GitHub Actions 1

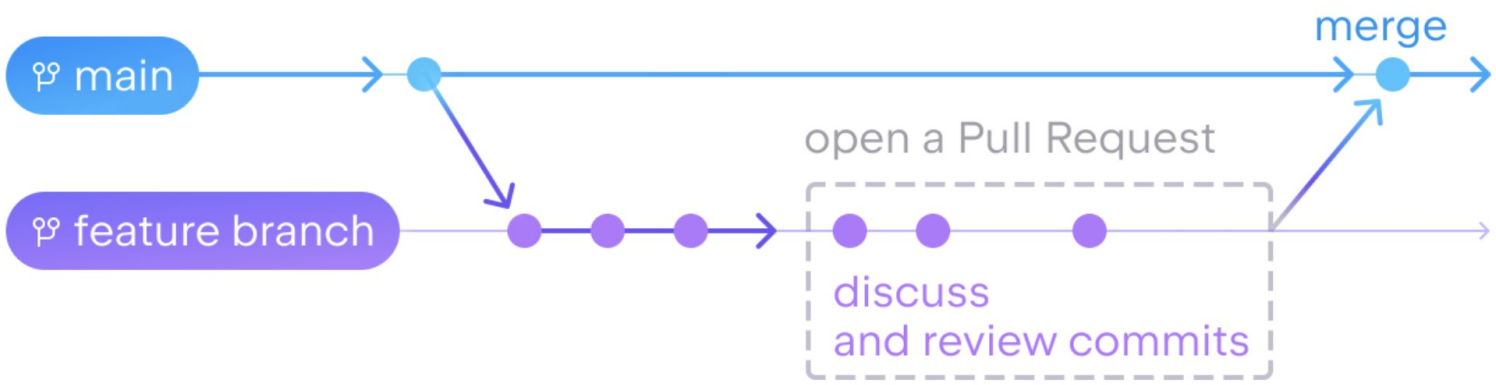
Læringsmål

- At kunne forklare og anvende GitHub Flow
- At kunne etablere et CI workflow med GitHub Actions

Indhold

- GitHub Flow
- Continuous Integration (CI) pipeline
- GitHub Actions
- CI pipeline med GitHub Actions

GitHub Flow (simplified version)



- The `main` branch is for production code only.
- Development is done in separate `feature` branches which are then merged into the `main` branch when ready.
- Pull requests are used for code reviews and must satisfy branch protection rules before being merged.
- Releases are cut from the `main` branch and tagged for easy versioning.

https://blog.jetbrains.com/space/2023/04/18/space-git-flow/

GitHub Flow

- Create a branch in your repository
- Make changes
- Create a pull request
- Address review comments
- Merge your pull request
- Delete your branch

GitHub Flow (in detail)

1. Update the local main branch
2. Create a local feature branch
3. Make commits to the feature branch
4. Update the local main branch
5. Merge the main branch into the feature branch
6. Push the local feature branch to the remote
7. On GitHub create a pull request and assign a reviewer
8. Reviewer reviews the changes
9. Merge the changes
10. Delete the feature branch

GitHub Actions

GitHub Actions

Automate your workflow from idea to production

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

Get started with Actions >

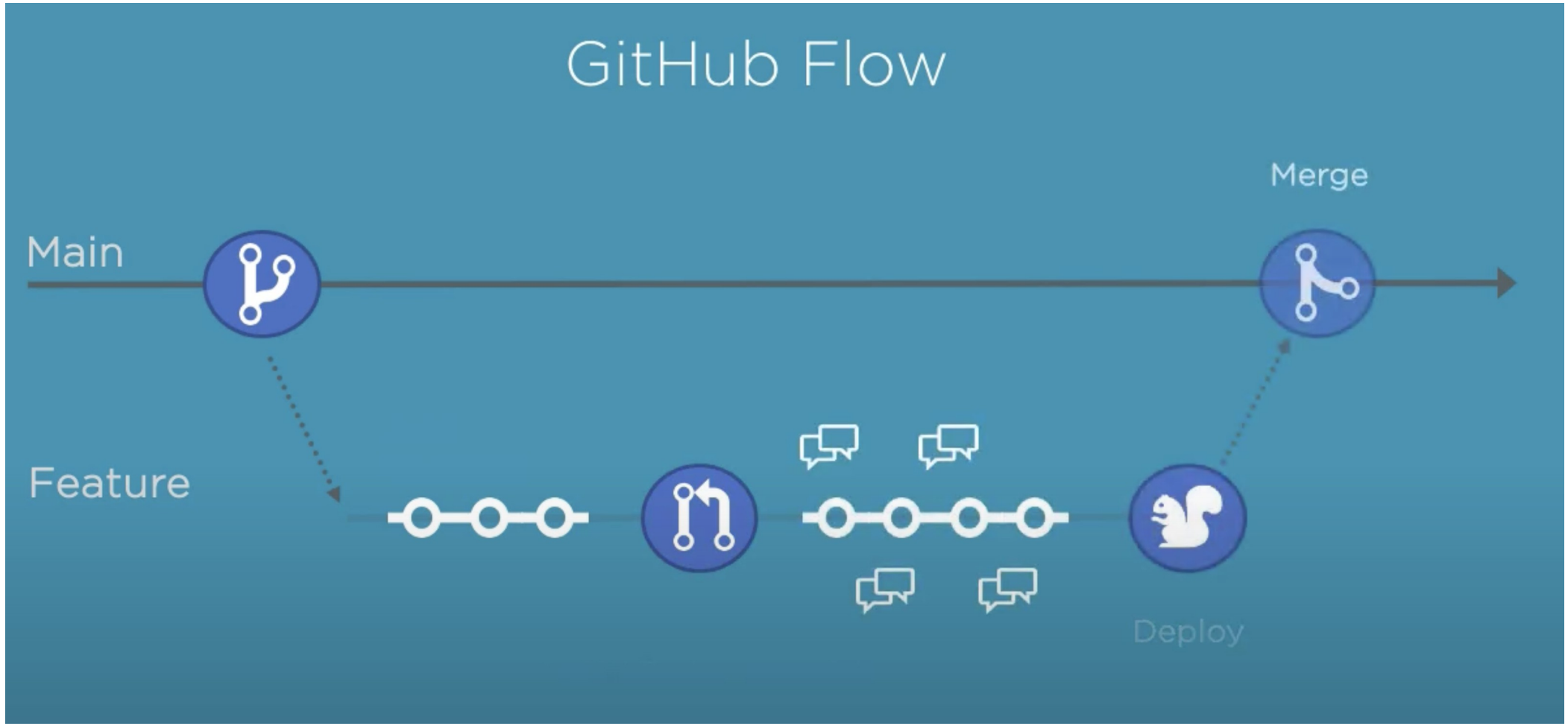
Questions? **Contact Sales** >

kea
KØBENHAVNS ERHVERVSAKADEMI

GitHub Actions

- GitHub Actions used to automate a team's software workflow.
- Can automate testing, continuously deploy, review code, manage issues and pull requests.
- Workflows are stored as code in the repository.

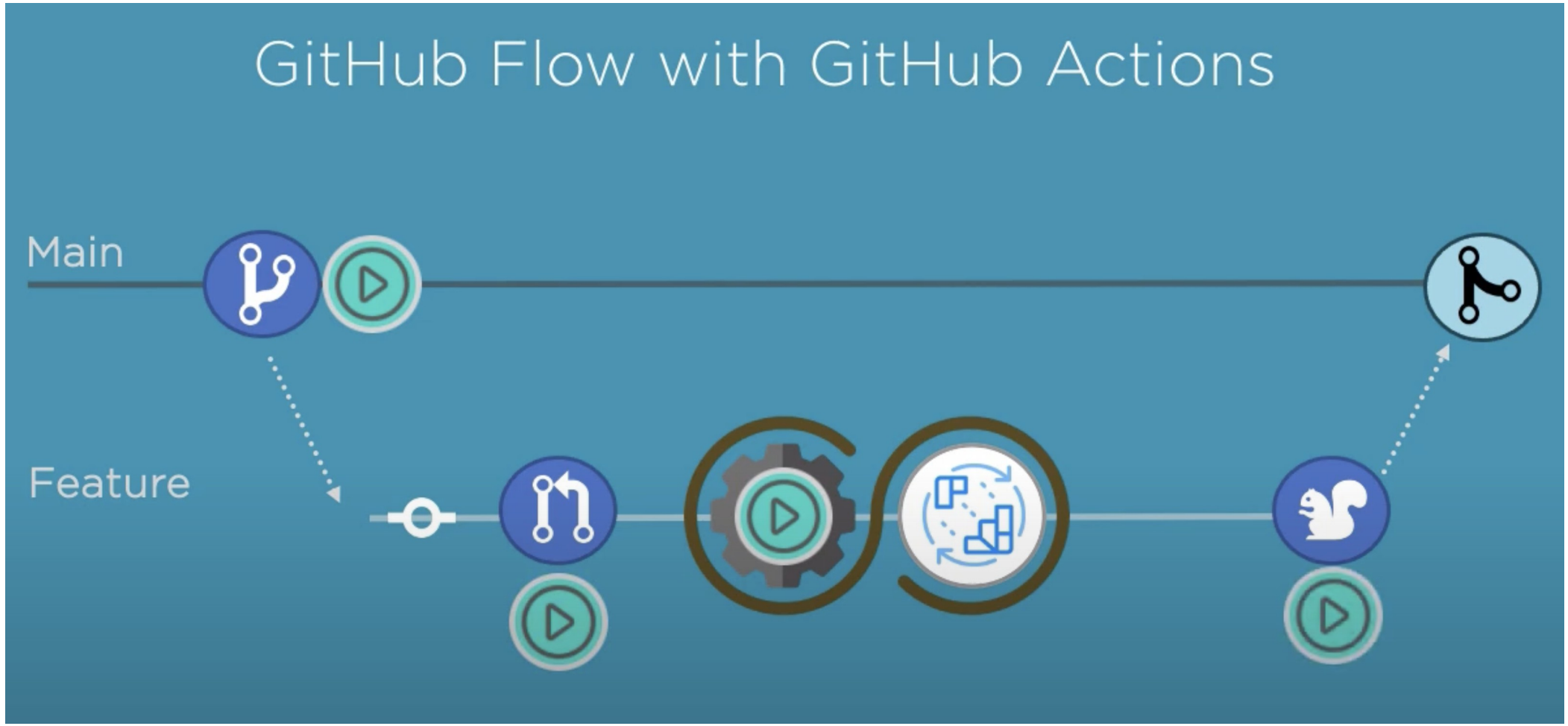
GitHub Flow



GitHub Actions

kea
KØBENHAVNS ERHVERVSAKADEMI

GitHub Flow with GitHub Actions



GitHub Actions

kea
KØBENHAVNS ERHVERVSAKADEMI

Workflow

- configurable automated process that will run one or more jobs

- defined by a YAML file checked in to your repository (in the .github/workflows directory)

- run when triggered by an event in your repository e.g. pull request or manually, or at a defined schedule.

Triggers

GitHub Actions

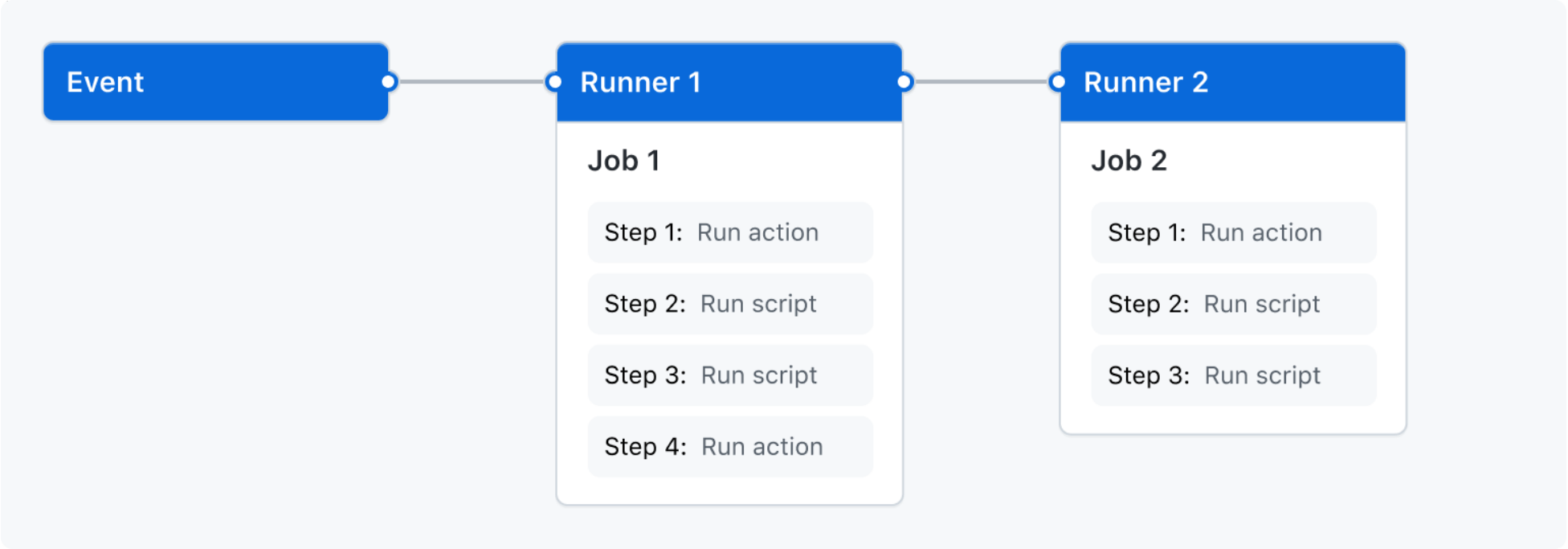
- Job
  - set of steps in a workflow.
  - either a shell script that will be executed, or an action that will be run.
  - steps are executed in order and are dependent on each other.
  - each step is executed on the same runner and can share data from one step to another. e.g a step that builds the application followed by a step that tests the application that was built.

GitHub Actions

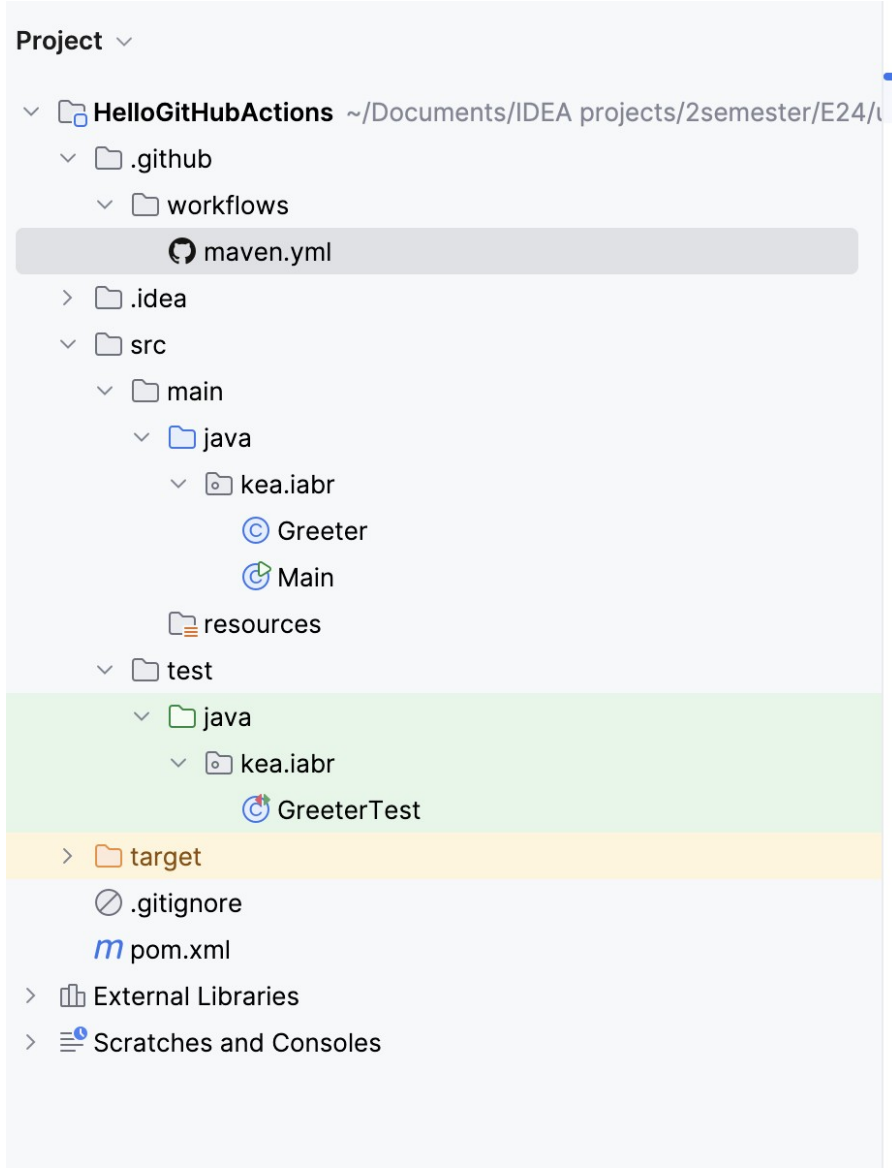
- Runner
  - a server that runs the workflows when they're triggered.
  - Each runner can run a single job at a time.
  - GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows.
  - Each workflow run executes in a fresh, newly-provisioned virtual machine.

GitHub Actions

- Understanding GitHub Actions



Workflow file



maven.yml

```yaml
name: Java CI with Maven

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

  workflow_dispatch:

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 22
      uses: actions/setup-java@v4.3.0
      with:
        java-version: '22'
        distribution: 'sapmachine'
        cache: maven
    - name: Build with Maven
      run: mvn -B test
```

events (triggers)

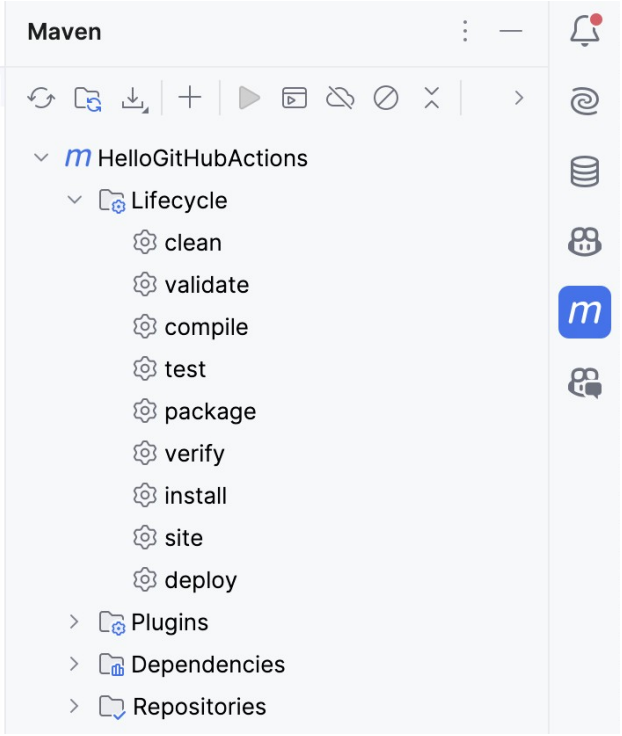
job(s)

steps

actions/commands

kea
KØBENHAVNS ERHVERVSAKADEMI

Maven



- Maven in 5 Minutes

Maven Phases

Although hardly a comprehensive list, these are the most common *default* lifecycle phases executed.

- **validate**: validate the project is correct and all necessary information is available
- **compile**: compile the source code of the project
- **test**: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package**: take the compiled code and package it in its distributable format, such as a JAR.
- **integration-test**: process and deploy the package if necessary into an environment where integration tests can be run
- **verify**: run any checks to verify the package is valid and meets quality criteria
- **install**: install the package into the local repository, for use as a dependency in other projects locally
- **deploy**: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

There are two other Maven lifecycles of note beyond the *default* list above. They are

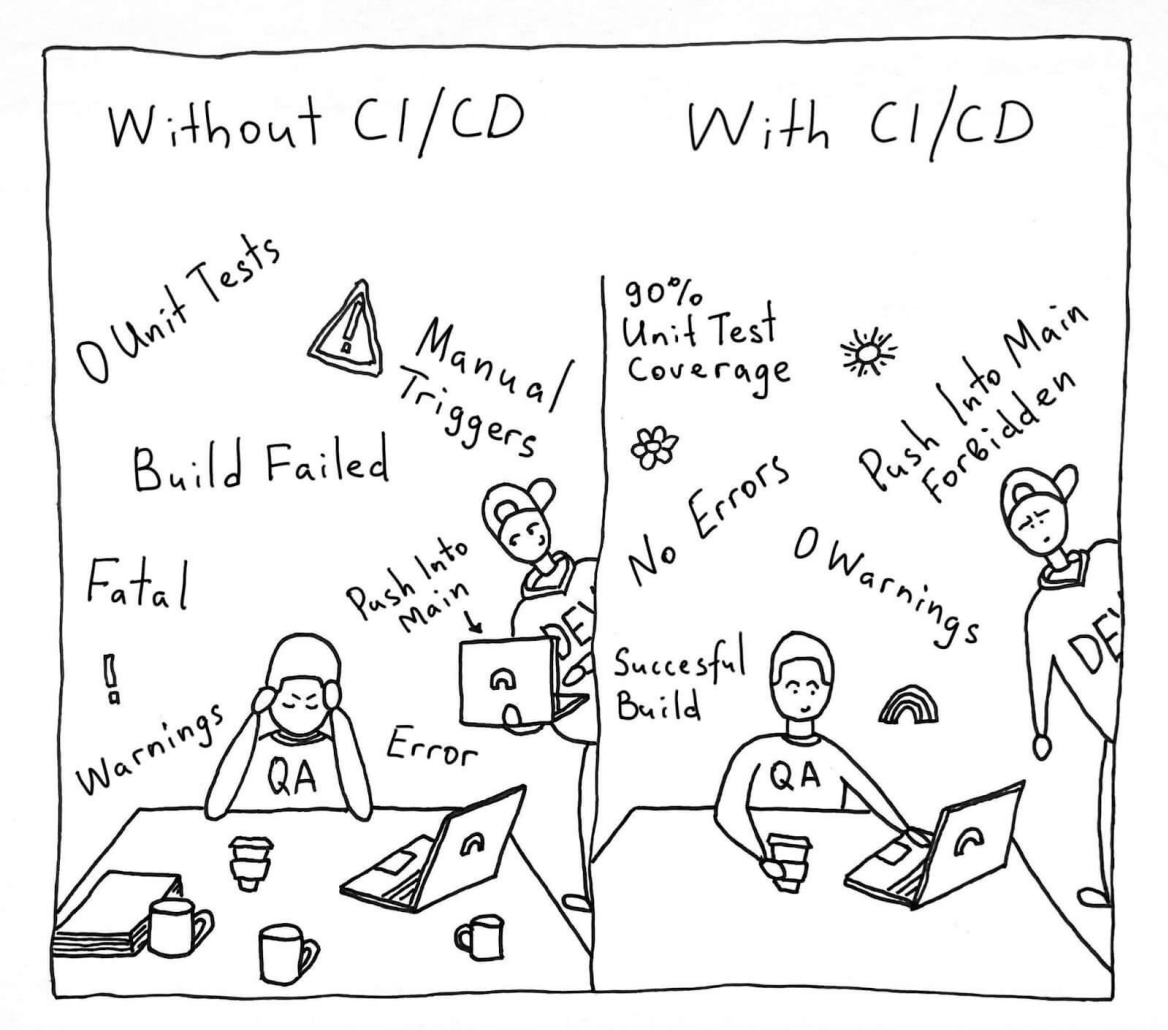
- **clean**: cleans up artifacts created by prior builds
- **site**: generates site documentation for this project

Continuous Integration (CI)

- Practice of automating the integration of code changes from multiple contributors into a single software project.
- Best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run.
- Automated tools are used to assert the new code’s correctness before integration.

Continuous Integration

- Should not commit code which:
  - breaks the build
  - does not pass tests
  - does not meet coding standards
  - etc…



https://blog.jetbrains.com/teamcity/2023/08/how-to-choose-cicd-tool/

kea
KØBENHAVNS ERHVERVSAKADEMI

CI and GitHub Actions

- Use GitHub Actions to create workflows that can build the code in the repository and run tests.
- Workflows (can) run on GitHub-hosted virtual machines.
- Configure a CI workflow to run when a GitHub event occurs (e.g. pull request).
- GitHub runs CI tests and provides the results of each test in the pull request.
- When all CI tests in a workflow pass, the changes are ready to be reviewed by a team member or merged.

Demo: CI with GitHub Actions

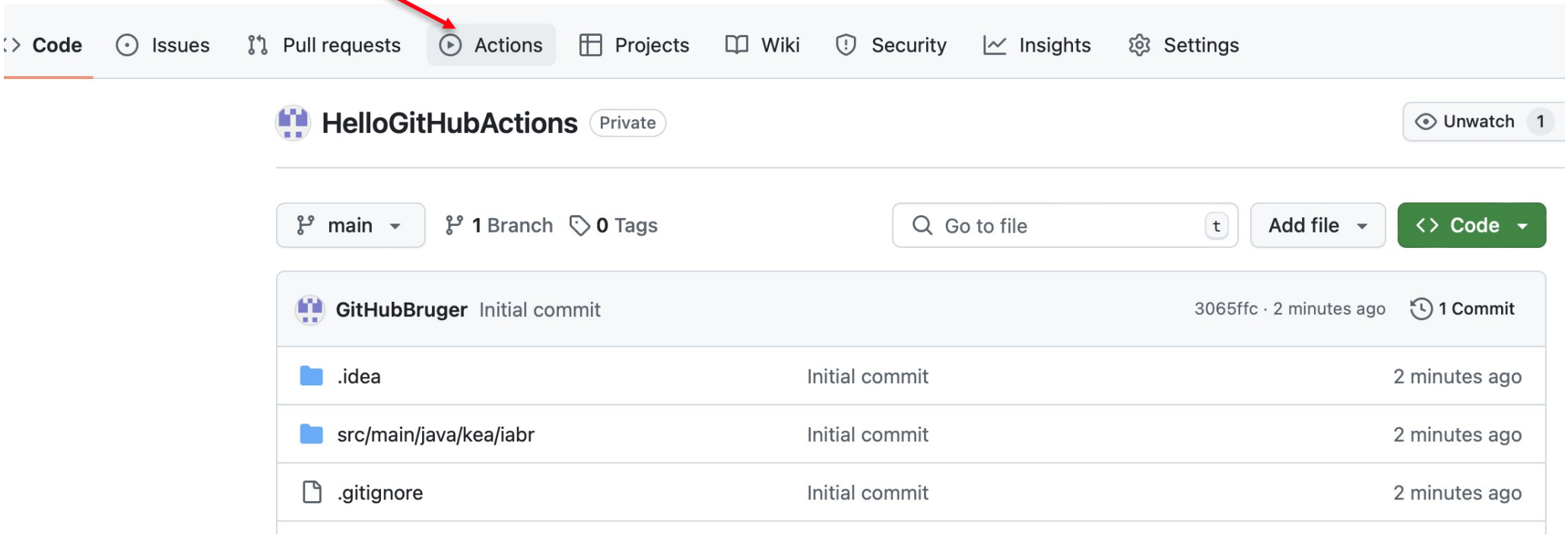
- Set up a CI pipeline using GitHub Actions
- Add a “feature” to the code (following GitHub flow)
- See CI in action
- Add a failing test to see workflow fail and stop the pull request

Demo: CI with GitHub Actions

1. Create a new project and share on GitHub
2. Update the local main branch
3. Create a local feature branch
4. Make commits to the feature branch
5. Update the local main branch
6. Merge the main branch into the feature branch
7. Push the local feature branch to the remote
8. On GitHub create a pull request (**triggering the GitHub Action workflow**) and assign a reviewer
9. Reviewer reviews the changes
10. Merge the changes
11. Delete the feature branch

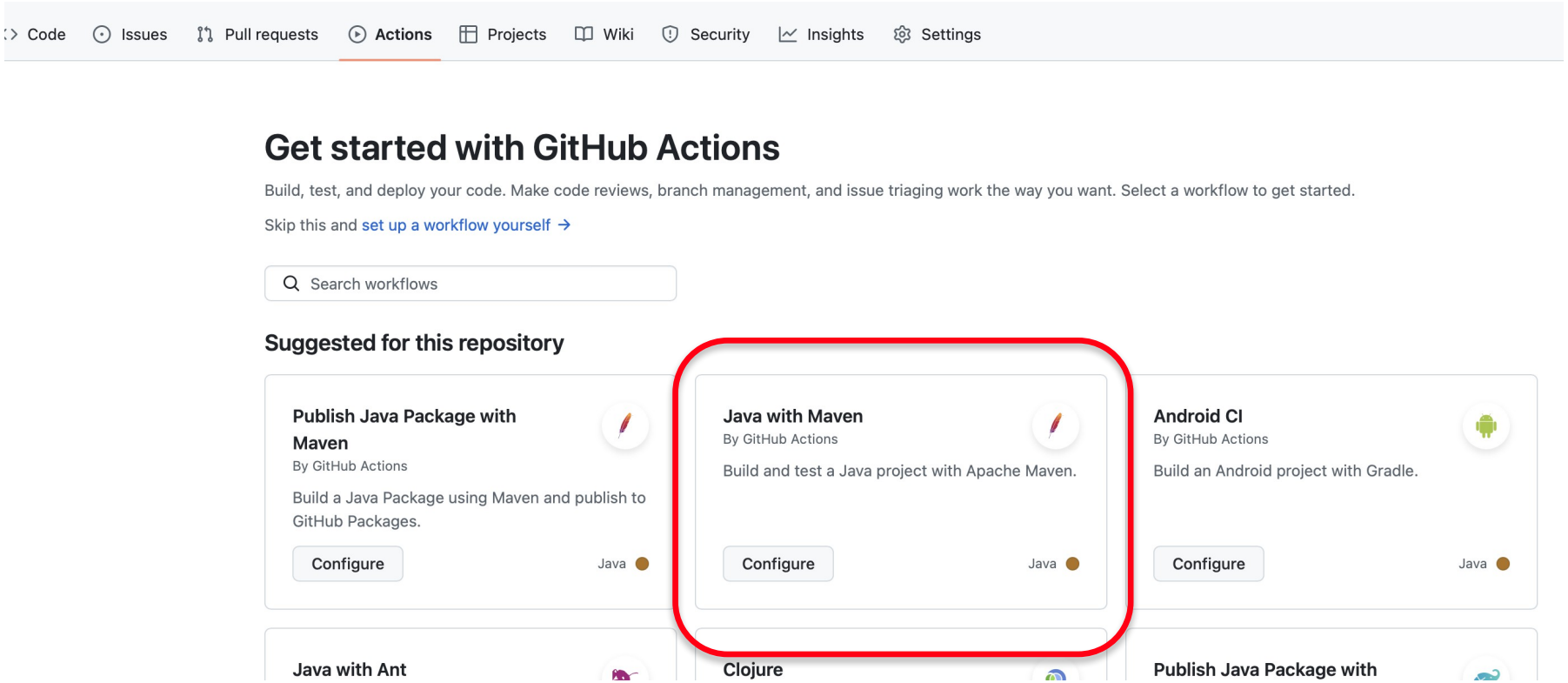
Demo: CI with GitHub Actions

- Add a workflow file (on GitHub)



Demo: CI with GitHub Actions

- Add a workflow file (on GitHub)



Edit | Preview | Spaces | 2 | No wrap

```yaml
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-maven

# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

name: Java CI with Maven

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 17
      uses: actions/setup-java@v4
      with:
        java-version: '17'
        distribution: 'temurin'
        cache: maven
    - name: Build with Maven
      run: mvn -B package --file pom.xml

    # Optional: Uploads the full dependency graph to GitHub to improve the quality of Dependabot alerts this repository can receive
    - name: Update dependency graph
      uses: advanced-security/maven-dependency-submission-action@571e99aab1055c2e71a1e2309b9691de18d6b7d6

```

Demo: CI with GitHub Actions

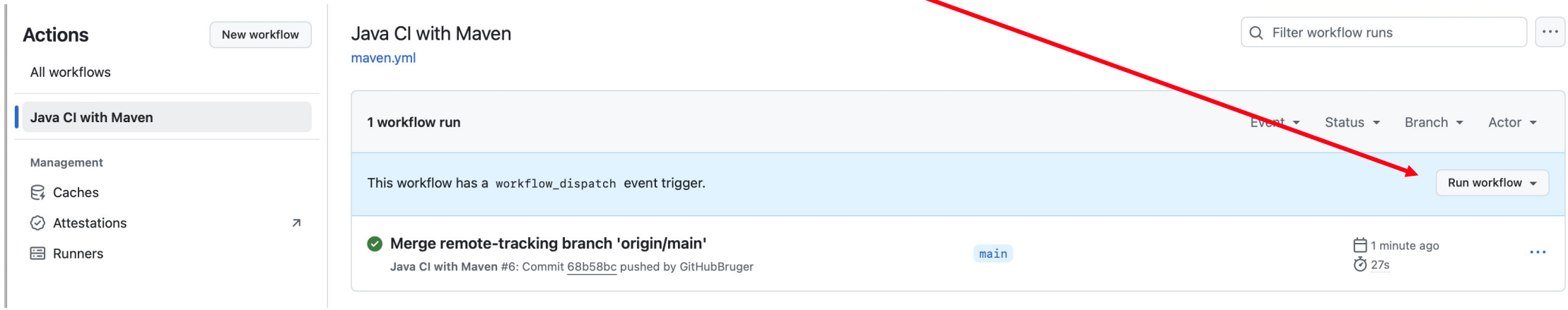
- Edit workflow file maven.yml

maven.yml

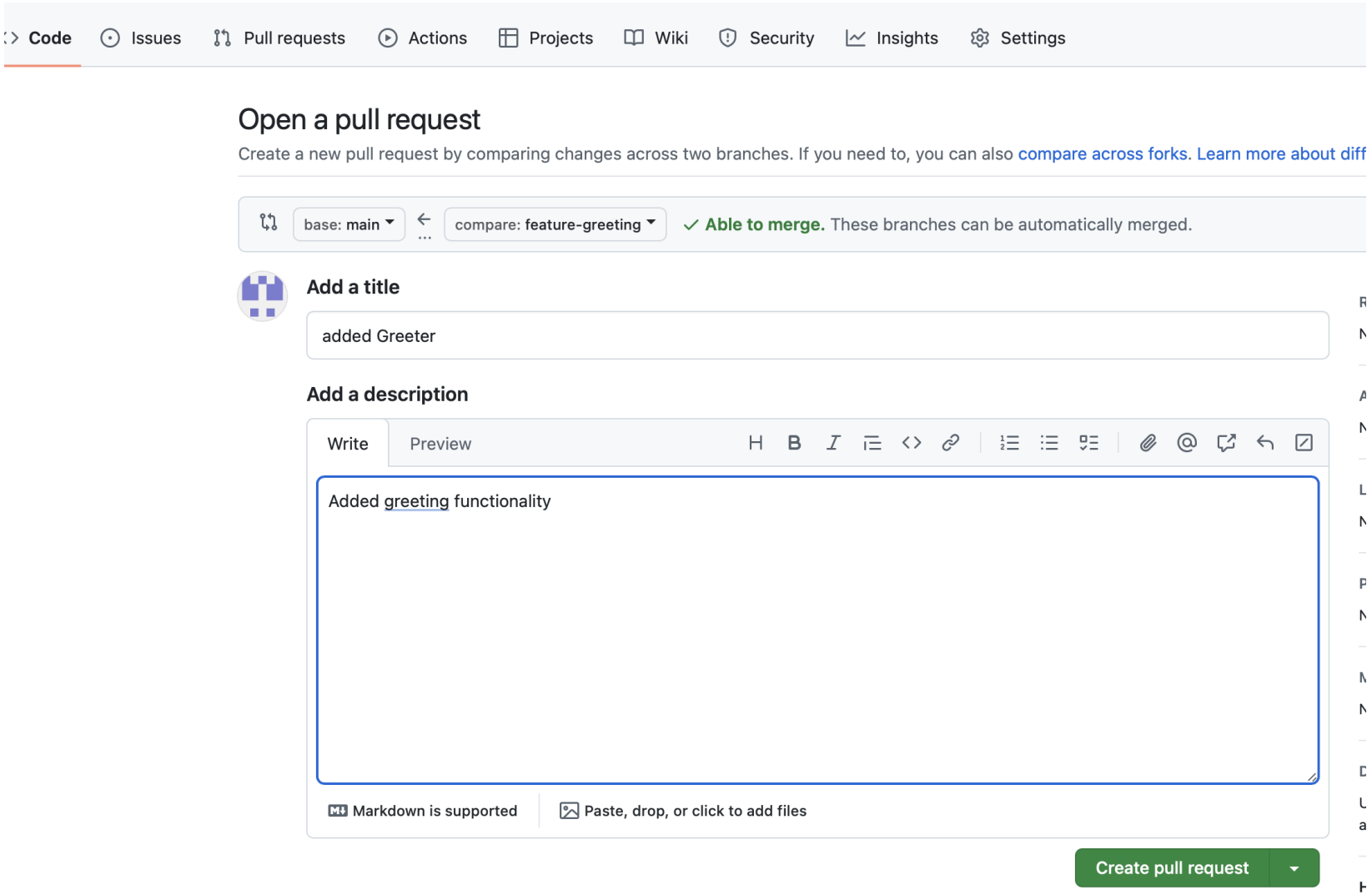
```yaml
name: Java CI with Maven

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

  workflow_dispatch:

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 22
      uses: actions/setup-java@v4.3.0
      with:
        java-version: '22'
        distribution: 'sapmachine'
        cache: maven
    - name: Build with Maven
      run: mvn -B test
```

Demo: CI with GitHub Actions

- `workflow_dispatch`

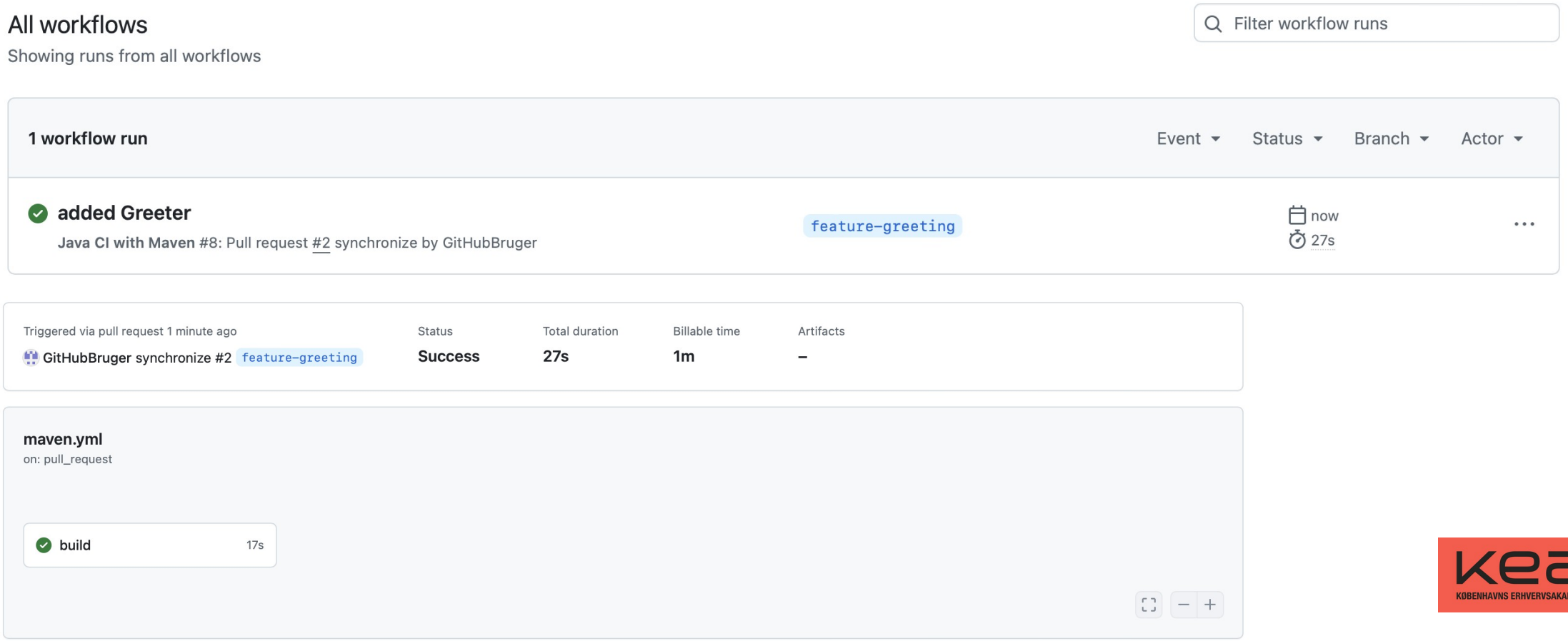


Demo: CI with GitHub Actions



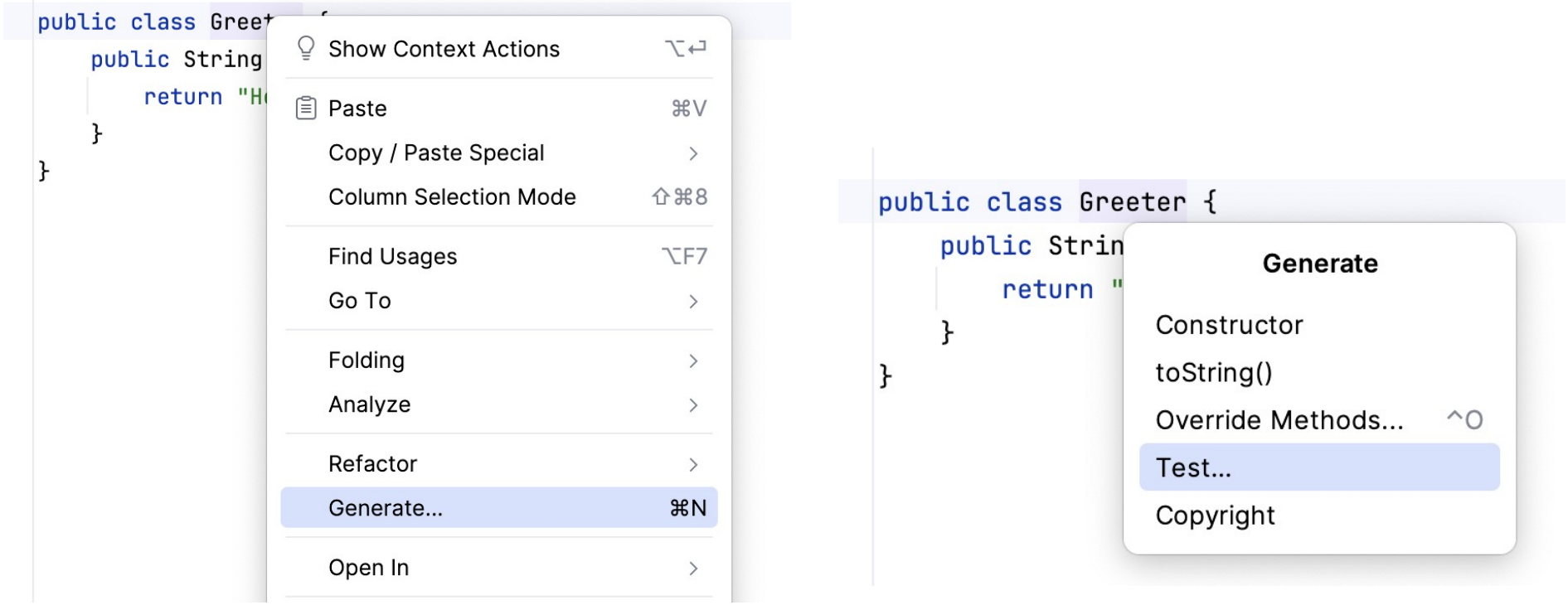
Demo: CI with GitHub Actions

- Workflow run



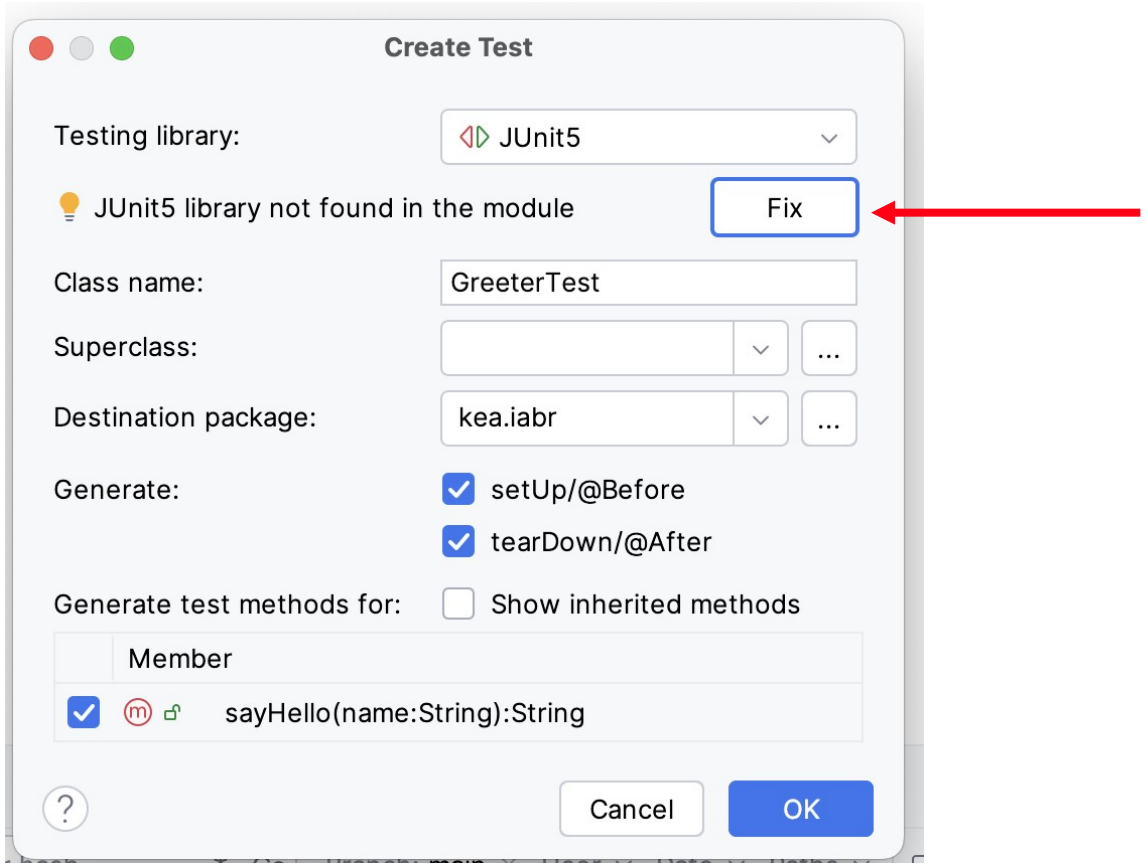
Demo: CI with GitHub Actions

- Adding test methods



Demo: CI with GitHub Actions

- Adding test methods



Edit the pom.xml file

- Add the maven-surefire-plugin (to run the tests)

pom.xml (HelloGitHubActions)

```
2   <project xmlns="http://maven.apache.org/POM/4.0.0"
11      <properties>
14          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15      </properties>
16      <dependencies>
17          <dependency>
18              <groupId>org.junit.jupiter</groupId>
19              <artifactId>junit-jupiter</artifactId>
20              <version>5.8.1</version>
21              <scope>test</scope>
22          </dependency>
23      </dependencies>
24      <build>
25          <plugins>
26              <plugin>
27                  <groupId>org.apache.maven.plugins</groupId>
28                  <artifactId>maven-surefire-plugin</artifactId>
29                  <version>3.2.5</version>
30              </plugin>
31          </plugins>
32      </build>
33  </project>
```

KØBENHAVNS ERHVERVSAKADEMI

Demo: Making the test fail for demo purposes

```java
class GreeterTest {

    @BeforeEach
    void setUp() {
    }

    @AfterEach
    void tearDown() {
    }

    @Test
    void sayHello() {
        Greeter greeter = new Greeter();
        String name = "Test";
        String expectedResult = "Hello " + name;

        String actualResult = greeter.sayHello(name);

        //fail(); //make test fail to demonstrate the workflow failing in CI pipeline
        assertEquals(expectedResult, actualResult);
    }
}
```

Exercise:

- Follow the steps in Hello GitHub Actions Demo v2 to set up a CI pipeline with GitHub Actions.

- (Important to complete as preparation for tomorrow’s lesson)