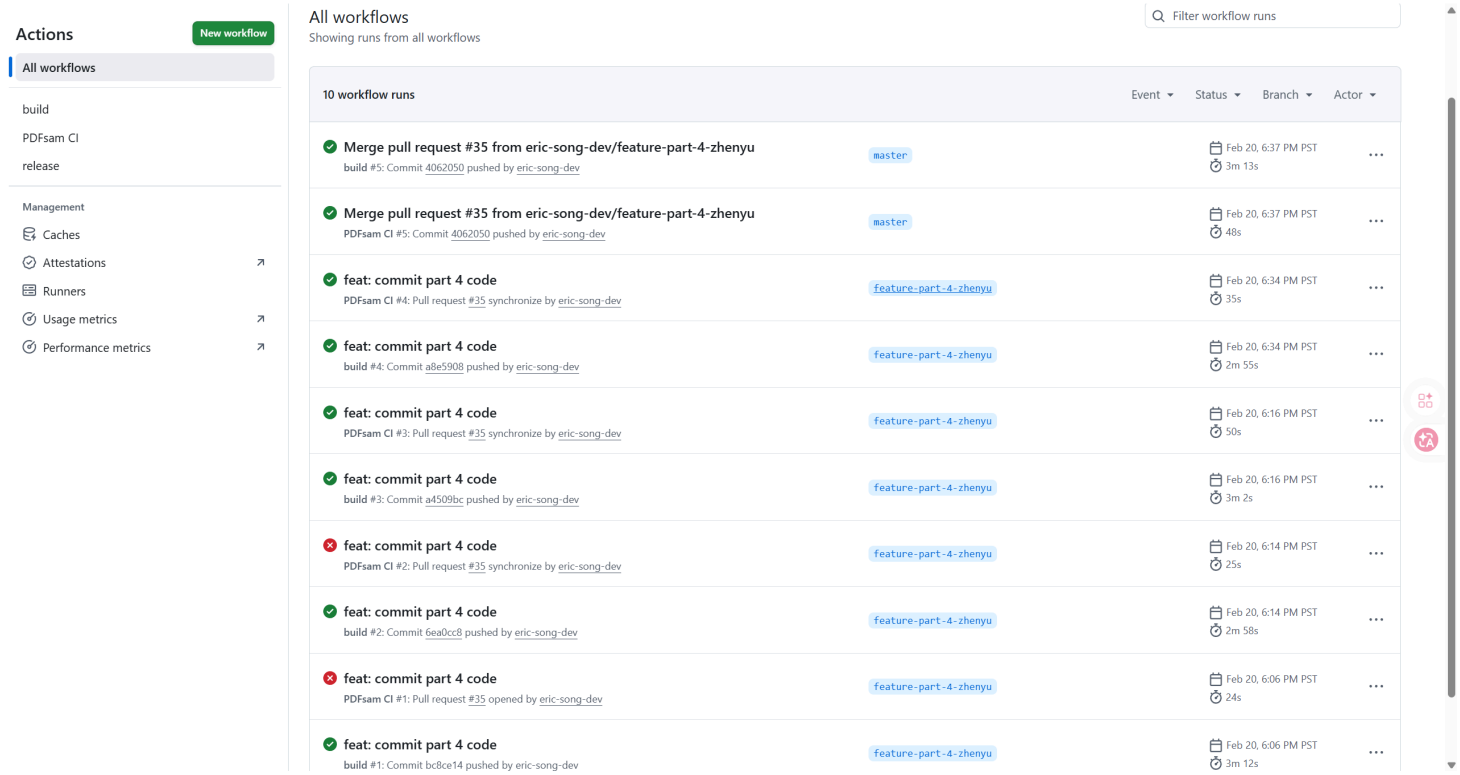
| Workflow | Build Command |
| --- | --- |
| ZhenyuCi | `mvn clean test --batch-mode -pl pdfsam-model -am -Dmaven.antrun.skip=true` |
| ZianCi | `mvn clean test --batch-mode -pl pdfsam-persistence -am -Dmaven.antrun.skip=true` |
| KingsonCi | `mvn clean test --batch-mode -pl pdfsam-core -am -Dmaven.antrun.skip=true` |

- `mvn clean test` — Cleans the build directory, compiles, and runs JUnit tests
- `--batch-mode` — Disables interactive input and produces cleaner log output suitable for CI
- `-pl <module>` — Only builds and tests the specified non-GUI module
- `-am` (also-make) — Also builds any modules that the target depends on (e.g., `pdfsam-i18n`, `pdfsam-themes`)
- `-Dmaven.antrun.skip=true` — Skips the Ant run plugin tasks (e.g., resource copying or pre-processing steps) that are unnecessary in the CI test environment

Splitting into three independent workflows allows each team member's module to build and report status separately.

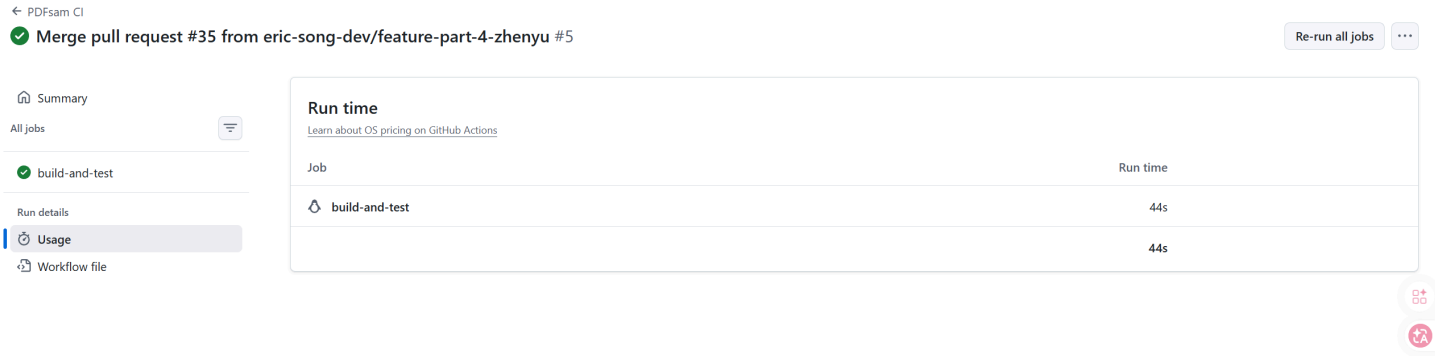
# 2.3 Screenshots

Action run history demonstrating the iterative process from initial build failures to consistent successes.




Modified CI workflow configuration (ci.yml) to bypass the problematic plugin by appending a skip parameter.
![image ci.yml Configuration](screenshot/ci.yml Configuration.png)

GitHub Actions successfully completed the build-and-test job after modifying the workflow configuration.




Detailed Maven logs confirming successful compilation and testing across all core modules.

```
[INFO] --- jacoco:0.8.12:report (report) @ pdfsam-core ---
[INFO] Skipping JaCoCo execution due to missing execution data file.
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for PDFsam 5.4.5-SNAPSHOT:
[INFO]
[INFO] PDFsam ............................................. SUCCESS [  5.718 s]
[INFO] PDFsam internationalization ........................ SUCCESS [  7.959 s]
[INFO] PDFsam model ....................................... SUCCESS [  6.675 s]
[INFO] PDFsam persistence ................................. SUCCESS [  4.779 s]
[INFO] PDFsam themes ...................................... SUCCESS [  0.137 s]
[INFO] PDFsam test ........................................ SUCCESS [  0.181 s]
[INFO] PDFsam core ........................................ SUCCESS [  5.439 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  33.191 s
[INFO] Finished at: 2026-02-21T02:38:17Z
[INFO] ------------------------------------------------------------------------
```

# 🚀 3. Build and Test Results

## 3.1 Triggering the CI Build

After committing the workflow file to the repository, every push to `master` automatically triggers the CI pipeline.

```
# Add the CI workflow files
git add .github/workflows/ZhenyuCi.yml
git add .github/workflows/ZianCi.yml
git add .github/workflows/KingsonCi.yml
git commit -m "Add GitHub Actions CI workflows for automated build and testing"
git push origin master
```

## 3.2 GitHub Actions Dashboard

After pushing, the CI workflow can be monitored from the **Actions** tab at:
https://github.com/eric-song-dev/pdfsam/actions

The dashboard shows:

- **Workflow runs** with status (✅ success, ❌ failure, 🔄 in progress)
- **Run duration** and **trigger event** (push, pull_request)
- **Detailed logs** for each step of the build process

## 3.3 Build Steps and Output

Each of the three workflows executes the same steps, differing only in the target module:

| Step | Description | Expected Duration |
|---|---|---|
| **Checkout** | Clone the repository | ~5s |
| **Set up JDK 21** | Install Temurin JDK 21, restore Maven cache | ~15s |
| **Build and Test** | `mvn clean test --batch-mode -pl <module> -am -Dmaven.antrun.skip=true` | ~30s |

## 3.4 Test Execution Summary

The CI build compiles and runs all tests across the three targeted non-GUI modules ( `pdfsam-model` , `pdfsam-core` , `pdfsam-persistence` ), including both the project's existing test suite and our custom tests from Parts 1–3 (partition tests, FSM tests, and white-box tests).

**Per-Module CI Output:**

| Workflow | Module | Tests Run | Failures | Errors | Skipped | Result |
|---|---|---|---|---|---|---|
| ZhenyuCi | `pdfsam-model` | 176 | 0 | 0 | 2 | ✅ BUILD SUCCESS |
| KingsonCi | `pdfsam-core` | 139 | 0 | 0 | 0 | ✅ BUILD SUCCESS |
| ZianCi | `pdfsam-persistence` | 75 | 0 | 0 | 0 | ✅ BUILD SUCCESS |
| **Total** | | **390** | **0** | **0** | **2** | ✅ **ALL PASSED** |

All **390 tests** across three modules passed successfully with **zero failures or errors**, confirming that the CI pipeline correctly builds and validates the project.

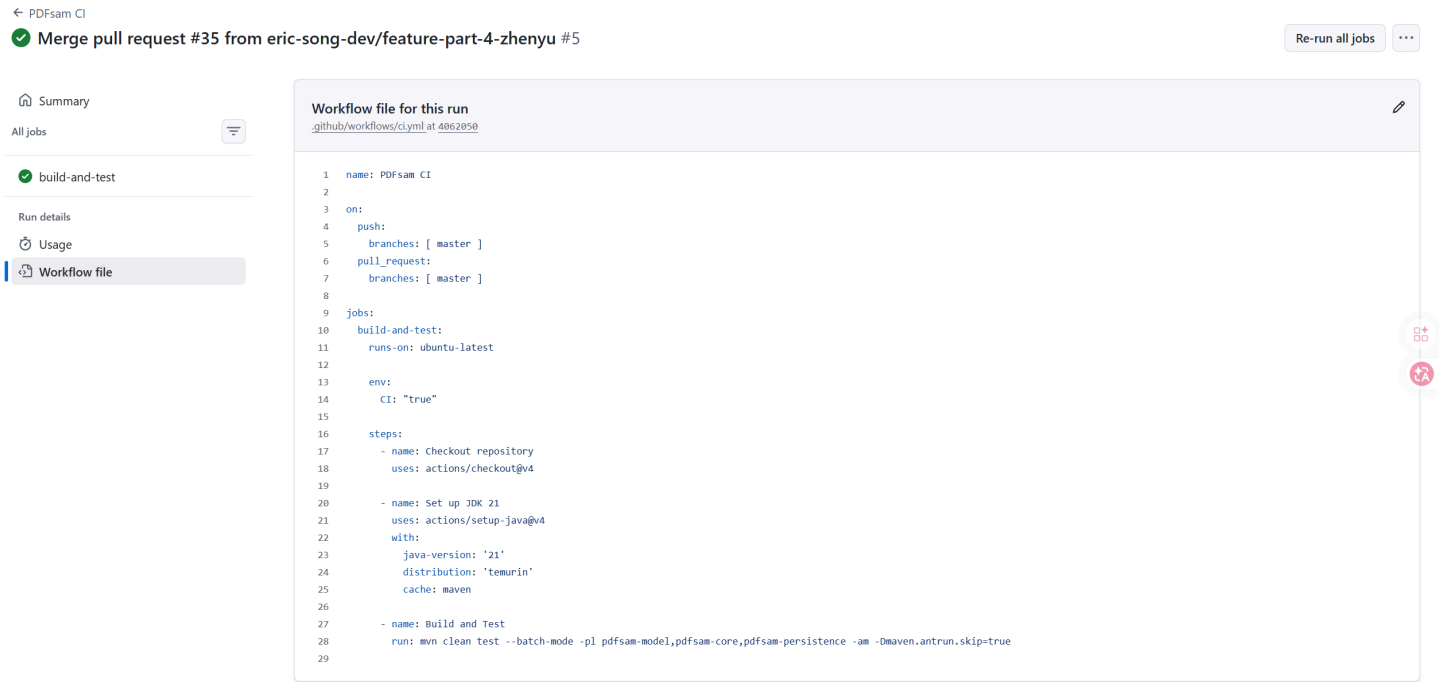
# ⚠️ 4. Issues and Resolutions

**Problem:** During the build process, the maven-antrun-plugin fails in the pdfsam-i18n module because it cannot find the Messages_iw.properties file to execute the rename-hebrew-properties task.

**Resolution:** Bypassed the problematic plugin execution by appending **-Dmaven.antrun.skip=true** to the Maven build command in the CI workflow, allowing the core compilation and testing to proceed.

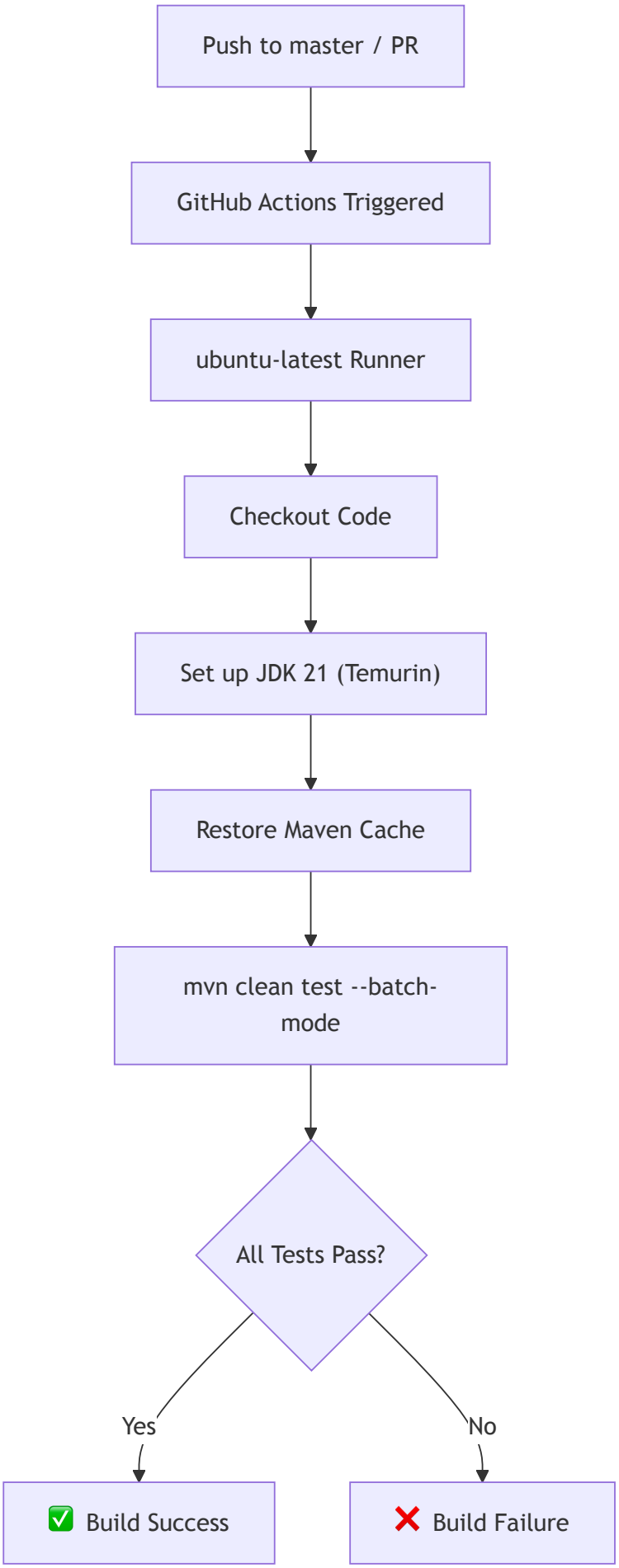
Build and Test 14s

```
6179 [INFO] Executing tasks
6180 [INFO] ------------------------------------------------------------------------
6181 [INFO] Reactor Summary for PDFsam 5.4.5-SNAPSHOT:
6182 [INFO]
6183 [INFO] PDFsam ..................................... SUCCESS [  4.806 s]
6184 [INFO] PDFsam internationalization ..................... FAILURE [  3.270 s]
6185 [INFO] PDFsam model ................................... SKIPPED
6186 [INFO] PDFsam persistence .............................. SKIPPED
6187 [INFO] PDFsam themes ................................... SKIPPED
6188 [INFO] PDFsam test ..................................... SKIPPED
6189 [INFO] PDFsam core ..................................... SKIPPED
6190 [INFO] ------------------------------------------------------------------------
6191 [INFO] BUILD FAILURE
6192 [INFO] ------------------------------------------------------------------------
6193 [INFO] Total time:  10.369 s
6194 [INFO] Finished at: 2026-02-21T02:14:39Z
6195 [INFO] ------------------------------------------------------------------------
6196 Error:  Failed to execute goal org.apache.maven.plugins:maven-antrun-plugin:3.2.0:run (rename-hebrew-properties) on project pdfsam-i18n: An Ant BuildException has occurred: Warning:
     Could not find file /home/runner/work/pdfsam/pdfsam/pdfsam-i18n/target/classes/org/pdfsam/i18n/Messages_iw.properties to copy.
6197 Error:  around Ant part ...<move file="/home/runner/work/pdfsam/pdfsam/pdfsam-i18n/target/classes/org/pdfsam/i18n/Messages_iw.properties"
     tofile="/home/runner/work/pdfsam/pdfsam/pdfsam-i18n/target/classes/org/pdfsam/i18n/Messages_he.properties" />... @ 4:225 in /home/runner/work/pdfsam/pdfsam/pdfsam-
     i18n/target/antrun/build-main.xml
6198 Error:  -> [Help 1]
6199 Error:
6200 Error:  To see the full stack trace of the errors, re-run Maven with the -e switch.
6201 Error:  Re-run Maven using the -X switch to enable full debug logging.
6202 Error:
6203 Error:  For more information about the errors and possible solutions, please read the following articles:
6204 Error:  [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
6205 Error:
6206 Error:  After correcting the problems, you can resume the build with the command
6207 Error:    mvn <args> -rf :pdfsam-i18n
6208 Error: Process completed with exit code 1.
```



# 📋 5. CI Pipeline Summary

## 5.1 CI Architecture


Push to master / PR
GitHub Actions Triggered
ubuntu-latest Runner
Checkout Code
Set up JDK 21 (Temurin)
Restore Maven Cache
mvn clean test --batch-mode
All Tests Pass?
Yes
No
✅ Build Success
❌ Build Failure

## 5.2 Files Changed

| File | Owner | Action | Description |
|---|---|---|---|
| `.github/workflows/ZhenyuCi.yml` | Zhenyu Song | **NEW** | CI workflow for `pdfsam-model` module |
| `.github/workflows/ZianCi.yml` | Zian Xu | **NEW** | CI workflow for `pdfsam-persistence` module |
| `.github/workflows/KingsonCi.yml` | Kingson Zhang | **NEW** | CI workflow for `pdfsam-core` module |

## 5.3 Running CI Locally

To replicate the CI environment locally:

```
# Simulate CI environment
export CI=true

# Run the same command as CI
mvn clean test --batch-mode -pl pdfsam-model -am
mvn clean test --batch-mode -pl pdfsam-core -am
mvn clean test --batch-mode -pl pdfsam-persistence -am
```

# 🎯 6. Conclusion

This report documents the setup of **Continuous Integration** for PDFsam Basic using **GitHub Actions**.

| Aspect | Details |
|---|---|
| **CI System** | GitHub Actions |
| **Trigger** | Push to `master`, Pull Requests to `master` |
| **Environment** | Ubuntu (latest), JDK 21 (Temurin) |
| **Build Tool** | Maven with `--batch-mode` |
| **Test Framework** | JUnit 5 |
| **Headless Support** | Monocle + `NoHeadless` tag exclusion |
| **Caching** | Maven dependency caching via `setup-java` |

## Key Takeaways

- **GitHub Actions** provides seamless CI integration for GitHub-hosted repositories with minimal configuration
- **Leveraging existing Maven profiles** ( `no-headless-failing-tests` ) made the CI setup straightforward — the project was already prepared for CI environments
- **Automatic test execution** on every push ensures that all regression tests (Part 1 partition tests, Part 2 FSM tests, Part 3 white-box tests) are continuously validated

The CI pipeline complements the testing efforts from **Part 1** (partition testing), **Part 2** (FSM testing), and **Part 3** (white-box testing) by ensuring all tests are automatically and consistently executed in a clean environment on every code change.