

=====

Tutor: [Amuthan Sakhtivel](#)

Reference: [Youtube](#)

Course: [Github Actions in Test Automation](#)

=====

1. Document prepared by:

- a. [Rajat Verma](#)
 - i. <https://www.linkedin.com/in/rajat-v-3b0685128/>
 - ii. <https://github.com/rajatt95>
 - iii. <https://rajatt95.github.io/>

2. More documents:

- a. <https://github.com/rajatt95/Documents>

3. Last worked on this Document:

- a. Dec 13, 2022

4. Learnings from Tutor (Code Repository):

- a. This course
 - i. <https://github.com/stars/rajatt95/lists/youtube-tmb-github-actions>
- b. Other course(s):
 - i. <https://github.com/stars/rajatt95/lists/testing-mini-bytes-amuthan>

5. Softwares:

- a. Programming language - Java
- b. IDE - IntelliJ
- c. CI/CD -
 - i. Github Actions
 - 1. Tests Execution
 - a. Web - **Selenoid, Selenium WebDriver**
 - b. Mobile - **Appium**
 - d. Build and Dependency Management Tool - **Maven**

6. Course content:

- a. [Part 1 | Github Actions Tutorial 🔥 Step by Step | Introduction | Github Actions in Test Automation](#)
- b. [Part 2 - Advantages of Github Action over Jenkins | Actions Marketplace | Custom Github Actions](#)
- c. [Part 3 - Workflow Triggers | Different Types of Triggers in Github Actions | Scheduling Workflow |](#)
- d. [Part 4 | Contexts | Different Contexts in Github Actions | How to use Contexts in GitHub Actions |](#)
- e. [Part 5 | Jobs in Github Actions | Containers in Github Actions | Running Jobs in Parallel |](#)
- f. [Part 6 - Steps in Github Actions 🔥 | Step Failure | Timeout for a Step | Github Actions Tutorial |](#)
- g. [Part 7 | Running web test in Github Actions | Setup Selenoid in Github Actions | Passing secrets |](#)
- h. [Part 8 | Reduce workflow time | Caching Maven | Caching Docker Image Layers | Github Actions](#)
- i. [Part 9 | No Cloud Providers Needed 🔥 | Run Android Appium Tests inside Github Runner 🔥](#)
- j. [Part 10 | Run iOS Appium Tests in Github Runner 🔥 | No cloud providers needed |](#)
- k. [Part 11 | Trigger Automated Test workflow in one repo from Development Workflow in another repo](#)

1. Learnings from Course (Youtube - TMB - Github Actions)

a. Links:

i. Github

- 1. <https://github.com/actions>
- 2. <https://github.com/features/actions>
- 3. <https://docs.github.com/en/actions/learn-github-actions/context>

4. Github Marketplace

- a. <https://github.com/marketplace?type=actions>
- b. <https://github.com/marketplace/actions/maven-cache>
- c. <https://github.com/marketplace/actions/android-emulator-runner>
- d. <https://github.com/marketplace/actions/repository-dispatcher>

5. Runner Images

- a. <https://github.com/actions/runner-images>
- b. Machines
 - i. <https://github.com/actions/runner-images/blob/main/images/linux/Ubuntu2204-Readme.md>

- ii. <https://github.com/actions/runner-images/blob/main/images/macos/macos-12-Readme.md>

6. Generate Personal Access Token

- a. <https://github.com/settings/tokens>
-

b. Github Actions Advantages:

- i. No installation needed
 - ii. Can execute jobs in parallel with no efforts
 - iii. Actions will be available for almost all possible setups
 - iv. Access rights to work with Github Actions
-

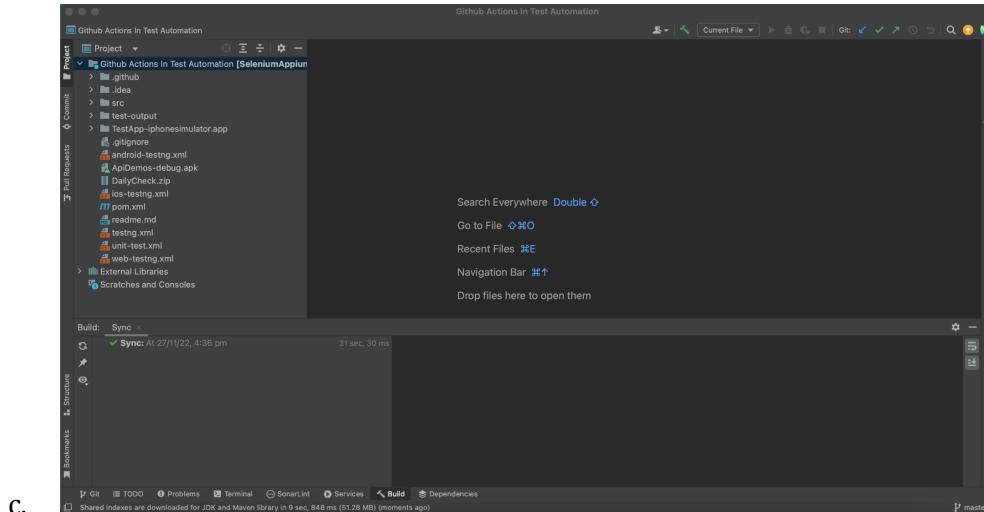
c. Terminologies

- i. Workflows
 - 1. Execution
 - a. Manual
 - b. Automated
 - i. Push
 - ii. Pull_Request
 - iii. Schedule
 - ii. Actions
 - iii. Workflow Triggers
 - iv. Jobs
 - v. Steps
 - vi. Parameters passing using secrets
 - vii. Contexts
 - viii. Job dependency
 - ix. Caching Maven | Caching Docker Image Layers
 - x. Tests
 - 1. Unit
 - 2. Web
 - 3. Mobile (Android, iOS)
 - xi. Trigger Automated Test workflow in one repo from Development Workflow in another repo
-

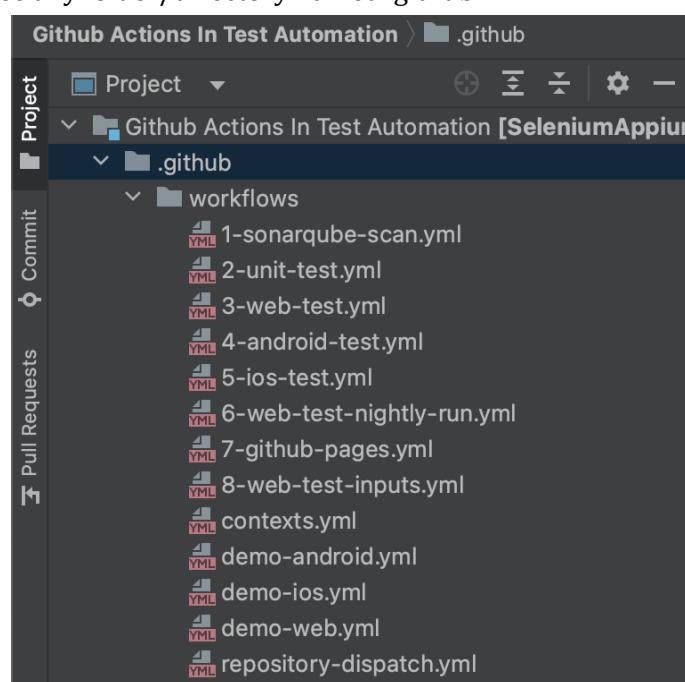
=====1_Part 1 | Github Actions Tutorial 🔥 Step by Step | Introduction | Github Actions in Test Automation=====

1. Github link for the Tests

- <https://github.com/rajatt95/SeleniumAppiumTestsInGitHubRunners>
- Clone/Pull/Download this project and import in an IDE

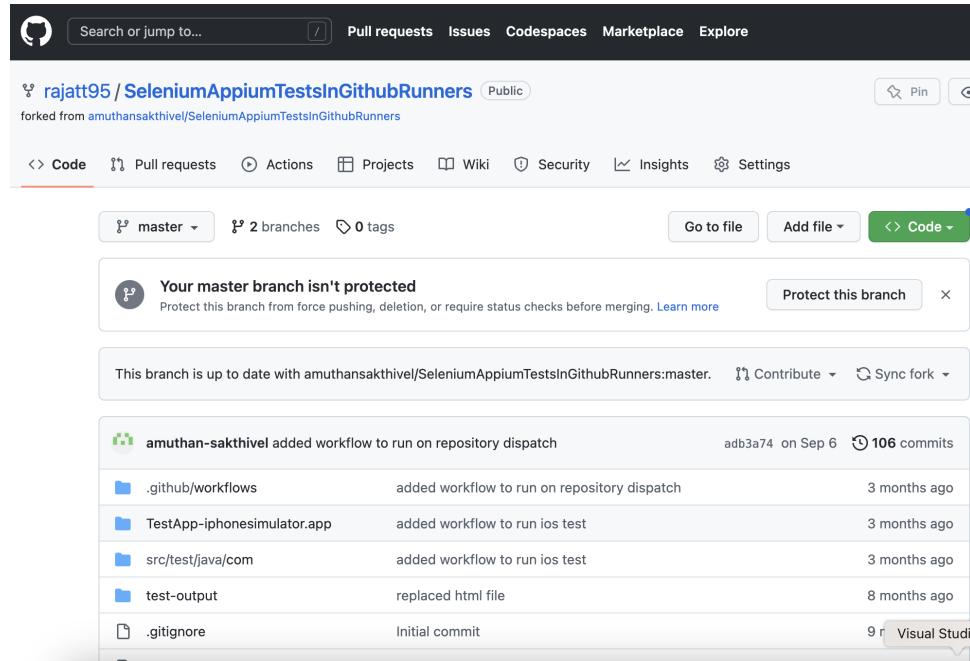


- You'll see any folder/directory named .github



- Workflow in Github Actions is similar to Pipelines/Jobs in Jenkins.

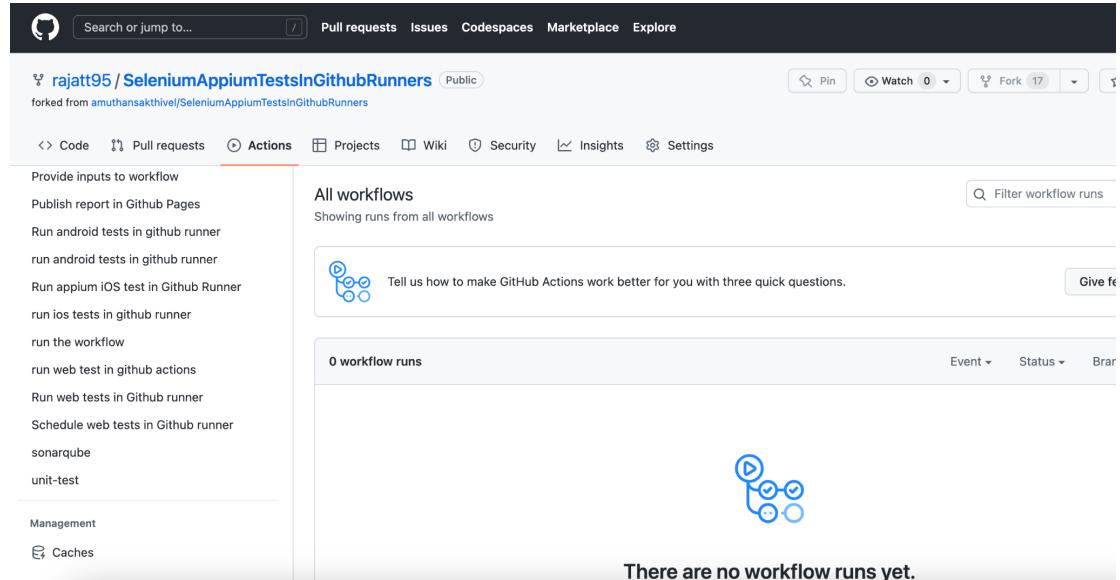
1. Go to <https://github.com/rajatt95/SeleniumAppiumTestsInGithubRunners>



The screenshot shows the GitHub repository page for 'SeleniumAppiumTestsInGithubRunners'. The repository is public and forked from 'amuthansakthivel/SeleniumAppiumTestsInGithubRunners'. The 'Code' tab is selected, showing the master branch. A message states 'Your master branch isn't protected'. Below this, it says 'This branch is up to date with amuthansakthivel/SeleniumAppiumTestsInGithubRunners:master'. A list of commits is shown, all made by 'amuthan-sakthivel' on Sep 6, 2023, with 106 commits. The commits include adding workflows for repository dispatch, .github/workflows, TestApp-iphonesimulator.app, src/test/java/com, test-output, and .gitignore.

a.

2. Click on the Actions (Tab #3)

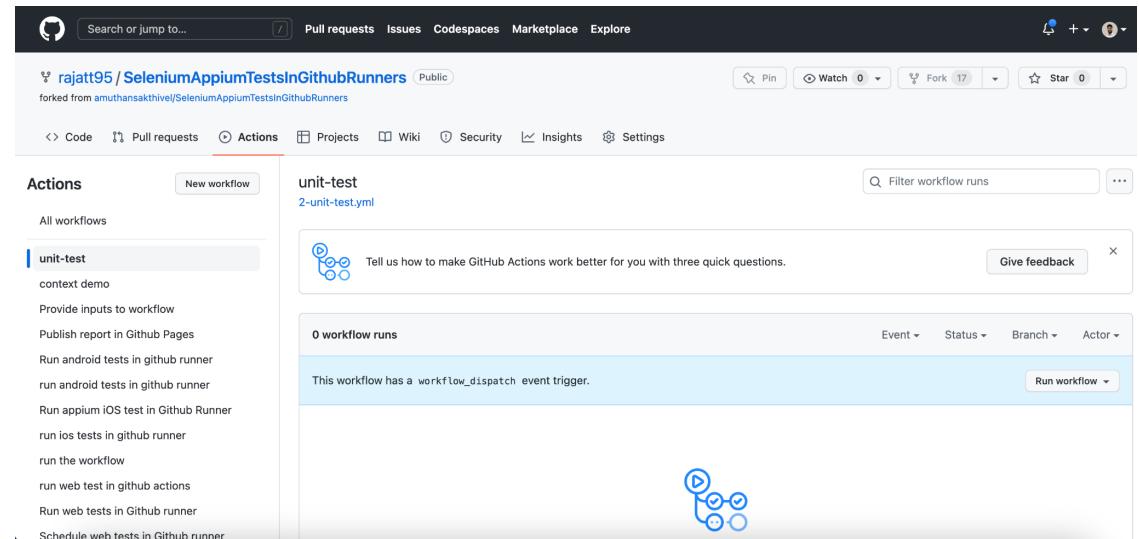


The screenshot shows the GitHub Actions page for the same repository. The 'Actions' tab is selected. On the left, a sidebar lists various workflow actions such as 'Provide inputs to workflow', 'Publish report in Github Pages', 'Run android tests in github runner', etc. The main area shows 'All workflows' with a note 'Showing runs from all workflows'. It includes a feedback section 'Tell us how to make GitHub Actions work better for you with three quick questions.' and a summary '0 workflow runs'. A message at the bottom states 'There are no workflow runs yet.'

a.

i. We'll be able to see all the Workflows

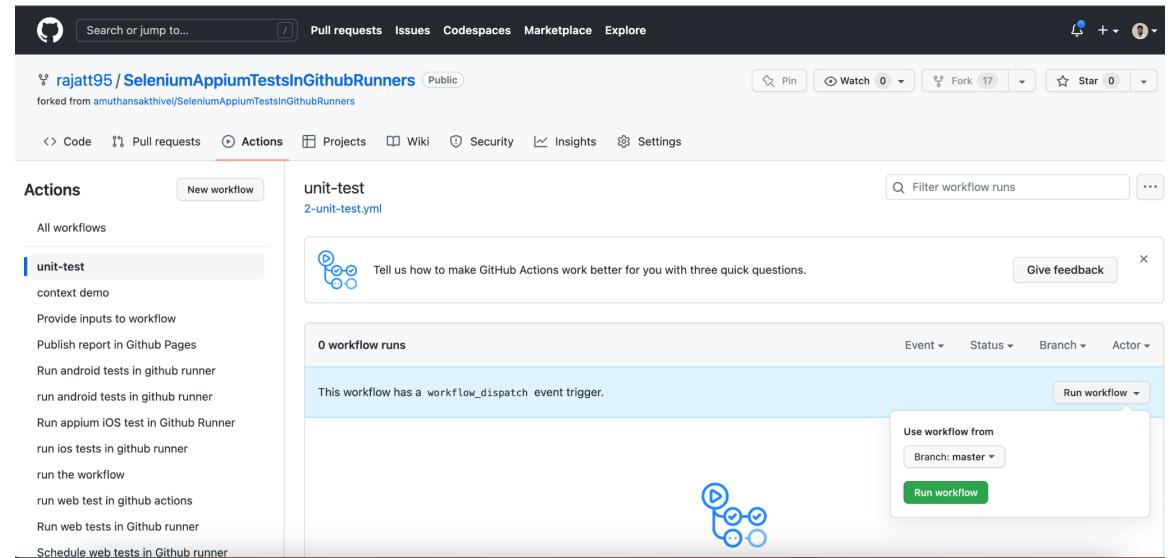
3. Click on the unit-test



The screenshot shows the GitHub Actions interface for a repository named 'SeleniumAppiumTestsInGithubRunners'. The 'Actions' tab is selected. On the left, there's a sidebar with a list of actions, including 'unit-test', 'context demo', and various test-related options like 'Run android tests in github runner' and 'Run appium iOS test in Github Runner'. The main area displays the 'unit-test' workflow, which has a single configuration file named '2-unit-test.yml'. A message box asks for feedback on how to make GitHub Actions better. Below it, a section titled '0 workflow runs' indicates that the workflow has a 'workflow_dispatch' event trigger. A large blue GitHub Actions icon is centered at the bottom.

- a. i. We can see all details for Workflow - **unit-test**.
-

4. Click on the Run Workflow button



This screenshot is similar to the previous one but shows a modal dialog box in the bottom right corner. The dialog is titled 'Run workflow from' and has a dropdown menu set to 'Branch: master'. It contains a prominent green 'Run workflow' button. The rest of the page is identical to the first screenshot, showing the 'unit-test' workflow details.

The screenshot shows the GitHub Actions page for the repository "rajatt95 / SeleniumAppiumTestsInGithubRunners". The "Actions" tab is selected. On the left, there's a sidebar with "All workflows" and a list of "unit-test" workflows. One workflow, "unit-test #1", is highlighted with a green checkmark. The main area displays the workflow configuration file "2-unit-test.yml" and its execution details. It shows 1 workflow run with an event trigger of "workflow_dispatch", status "Queued", and last updated "now". A feedback button and a "Give feedback" link are also present.

b.

5. Click on **unit-test** to see the Logs

The screenshot shows the GitHub Actions job logs for the "run-unit-test" job of the "unit-test #1" workflow. The job summary indicates it succeeded 18 seconds ago in 22s. The log details show the following steps:

Step	Description	Time
1	Set up job	1s
2	Run actions/checkout@v2	1s
3	Set up JDK 11	5s
4	Run unit test	12s
5	Post Set up JDK 11	0s
6	Post Run actions/checkout@v2	0s
7	Complete job	0s

a.

6. We can expand these links to see the details logs

a.

```

run-unit-test
succeeded 1 minute ago in 22s

> ⚡ Set up job
  ✓ Run actions/checkout@v2
    1 ► Run actions/checkout@v2
    12 Syncing repository: rajatt95/SeleniumAppiumTestsInGithubRunners
    13 ► Getting Git version info
    17 Temporarily overriding HOME='/home/runner/work/_temp/fe553256-ffa4-45a4-8589-63088105e3ef' before making global git config changes
    18 Adding repository directory to the temporary git global config as a safe directory
    19 /usr/bin/git config --global --add safe.directory
      '/home/runner/work/SeleniumAppiumTestsInGithubRunners/SeleniumAppiumTestsInGithubRunners'
    20 Deleting the contents of '/home/runner/work/SeleniumAppiumTestsInGithubRunners/SeleniumAppiumTestsInGithubRunners'
    21 ► Initializing the repository
    35 ► Disabling automatic garbage collection
    37 ► Setting up auth
    43 ► Fetching the repository
    298 ► Determining the checkout info
    299 ► Checking out the ref
    303 /usr/bin/git log -1 --format='%H'
    304 'adb3a7495a8fc83d5400612ac723b6d0805e3d671'

  ✓ Set up JDK 11
    1 ► Run actions/setup-java@v1

```

b.

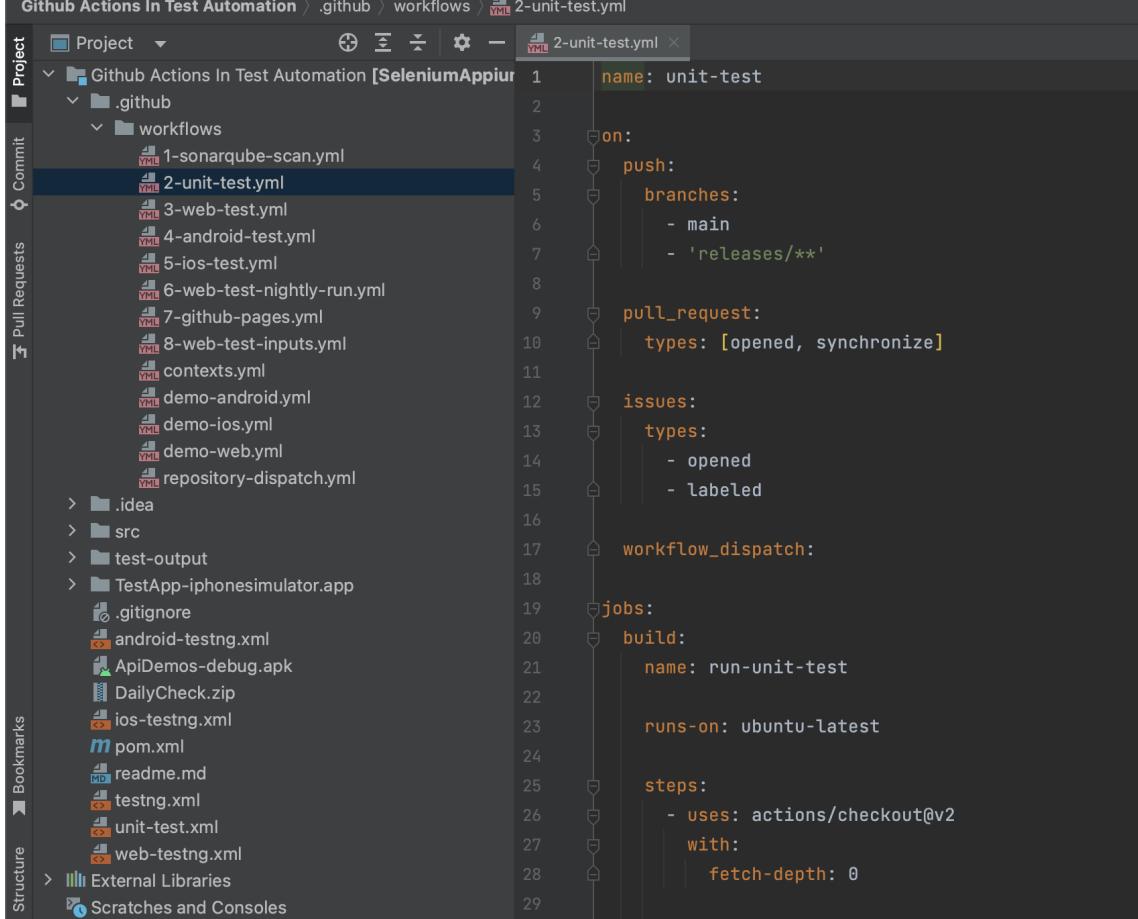
```

run-unit-test
succeeded 3 minutes ago in 22s

> ⚡ Run unit test
  ✓ Run unit test
    9327 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-testng-utils/3.0.0-M5/surefire-testng-utils-3.0.0-M5.jar (14 kB at 616 kB/s)
    9328 [INFO] -----
    9329 [INFO] -----
    9330 [INFO] T E S T S
    9331 [INFO] -----
    9332 [INFO] Running TestSuite
    9333 PASSED: testUnitTestInGithubRunner
    9334 =====
    9335 =====
    9336 /Users/amuthansakthivel/IdeaProjects/SeleniumAppiumTestsInGithubRunners
    9337 Tests run: 1, Failures: 0, Skips: 0
    9338 =====
    9339
    9340 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.361 s - in TestSuite
    9341 [INFO]
    9342 [INFO] Results:
    9343 [INFO]
    9344 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
    9345 [INFO]
    9346 [INFO] -----
    9347 [INFO] BUILD SUCCESS
    9348 [INFO] -----
    9349 [INFO] Total time: 11.097 s
    9350 [INFO] Finished at: 2022-11-27T11:17:31Z
    9351 [INFO] -----

```

-
1. Open the project in IDE
 2. We have a Workflow created named **2-unit-test.yml**



```

Github Actions In Test Automation > .github > workflows > 2-unit-test.yml
Project 2-unit-test.yml
  name: unit-test
  on:
    push:
      branches:
        - main
        - 'releases/**'
    pull_request:
      types: [opened, synchronize]
    issues:
      types:
        - opened
        - labeled
    workflow_dispatch:
  jobs:
    build:
      name: run-unit-test
      runs-on: ubuntu-latest
      steps:
        - uses: actions/checkout@v2
          with:
            fetch-depth: 0

```

- a.
- i. This Workflow has only 1 job
 1. But, we can have multiple jobs for 1 Workflow
 2. When we have **multiple jobs**,
 - a. All those jobs run **parallel**.
 - b. Customization
 - i. Possible to do - One job is dependent on another.
-

b. Keywords

i. **on** - When I want to trigger this Workflow

```
2-unit-test.yml ×
1   name: unit-test
2
3   on:
4     push:
5       branches:
6         - main
7         - 'releases/**'
```

1. push
 - a. branches
2. pull_request
 - a. types
3. workflow_dispatch - I want an option to trigger this Workflow manually as well

ii. **jobs**

```
jobs:
  build:
    name: run-unit-test
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0

      - name: Set up JDK 11
        uses: actions/setup-java@v1
        with:
          java-version: 11

      - name: Run unit test
        run: mvn clean test -Punit-test
```

1. build

```
jobs:
  build:
    name: run-unit-test
    runs-on: ubuntu-latest
```

- a.

- b. name

- c. **runs-on** - Where I want to run/execute the job (Github Actions support WIN/MAC/Linux)

- d. Steps

```
steps:
  - uses: actions/checkout@v2
    with:
      fetch-depth: 0

  - name: Set up JDK 11
    uses: actions/setup-java@v1
    with:
      java-version: 11

  - name: Run unit test
    run: mvn clean test -Punit-test
```

- i.
- ii. **name** - name of step
- iii. **uses** - Pull the project in the machine selected (Here, we have taken ubuntu-latest)
- iv. with
 - 1. Fetch-depth
- v. run
 - 1. Which command to execute

1. We'll cover these **Trigger events** in detail later.

- 1. For execution, we need
 - a. Machine - On which tests will execute
 - b. Project - which will have the tests (Web, Mobile, API)
 - c. Softwares setup - Java

1. In Workflow (**unit-test**)

- a. We are executing the command -

```
- name: Run unit test
  run: mvn clean test -Punit-test
```

- i.
 - 1. -P is for Profile.
 - 2. Tutor has used the Maven Profile
 - a. In pom.xml, he has maintained a Maven Profile named 'unit-test'.

```
71 <profiles>
72   <profile>
73     <id>unit-test</id>
74     <build>
75       <pluginManagement>
76         <plugins>
77           <plugin>
78             <groupId>org.apache.maven.plugins</groupId>
79             <artifactId>maven-surefire-plugin</artifactId>
80             <version>3.0.0-M5</version>
81             <configuration>
82               <suiteXmlFiles>
83                 <suiteXmlFile>unit-test.xml</suiteXmlFile>
84               </suiteXmlFiles>
85             </configuration>
86           </plugin>
87         </plugins>
88       </pluginManagement>
89     </build>
90   </profile>
```

b.

-
1. Till now, we have only executed 1 Unit test, but, we want to execute Tests created for
 - a. Web application using Selenium WebDriver
 - b. Mobile application using Appium
-

2. For Web

```
1 name: Run web tests in Github runner
2
3 on:
4   push:
5     branches: [ main ]
6
7 workflow_dispatch:
8
9 jobs:
10   build:
11     runs-on: ubuntu-latest
12
13 steps:
14   - name: Start Selenoid server
15     uses: n-ton4/selenoid-github-action@master
16     id: start-selenoid
17     continue-on-error: false
18     with:
19       version: 1.10.1
20       args: -limit 10
21       browsers: chrome
22       last-versions: 1
23
24   - name: checkout
25     uses: actions/checkout@v2
26
27   - name: Run the tests
28     run:
29       mvn clean test -Pweb
```

a.

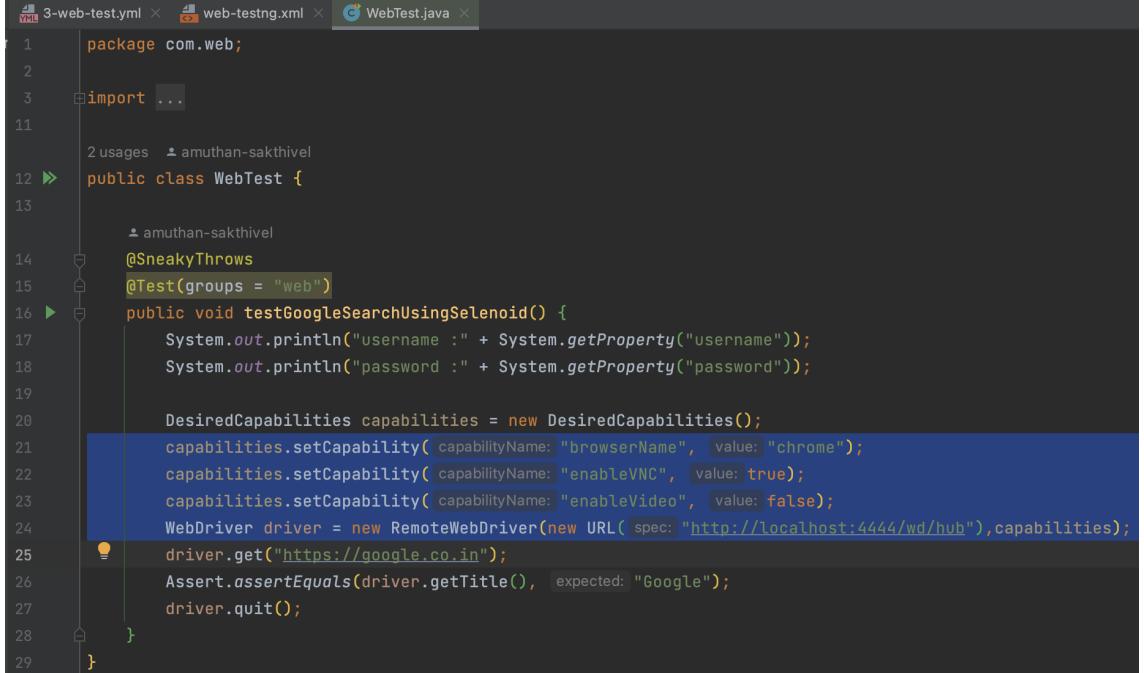
- i. This machine 'ubuntu-latest' already has a set of software installed
 - 1. Browser - Chrome
 - 2. Android Emulators
- ii. Chrome

```
steps:  
  - name: Start Selenoid server  
    uses: n-ton4/selenoid-github-action@master  
    id: start-selenoid  
    continue-on-error: false  
  with:  
    version: 1.10.1  
    args: -limit 10  
    browsers: chrome  
    last-versions: 1  
1. 
```

- a. Allowing 10 instances of Browsers for parallel execution
- iii. One thing to note, we do not have actions keyword here

```
steps:  
  - name: Start Selenoid server  
    uses: n-ton4/selenoid-github-action@master  
    id: start-selenoid  
    continue-on-error: false  
1.  
a. What does this n-ton4 mean?  
b. This means this is not developed by Github. Someone has developed this Utility and pushed it to the Github marketplace.
```

1. Test case written for Web

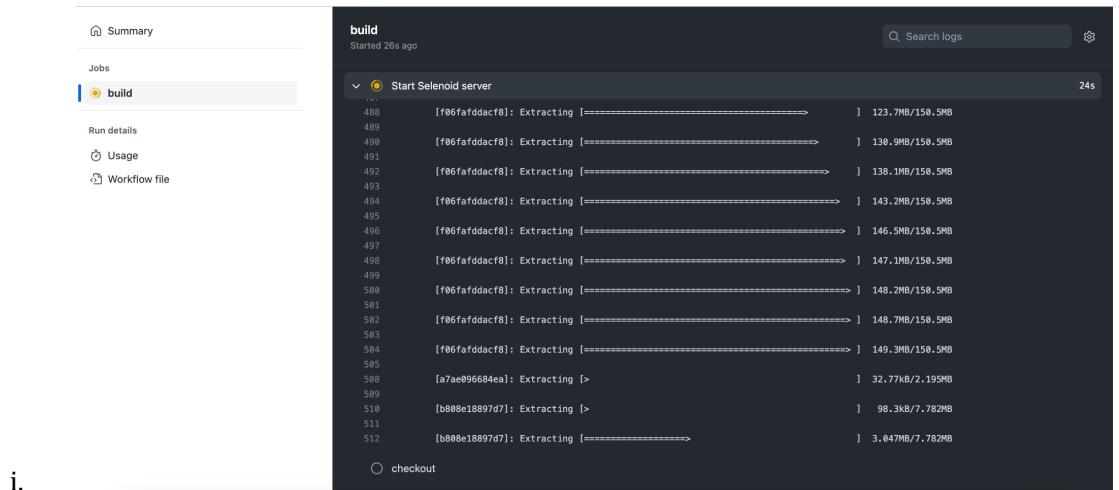


```
1 package com.web;
2
3 import ...
11
12 >     2 usages  amuthan-sakthivel
13
14     public class WebTest {
15
16         amuthan-sakthivel
17         @SneakyThrows
18         @Test(groups = "web")
19
20         public void testGoogleSearchUsingSelenoid() {
21             System.out.println("username :" + System.getProperty("username"));
22             System.out.println("password :" + System.getProperty("password"));
23
24             DesiredCapabilities capabilities = new DesiredCapabilities();
25             capabilities.setCapability(capabilityName: "browserName", value: "chrome");
26             capabilities.setCapability(capabilityName: "enableVNC", value: true);
27             capabilities.setCapability(capabilityName: "enableVideo", value: false);
28             WebDriver driver = new RemoteWebDriver(new URL(spec: "http://localhost:4444/wd/hub"), capabilities);
29             driver.get("https://google.co.in");
30             Assert.assertEquals(driver.getTitle(), expected: "Google");
31             driver.quit();
32         }
33     }
34 }
```

a.

2. To execute this Workflow manually,

- Go to <https://github.com/rajatt95/SeleniumAppiumTestsInGithubRunners>
- Click on **Actions**
- Click on **Run Web Tests in Github Runner**



i.

1. Solenoid

- a. It is a customized Selenium Hub Selenium Grid setup
 - i. Very optimized
-

- 1. Till now, we have used the machine that Github Action has provided us
 - a. That is also possible if you want to set up your custom machine.
-

=====2_Part 2 - Advantages of Github Action over Jenkins | Actions Marketplace | Custom Github Actions=====

1. Advantages:

a. No installation needed

- i. Earlier, we were working with Jenkins
 - 1. We had to maintain a machine (that was working as a Host)
 - 2. Software setup in that machine
 - a. Java path
 - b. Maven path
 - c. Download Plugins
- ii. Now, with Github Actions
 - 1. Everything is set up in cloud
 - 2. Ready-to-use

b. Can execute jobs in parallel with no efforts

- i. Add one job in the .yaml file and that's all.

c. Actions will be available for almost all possible setups

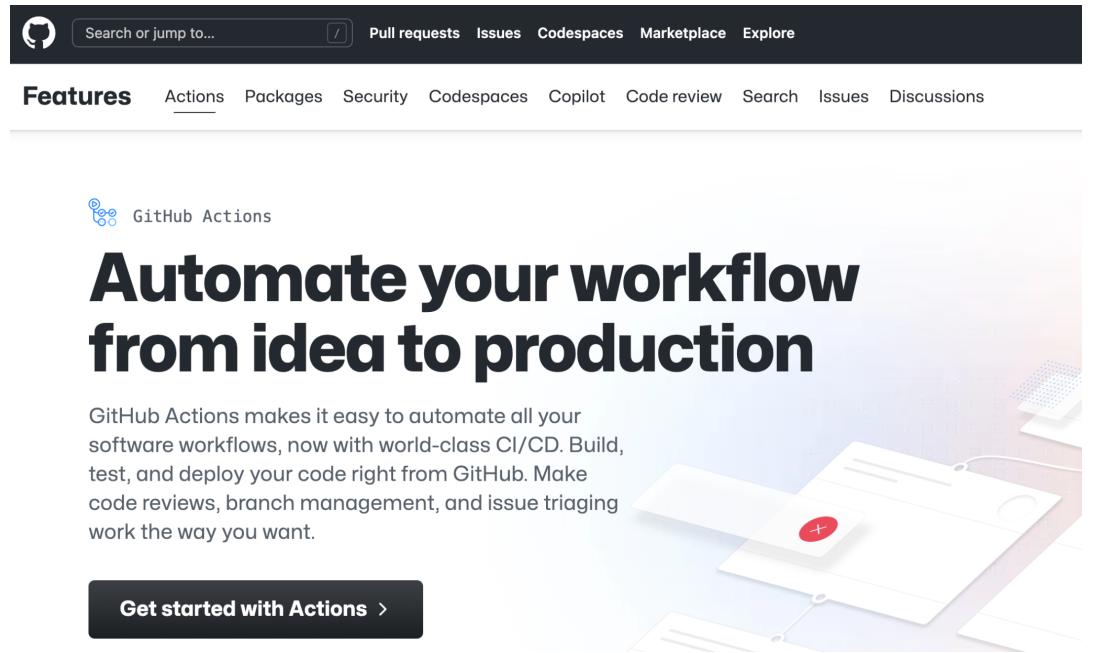
- i. Setup is ready to use - Calling it **Actions**
- ii. Actions are pushed into Github marketplace
 - 1. Re-Usable
 - 2. People have contributed and pushed

d. Access rights to work with Github Actions

- i. Earlier, Jenkins -> Access
 - 1. Test Automation Team (Very rare)
 - ii. Now, we maintain a code repository for test case
 - 1. We have the access rights to perform different operations
-

1. Github Actions

- a. <https://github.com/features/actions>

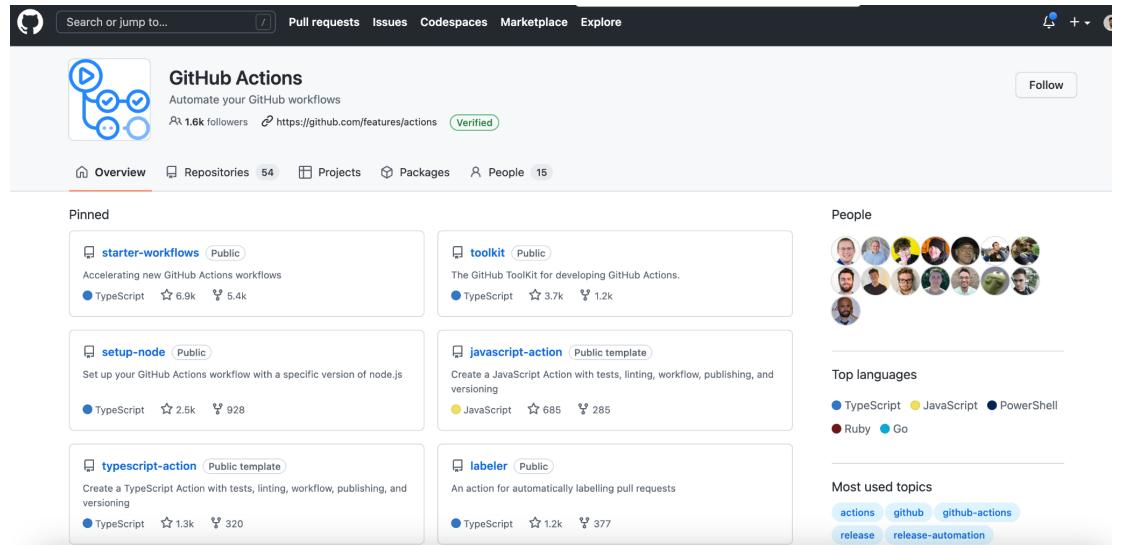


GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

[Get started with Actions >](#)

i.

- b. <https://github.com/actions>



GitHub Actions
Automate your GitHub workflows
1.6k followers <https://github.com/features/actions> Verified

Overview Repositories 54 Projects Packages People 15

Pinned

- starter-workflows Public Accelerating new GitHub Actions workflows TypeScript 6.9k 5.4k
- setup-node Public Set up your GitHub Actions workflow with a specific version of node.js TypeScript 2.6k 928
- typescript-action Public template Create a TypeScript Action with tests, linting, workflow, publishing, and versioning TypeScript 1.3k 320
- toolkit Public The GitHub ToolKit for developing GitHub Actions. TypeScript 3.7k 1.2k
- javascript-action Public template Create a JavaScript Action with tests, linting, workflow, publishing, and versioning JavaScript 685 285
- labeler Public An action for automatically labelling pull requests TypeScript 1.2k 377

People

Top languages

Most used topics

i.

- c. To set up Java

- i. <https://github.com/actions/setup-java>
- ii. This is an official action (Developed by Github Team)
 1. Actions present under

- a. <https://github.com/actions>

```
 README.md
```

Basic Configuration

Eclipse Temurin

```
steps:
- uses: actions/checkout@v3
- uses: actions/setup-java@v3
  with:
    distribution: 'temurin' # See 'Supported distributions' for available options
    java-version: '17'
- run: java HelloWorldApp.java
```

Azul Zulu OpenJDK

```
steps:
- uses: actions/checkout@v3
- uses: actions/setup-java@v3
  with:
    distribution: 'zulu' # See 'Supported distributions' for available options
    java-version: '17'
- run: java HelloWorldApp.java
```

2.

3.

4. These actions

1. Github Marketplace

- <https://github.com/marketplace?type=>
-

=====3_Part 3 - Workflow Triggers | Different Types of Triggers in Github Actions | Scheduling Workflow |=====

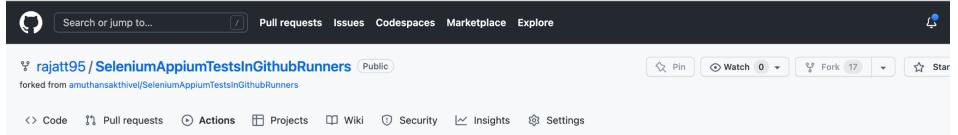
1. Workflow Triggers

- a. Workflow Triggers are events that cause Workflows to run
- b. There are multiple ways to trigger a Workflow
- c. Triggers
 - i. Example
 1. Execute test cases daily
 2. Execute on code push
 3. Execute on PR raised

1. To create Workflow

- a. Go to the code repo
- b. Click on Actions
- c. Click on New workflow
 - i. <https://github.com/rajatt95/SeleniumAppiumTestsInGithubRunners/actions/new>

1. There are many Templates available as well



Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Search workflows

Suggested for this repository



- 2.

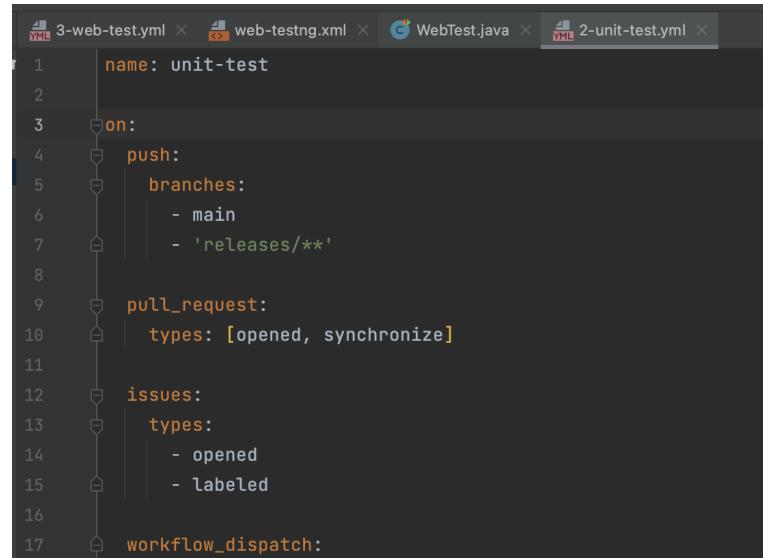
- ii. Click on Setup a Workflow yourself

1. These workflows will be saved inside the **.github folder**
2. You can give names accordingly

a. Execution-daily.yml

- b. # is used for comment in .yml file

1. Unit-test.yml file



```
1 name: unit-test
2
3 on:
4   push:
5     branches:
6       - main
7       - 'releases/**'
8
9   pull_request:
10    types: [opened, synchronize]
11
12  issues:
13    types:
14      - opened
15      - labeled
16
17  workflow_dispatch:
```

- a. Here, the Triggers are

i. Push

1. Trigger the job when there is a code push on the main branch or "releases/**"

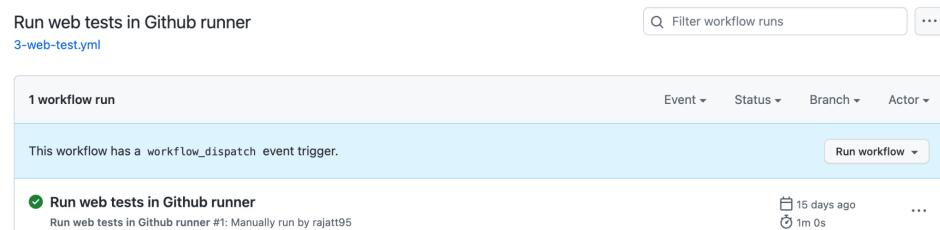
ii. Pull Request

1. Types
 - a. Opened
 - b. Synchronize

iii. Workflow Dispatch

1. For manual trigger -

- a. We'll be able to trigger the Workflow manually from Github repository



Run web tests in Github runner

3-web-test.yml

1 workflow run

This workflow has a `workflow_dispatch` event trigger.

Run workflow

Run web tests in Github runner

Run web tests in Github runner #1: Manually run by rajatt95

15 days ago

1m 0s

b.

- iv. There are many other ways as well for Triggers

1. branches-ignore

1. Schedule over a period of time

a.

```
6-web-test-nightly-run.yml ×
1   name: Schedule web tests in Github runner
2   #UTC
3   on:
4     schedule:
5       - cron: '0 22 * * *'
6   jobs:
7     build:
8       runs-on: ubuntu-latest
```

- i.
- Every night 10 PM
 - 1. **schedule** is the keyword
 - 2. The default time followed is **UTC**
-

1. Parameters passing

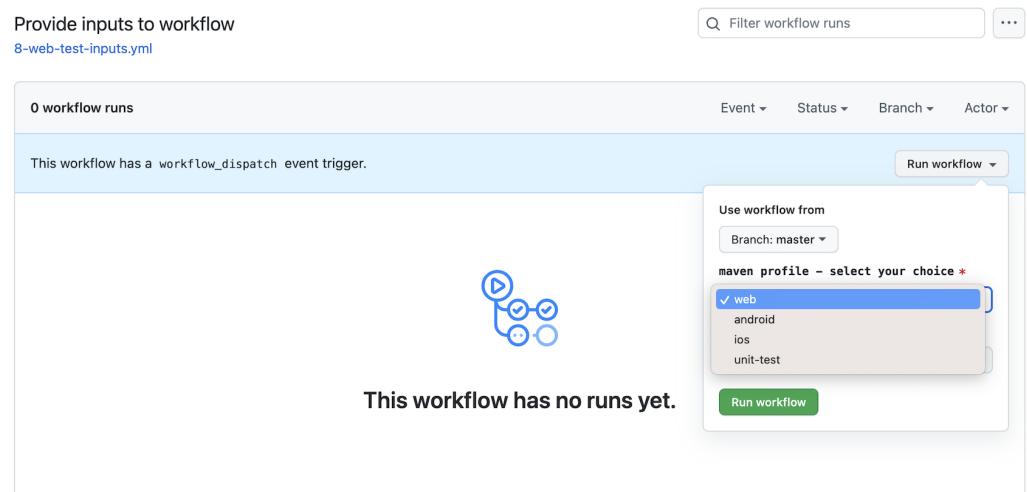
a. Jenkins

- i. We had the option to pass the parameters value

b. Github Actions

```
6-web-test-nightly-run.yml × 8-web-test-inputs.yml × 3-web-test.yml ×
1   name: Provide inputs to workflow
2
3   on:
4     workflow_dispatch:
5       inputs:
6         mavenProfile:
7           description: 'maven profile - select your choice'
8           options:
9             - 'web'
10            - 'android'
11            - 'ios'
12            - 'unit-test'
13           required: true
14           default: 'web'
15           type: choice
16         remoteURL:
17           description: 'selenoid url if hosted outside'
18           required: true
19           default: 'http://localhost:4444/wd/hub'
```

- i.
- 1. When the User tries to execute the Workflow manually (Line no. 5 to 19 is under **workflow_dispatch**), he/she has to provide the inputs first
 - a. **required: true** (It is a mandatory parameter)
 - b. **default: web**



2.

```

1 20
2 21   jobs:
3 22     build:
4 23       runs-on: ubuntu-latest
5 24
6 25     steps:
7 26       - name: Start Selenoid server
8 27         uses: n-ton4/selenoid-github-action@master
9 28         id: start-selenoid
10 29         continue-on-error: false
11 30         with:
12 31           version: 1.10.1
13 32           args: -limit 10
14 33           browsers: chrome
15 34           last-versions: 1
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

```

ii.

1. **`$(inputs.mavenProfile)`**
 - a. This is how we use variables in Github Workflows
2. **mavenProfile**
 - a. Customized

b.

```
1  name: Provide inputs to workflow
2
3  on:
4    workflow_dispatch:
5      inputs:
6        mavenProfile:
7          description: 'maven profile - select your choice'
8          options:
9            - 'web'
10           - 'android'
11           - 'ios'
12           - 'unit-test'
```

=====4_Part 4 | Contexts | Different Contexts in Github Actions | How to use Contexts in GitHub Actions |=====

1. Contexts

- a. Way to access information about
 - i. The workflow runs
 1. Who triggered it?
 2. When triggered?
 3. What is the Event that triggered it?
 - ii. Runner environments
 - iii. Jobs
 - iv. Steps
-

1. <https://docs.github.com/en/actions/learn-github-actions/context>

Context name	Type	Description
github	object	Information about the workflow run. For more information, see github context .
env	object	Contains environment variables set in a workflow, job, or step. For more information, see env context .
job	object	Information about the currently running job. For more information, see job context .
jobs	object	For reusable workflows only, contains outputs of jobs from the reusable workflow. For more information, see jobs context .
steps	object	Information about the steps that have been run in the current job. For more information, see steps context .
runner	object	Information about the runner that is running the current job. For more information, see runner context .
secrets	object	Contains the names and values of secrets that are available to a workflow run. For more information, see secrets context .
strategy	object	Information about the matrix execution strategy for the current job. For more information, see strategy context .

In this article

- About contexts
 - Determining when to use contexts
 - Context availability
 - Example: printing context information to the log
- github context
 - Example contents of the github context
 - Example usage of the github context
- env context
 - Example contents of the env context
 - Example usage of the env context
- job context
 - Example contents of the job context
 - Example usage of the job context
- jobs context

b. Context - **Github**

github context

The `github` context contains information about the workflow run and the event that triggered the run. You can also read most of the `github` context data in environment variables. For more information about environment variables, see [Using environment variables](#).

Warning: When using the whole `github` context, be mindful that it includes sensitive information such as `github.token`. GitHub masks secrets when they are printed to the console, but you should be cautious when exporting or printing the context.

Warning: When creating workflows and actions, you should always consider whether your code might execute untrusted input from possible attackers. Certain contexts should be treated as untrusted input, as an attacker could insert their own malicious content. For more information, see [Understanding the risk of script injections](#).

Property name	Type	Description
<code>github</code>	object	The top-level context available during any job or step in a workflow. This object contains all the properties listed below.
<code>github.action</code>	string	The name of the action currently running, or the <code>id</code> of a step. GitHub removes special characters, and uses the name <code>_run</code> when the current step runs a script without an <code>id</code> . If you use the <code>same_action</code> more than once in the same job, the name will

In this article

- About contexts
 - Determining when to use contexts
 - Context availability
 - Example: printing context information to the log
- github context
 - Example contents of the github context
 - Example usage of the github context
- env context
 - Example contents of the env context
 - Example usage of the env context
- job context
 - Example contents of the job context
 - Example usage of the job context
- jobs context

i.

1. **github.action** - The name of the action currently running.
2. **github.action_path** - The path where an action is located.
3. **github.actor** - The username of the user that triggered the initial workflow run.
4. **github.api_url** - The URL of the GitHub REST API.

ii. We can get complete information

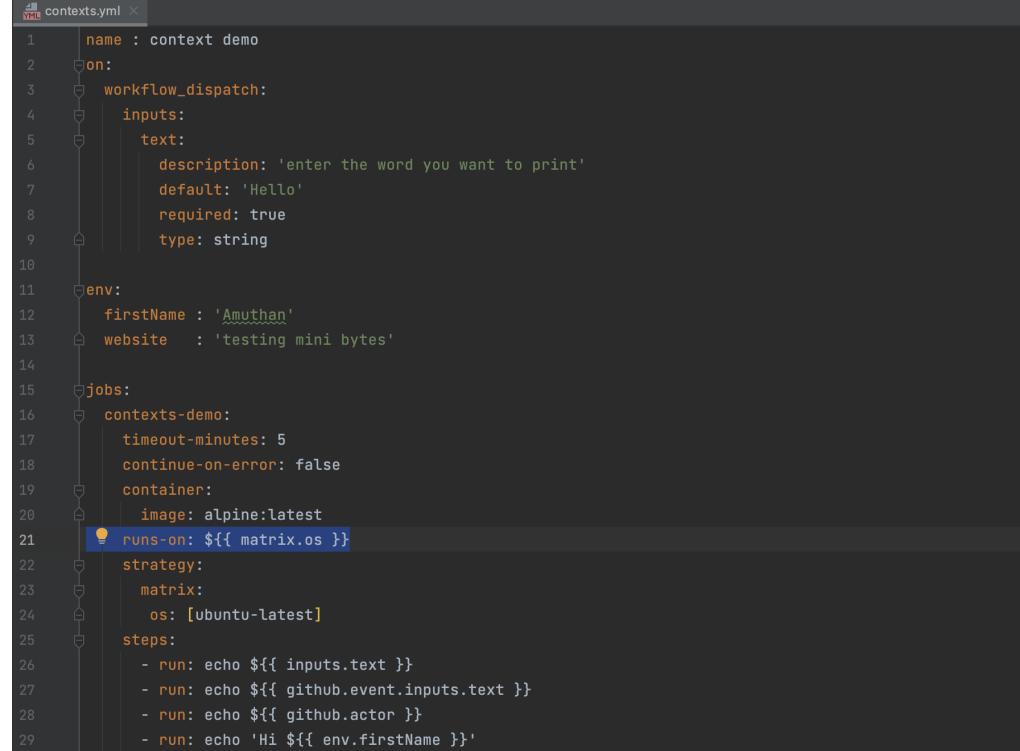
```
{
  "token": "***",
  "job": "dump_contexts_to_log",
  "ref": "refs/heads/my_branch",
  "sha": "c27d339ee6075c1f744c5d4b200f7901aad2c369",
  "repository": "octocat/hello-world",
  "repository_owner": "octocat",
  "repositoryUrl": "git://github.com/octocat/hello-world.git",
  "run_id": "1536140711",
  "run_number": "314",
  "retention_days": "90",
  "run_attempt": "1",
  "actor": "octocat",
  "workflow": "Context testing",
  "head_ref": "",
  "base_ref": "",
  "event_name": "push",
  "event": {
    ...
  },
  "server_url": "https://github.com",
  "api_url": "https://api.github.com",
  "graphql_url": "https://api.github.com/graphql",
  "ref_name": "my_branch",
  "ref_protected": false,
  "ref_type": "branch",
  "secret_source": "Actions",
  "workspace": "/home/runner/work/hello-world/hello-world",
  "action": "github_step",
  "event_path": "/home/runner/work/_temp/_github_workflow/event.json",
  "action_repository": "",
  "action_ref": "",
  "path": "/home/runner/work/_temp/_runner_file_commands/add_path_b037e7b5-1c88-48e2-bf7",
  "env": "/home/runner/work/_temp/_runner_file_commands/set_env_b037e7b5-1c88-48e2-bf7"
}
```

1.

1. More contexts

- a. env
 - b. job
 - c. steps
 - d. secrets
 - e. runner
-

1. contexts.yml



```
name : context demo
on:
  workflow_dispatch:
    inputs:
      text:
        description: 'enter the word you want to print'
        default: 'Hello'
        required: true
        type: string
env:
  firstName : 'Amuthan'
  website   : 'testing mini bytes'
jobs:
  contexts-demo:
    timeout-minutes: 5
    continue-on-error: false
    container:
      image: alpine:latest
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest]
    steps:
      - run: echo ${{ inputs.text }}
      - run: echo ${{ github.event.inputs.text }}
      - run: echo ${{ github.actor }}
      - run: echo 'Hi ${{ env.firstName }}'
```

- a.
- i. This Workflow can only be triggered manually
 1. Line #3 (**workflow_dispatch**)
 - ii. Also, it will ask the User for input (**text**)
 1. Line #4 (**inputs**)
 - a. It is a required parameter
 - i. Line #8 (**required: true**)
 - iii. 2 environments variables added
 1. Line #12 and 13 (firstName and website)
 2. These env variables will be applicable for all the jobs and steps mentioned in the .yml file
 - a. Jobs
 - i. context-demo
 - ii. Context-demo-2

3. Environment variables can be used at the jobs/steps level as well

 4. If an environment variable with the same name is provided at the job and step level,
 - a. Then, the preference will be given to the nearest one
 - i. In this case, to the Step level.

```

10
11     env:
12         firstName : 'Amuthan'
13         website   : 'testing mini bytes'
14
15     jobs:
16         contexts-demo:
17             timeout-minutes: 5
18             continue-on-error: false
19             container:
20                 image: alpine:latest
21                 runs-on: ${ matrix.os }
22             strategy:
23                 matrix:
24                     os: [ubuntu-latest]
25             steps:
26                 - run: echo ${ inputs.text }
27                 - run: echo ${ github.event.inputs.text }
28                 - run: echo ${ github.actor }
29                 - run: echo 'Hi ${ env.firstName }'
30                 - run: echo 'website - ${ env.website }'
31                 - run: echo ${ job.status }
32                 - run: echo ${ runner.os }
33                 - run: echo ${ secrets.PASSWORD } ${ env.firstName }
34             env:
35                 firstName : 'Testing'
36

```

2.

a. Line #21

- i. OS value is also passed from Matrix
 - 1. Refer to Lines #23 and 24
 - ii. The benefit of using matrix -
 - 1. We can parametrize the job
 - a. Job is running on one machine
 - i. os: [ubuntu-latest]
 - b. The job may run on multiple machines as well
 - i. os [ubuntu-latest, windows-latest]
-

3. Who has triggered this Workflow?

```

steps:
    - run: echo ${ inputs.text }
    - run: echo ${ github.event.inputs.text }
    - run: echo ${ github.actor }
    - run: echo 'Hi ${ env.firstName }'
    - run: echo 'website - ${ env.website }'
    - run: echo ${ job.status }
    - run: echo ${ runner.os }
    - run: echo ${ secrets.PASSWORD } ${ env.firstName }

```

a.

The screenshot shows the GitHub Actions interface. On the left, a sidebar lists 'Jobs' with four entries: 'contexts-demo (ubuntu-latest)' (green circle), 'contexts-demo-2' (green circle), 'job3' (green circle), and 'job4' (grey circle). Below this are 'Run details', 'Usage', and 'Workflow file'. The main area is titled 'contexts-demo (ubuntu-latest)' and shows a log entry for the 'Run echo rajatt95' step. The log output is as follows:

```

1 ▶ Run echo rajatt95
2 echo rajatt95
3 shell: sh -e {0}
4 env:
5   firstname: Amuthan
6   website: testing mini bytes
7 rajatt95

```

Below the log, there are several other step entries with status icons and execution times.

b.

- We did not provide the Step names in the .yml file
 - That's why we are getting the commands directly
 - Run echo rajatt95**

1. Setup Secrets

- <https://github.com/rajatt95/SeleniumAppiumTestsInGitHubRunners/settings/secrets/actions>
 - Go to the Github repository
 - Click on Secrets (under Security)
 - Click on Actions
 - Add one (**CUSTOM_SECRET_NAME**)

The screenshot shows the GitHub repository settings page under the 'Actions' tab. The left sidebar includes sections for General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets), and Actions. The 'Actions secrets' tab is selected, displaying environment and repository secrets. An environment secret named 'CUSTOM_SECRET_NAME' is listed, updated 13 seconds ago, with edit and delete icons.

v.

```

b. contexts.yml
25   steps:
26     - run: echo ${{ inputs.text }}
27     - run: echo ${{ github.event.inputs.text }}
28     - run: echo ${{ github.actor }}
29     - run: echo 'Hi ${{ env.firstName }}'
30     - run: echo 'website - ${{ env.website }}'
31     - run: echo ${{ job.status}}
32     - run: echo ${{ runner.os}}
33       - run: echo ${{ secrets.PASSWORD }} ${{ env.firstName }}
34     - run: echo ${{ secrets.CUSTOM_SECRET_NAME }} ${{ env.firstName }}
35   env:
36     firstName : 'Testing'

```

1. If you want a configuration where **1 job is dependent on another**

- a. We have that setup done here already

```

i. contexts-demo-2:
  needs: [contexts-demo]
  runs-on: ubuntu-latest
  steps:
    - run: echo 'contexts-demo-2 running after contexts-demo'

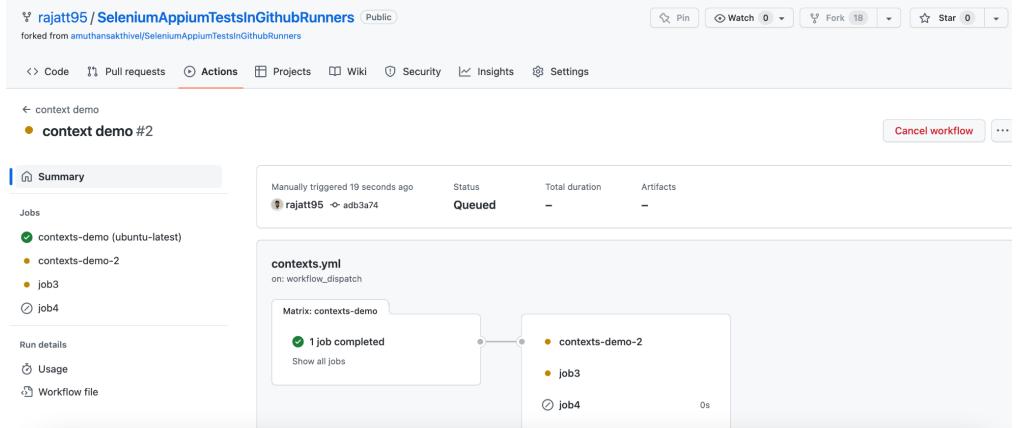
```

- 1. This job '**context-demo-2**' is dependent on '**context-demo**' job
- 2. These jobs will not run in parallel

2. Execute the Action

- a. <https://github.com/rajatt95/SeleniumAppiumTestsInGithubRunners/actions/workflows/contextes.yml>

i. Click on **Run workflow**



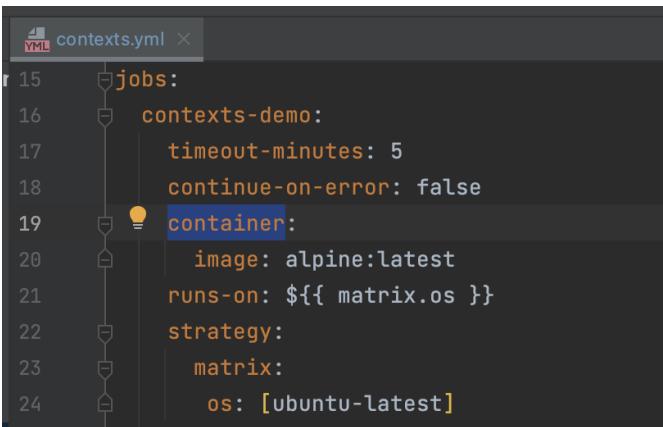
1.

- a. This Workflow will wait for the 1st job to complete
 - i. Once 1st job is completed, it will start the 2nd job

=====5_Part 5 | Jobs in GitHub Actions | Containers in Github Actions | Running Jobs in Parallel =====

1. A Workflow run is made up of one or more jobs, which run in parallel (Default)
 - a. To run jobs **sequentially**, you can define dependencies on other jobs using the **needs** keyword.
 2. We have the Liberty to run each job in a different host.
-

1. Jobs are the building blocks of Workflow
 2. **Workflow** is composed of **Jobs**
 - a. **Jobs** are composed of multiple **Steps**
-



```
YAML contexts.yml x
15   jobs:
16     contexts-demo:
17       timeout-minutes: 5
18       continue-on-error: false
19       container:
20         image: alpine:latest
21         runs-on: ${{ matrix.os }}
22       strategy:
23         matrix:
24           os: [ubuntu-latest]
```

1. **a. timeout-minutes** - Max. allowed time for the job execution
 - i. If any job is taking more than the time mentioned here, then, the job will get automatically canceled.
- b. container** - We can provide the specific version for execution
 - i. Example -
 1. I do not want to execute the automated tests on the Chrome browser which comes with ubuntu-latest machine
 2. Then, I can provide the version of the Browser
 - a. Also, we do not need to install Docker explicitly
 - i. Docker will be installed by default in the "ubuntu-latest" machine.
 - b. We only need to provide the **Image name**
 - c. We can define the **Ports** and **Volumes** as well for the Image

1. The job should **always run**

- a. Irrespective of the execution result of the job (contexts-demo)

```
job3:  
  runs-on: ubuntu-latest  
  if: ${{ always() }}  
  needs: [ contexts-demo ]  
  steps:  
    - run: echo 'job3'
```

b.

1. Job should **run Only-On-Failure** of job (contexts-demo)

```
job4:  
  runs-on: ubuntu-latest  
  if: ${{ failure() }}  
  needs: [ contexts-demo ]  
  steps:  
    - run: echo 'job4'
```

a.

- i. We have one parameter added in context-demo job

1. **continue-on-error**

```
jobs:  
  contexts-demo:  
    timeout-minutes: 5  
    continue-on-error: false  
    container:  
      image: alpine:latest  
      runs-on: ${{ matrix.os }}
```

a.

- i. If this is marked as true, then, the result for this job will always be **PASSED**

1. It will never be marked as **FAIL**

- a. That means Job4 will never execute.

The screenshot shows a GitHub Actions run summary for a workflow named 'context demo'. The run was triggered manually 1 hour ago by user 'rajatt95' and completed successfully in 27 seconds. The summary page lists four jobs under the 'contexts-demo' matrix:

- contexts-demo (ubuntu-latest)**: Passed
- contexts-demo-2**: Passed
- job3**: Passed
- job4**: Skipped

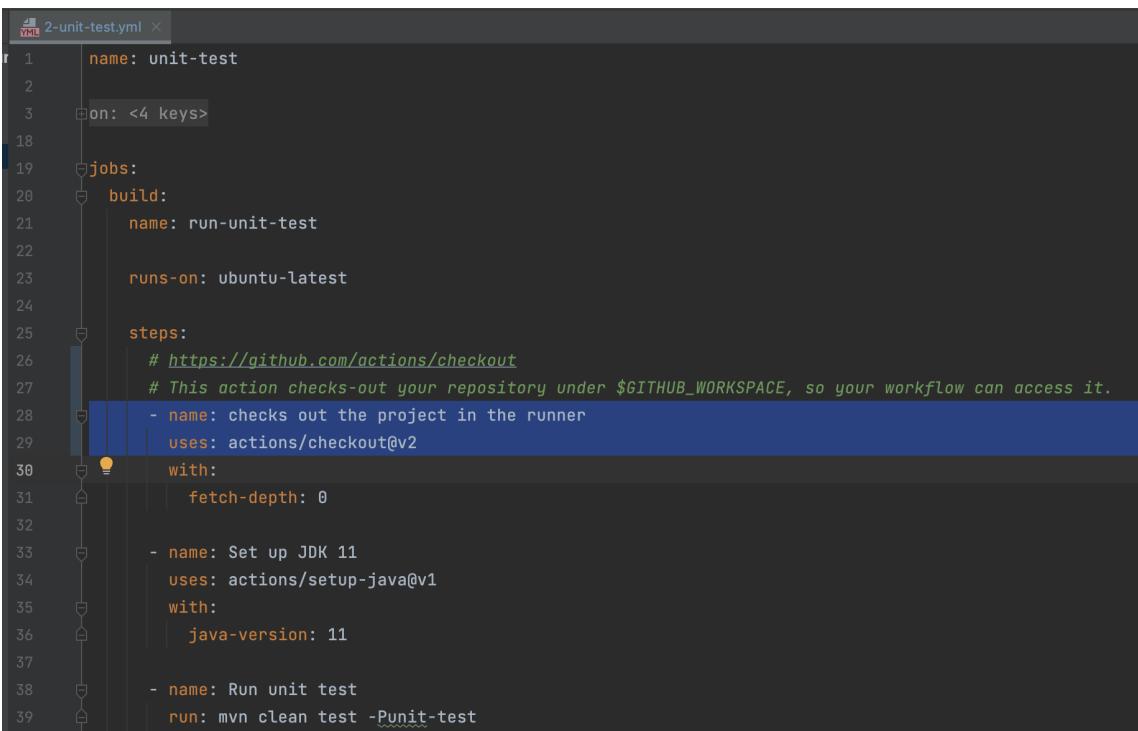
A tooltip for the 'Matrix: contexts-demo' section indicates that 1 job completed. The tooltip also includes a link to 'Show all jobs'.

- 2.
- job4 is skipped
 - Why?
 - Because the result of the **job** (contexts-demo) is **passed**.
 - job4 will execute only** when the result of the job (contexts-demo) is **failed**.
-

=====6_Part 6 - Steps in Github Actions 🔥 | Step Failure | Timeout for a Step | Github Actions Tutorial |=====

1. Steps

- a. Jobs are composed of one or more Steps
 - b. We can use the already existing Actions
 - i. From Github Action Marketplace
 - c. Or
 - i. We can create customized Actions
-



```
1  name: unit-test
2
3  on: <4 keys>
4
5  jobs:
6    build:
7      name: run-unit-test
8
9      runs-on: ubuntu-latest
10
11    steps:
12      # https://github.com/actions/checkout
13      # This action checks-out your repository under $GITHUB_WORKSPACE, so your workflow can access it.
14      - name: checks out the project in the runner
15        uses: actions/checkout@v2
16        with:
17          fetch-depth: 0
18
19      - name: Set up JDK 11
20        uses: actions/setup-java@v1
21        with:
22          java-version: 11
23
24      - name: Run unit test
25        run: mvn clean test -Punit-test
```

1. a. Steps

- i. checks out the project in the runner
 - ii. Set up JDK 11
 - iii. Run unit test
- b. If the 1st step gets failed due to some issue
 - i. Then, 2nd and 3rd steps will get skipped.
 - ii. To change this behavior (If you think, this step is not very important)
 - 1. Add one property with the step
 - a. **continue-on-error: true**
 - c. Other parameters may be added
 - i. **timeout-minutes**

```

ii. - name: Run unit test
      # If pom.xml file is located inside runner directory/folder
      working-directory: /runner
      # If you want to execute the command in Powershell
      shell: powershell
      # Execute multiple Maven commands
      run: |
        mvn -v
        mvn clean test -Punit-test

```

=====7_Part 7 | Running web test in Github Actions | Setup Selenoid in Github Actions | Passing secrets |=====

1. Running Web Tests

- Choose the Trigger
 - Choose the Runner
 - Setup the Project
 - Setup Java
 - Setup Selenoid Container
 - Run the tests by passing username and password
-

```

1. package com.web;
2
3 import ...
11
12 > public class WebTest {
13
14     @SneakyThrows
15     @Test(groups = "web")
16     public void testGoogleSearchUsingSelenoid() {
17         System.out.println("username :" + System.getProperty("username"));
18         System.out.println("password :" + System.getProperty("password"));
19
20         DesiredCapabilities capabilities = new DesiredCapabilities();
21         capabilities.setCapability( capabilityName: "browserName", value: "chrome");
22         capabilities.setCapability( capabilityName: "enableVNC", value: true);
23         capabilities.setCapability( capabilityName: "enableVideo", value: false);
24         WebDriver driver = new RemoteWebDriver(new URL( spec: "http://localhost:4444/wd/hub"),capabilities);
25         driver.get("https://google.co.in");
26         Assert.assertEquals(driver.getTitle(), expected: "Google");
27         driver.quit();
28     }
29 }

```

1. demo-web.yml

```
demo-web.yml ×
1  # Name of the Workflow
2  name: run web test in github actions
3
4  # To Trigger the Workflow manually
5  on: workflow_dispatch
6  jobs:
7    # run-web-test is the name of the Job
8    run-web-test:
9      # ubuntu-latest (machine on which Test case will get executed)
10     runs-on: ubuntu-latest
11
12    steps:
13      - name: pull the project in to the runner
14        uses: actions/checkout@v3
15
16      - name: set up java
17        uses: actions/setup-java@v3
18        with:
19          distribution: 'temurin'
20          java-version: '11'
21
22      - name: store docker image cache
23        uses: satakey/action-docker-layer-caching@v0.0.11
24        continue-on-error: true
25
26      - name: Start Selenoid server
27        uses: n-ton4/selenoid-github-action@master
28        id: start-selenoid
29        continue-on-error: false
```

2. Add a Secret

- a. Go to Code repository
- b. Go to Settings
- c. Click on Secrets (Under Security)
- d. Click on Actions
- e. New Repository Secret

Code Pull requests Actions Projects Wiki Security Insights Settings

General Actions secrets New repository secret

Access Collaborators Moderation options

Code and automation Branches Tags Actions Webhooks Environments Codespaces Pages

Security Code security and analysis Deploy keys Secrets

Environment secrets Manage environments

There are no secrets for this repository's environments.

Repository secrets

CUSTOM_SECRET_NAME Updated 5 hours ago

CUSTOM_SECRET_USERNAME Updated 1 minute ago

f. Actions

```

- name: run the web test
  run: |
    mvn clean test -Pweb -Dusername=${{secrets.USERNAME}}
    mvn clean test -Pweb -Dusername=${{secrets.CUSTOM_SECRET_USERNAME}}
  
```

3.

demo-web.yml

```

# Name of the Workflow
name: run web test in github actions

# To Trigger the Workflow manually
on: workflow_dispatch

jobs:
  # run-web-test is the name of the Job
  run-web-test:
    # ubuntu-latest (machine on which Test case will get executed)
    runs-on: ubuntu-latest

    steps:
      - name: pull the project in to the runner
        uses: actions/checkout@v3

      # This is an Optional Step
      # Why? - Because the machine (ubuntu-latest) has many softwares installed already
      # https://github.com/actions/runner-images
      - name: set up java
        uses: actions/setup-java@v3
        with:
          distribution: 'temurin'
          java-version: '11'
  
```

1.

- a. <https://github.com/actions/runner-images>
- b. Ubuntu-Latest
 - i. <https://github.com/actions/runner-images/blob/main/images/linux/Ubuntu2204-Readme.md>

Java

Version	Vendor	Environment Variable
8.0.352+8	Eclipse Temurin	JAVA_HOME_8_X64
11.0.17+8 (default)	Eclipse Temurin	JAVA_HOME_11_X64
17.0.5+8	Eclipse Temurin	JAVA_HOME_17_X64

ii.

=====8_Part 8 | Reduce workflow time | Caching Maven | Caching Docker Image Layers | Github Actions=====

1. So far,
 - a. All the dependency is getting downloaded every time when we are triggering the Workflow (Manual/Automated)
 - i. Start Selenoid Server - **32 seconds**
 - 1. **Why is it taking so long?**
 - a. Because every time, we are pulling the latest Docker image for the Selenoid server.
 - b. **Job time can be drastically reduced by caching the dependencies or docker image layers.**

1. Maven Cache Github Action

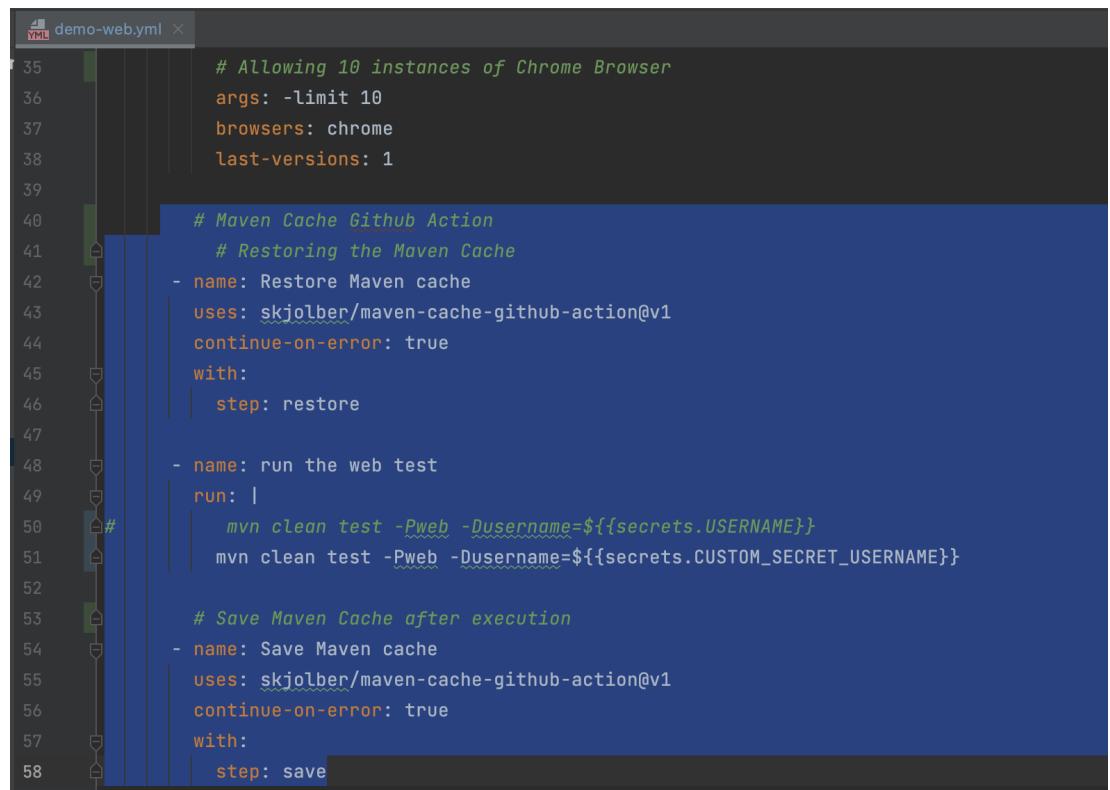
- <https://github.com/marketplace/actions/maven-cache>

Usage

The `skjolber/maven-cache-github-action` action must be present **twice** in your build job, with `step: restore` and `step: save` parameters:

```
jobs:  
  hello_world_job:  
    runs-on: ubuntu-latest  
    name: Maven build with caching  
    steps:  
      - name: Checkout  
        uses: actions/checkout@v2  
      - name: Set up JDK 1.8  
        uses: actions/setup-java@v1  
        with:  
          java-version: 1.8  
      - name: Restore Maven cache  
        uses: skjolber/maven-cache-github-action@v1  
        with:  
          step: restore  
      - name: Build hello-world application with Maven  
        run: mvn --batch-mode --update-snapshots verify  
      - name: Save Maven cache  
        uses: skjolber/maven-cache-github-action@v1  
        with:  
          step: save
```

i.



```
YAML demo-web.yml ×  
 35   # Allowing 10 instances of Chrome Browser  
 36   args: -limit 10  
 37   browsers: chrome  
 38   last-versions: 1  
 39  
 40   # Maven Cache Github Action  
 41   # Restoring the Maven Cache  
 42   - name: Restore Maven cache  
 43     uses: skjolber/maven-cache-github-action@v1  
 44     continue-on-error: true  
 45     with:  
 46       step: restore  
 47  
 48   - name: run the web test  
 49     run: |  
 50       mvn clean test -Pweb -Dusername=${{secrets.USERNAME}}  
 51       mvn clean test -Pweb -Dusername=${{secrets.CUSTOM_SECRET_USERNAME}}  
 52  
 53   # Save Maven Cache after execution  
 54   - name: Save Maven cache  
 55     uses: skjolber/maven-cache-github-action@v1  
 56     continue-on-error: true  
 57     with:  
 58       step: save
```

b.

- For the 1st time, it will take time to download and cache it

1. And from 2nd time, it will use the dependencies downloaded in 1st-time

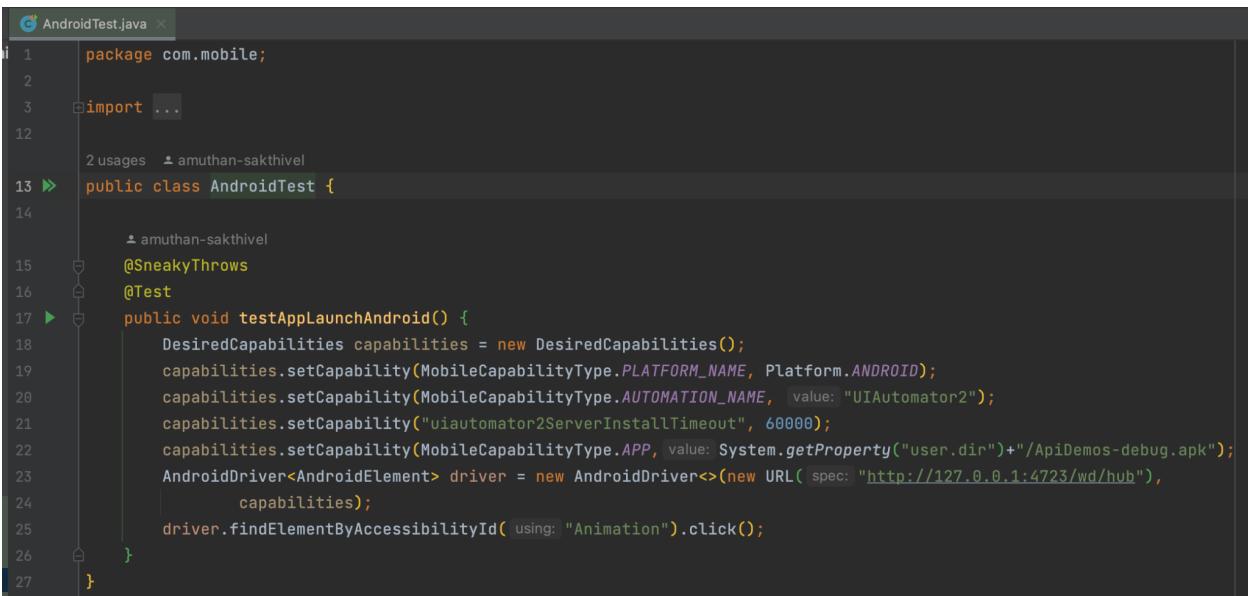
C.

```
demo-web.yml
1 # Name of the Workflow
2 name: run web test in github actions
3
4 # To Trigger the Workflow manually
5 on: workflow_dispatch
6 jobs:
7   run-web-test:
8     # ubuntu-latest (machine on which Test case will get executed)
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: pull the project in to the runner
13       uses: actions/checkout@v3
14
15     # This is an Optional Step
16     # Why? - Because the machine (ubuntu-latest) has many softwares installed already
17     # https://github.com/actions/runner-images
18     - name: set up java
19       uses: actions/setup-java@v3
20       with:
21         distribution: 'temurin'
22         java-version: '11'
23
24     - name: store docker image cache
25       uses: satackey/action-docker-layer-caching@v0.0.11
26       continue-on-error: true
27
```

-
1. Earlier - Start Selenoid Server - **32 seconds**
 2. Now - Start Selenoid Server - **5 seconds**
-

=====9_Part 9 | No Cloud Providers Needed 🔥 | Run Android Appium Tests inside Github Runner 🔥 =====

1. We can run the Android tests in Github Runner itself
 - a. Setup project
 - b. Setup Java
 - c. Setup Node
 - d. Download Appium
 - e. Run Appium
 - f. Start Emulator
 - g. Run the test
-



```
1 package com.mobile;
2
3 import ...
12
13 public class AndroidTest {
14
15     @SneakyThrows
16     @Test
17     public void testAppLaunchAndroid() {
18         DesiredCapabilities capabilities = new DesiredCapabilities();
19         capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, Platform.ANDROID);
20         capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, value: "UIAutomator2");
21         capabilities.setCapability("uiautomator2ServerInstallTimeout", 60000);
22         capabilities.setCapability(MobileCapabilityType.APP, value: System.getProperty("user.dir")+"/"+ApiDemos-debug.apk");
23         AndroidDriver<AndroidElement> driver = new AndroidDriver<>(new URL( spec: "http://127.0.0.1:4723/wd/hub"),
24             capabilities);
25         driver.findElementByAccessibilityId( using: "Animation").click();
26     }
27 }
```

-
1. <https://github.com/actions/runner-images/blob/main/images/macos/macos-12-Readme.md>
-

```
demo-android.yml ×
1 # Name of the Workflow
2   name: run android tests in github runner
3
4   on:
5     push:
6
7       # To Trigger the Workflow manually
8       workflow_dispatch:
9
10  jobs:
11    # run-mobile-tests is the name of the Job
12    run-mobile-tests:
13      # macos-latest (machine on which Test case will get executed)
14      runs-on: macos-latest
15
16      steps:
17        # This is an Optional Step
18        # Why? - Because the machine (macos-latest) has many softwares installed already
19        # Java 8 is the default version
20        # https://github.com/actions/runner-images
21        - name: setup java
22          uses: actions/setup-java@v3
23          with:
24            distribution: 'temurin'
25            java-version: '11'
26
27        - name: pull the project in to the runner
28          uses: actions/checkout@v3
```

```
demo-android.yml ×
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

1. <https://github.com/marketplace/actions/android-emulator-runner>

Marketplace / Actions / Android Emulator Runner

 GitHub Action
Android Emulator Runner
v2.27.0 [Latest version](#)

[Use latest version](#)

GitHub Action - Android Emulator Runner

[Main workflow](#) [passing](#)

A GitHub Action for installing, configuring and running hardware-accelerated Android Emulators on macOS virtual machines.

The old ARM-based emulators were slow and are no longer supported by Google. The modern Intel Atom (x86 and x86_64) emulators can be fast, but rely on two forms of hardware acceleration to reach their peak potential: [Graphics Acceleration](#), e.g. `emulator -gpu host` and [Virtual Machine\(VM\) Acceleration](#), e.g. `emulator -accel`. Note: GPU and VM Acceleration are two different and non-mutually exclusive forms of Hardware Acceleration.

This presents a challenge when running emulators on CI especially when running emulators within a docker container, because [Nested Virtualization](#) must be supported by the host VM

a.

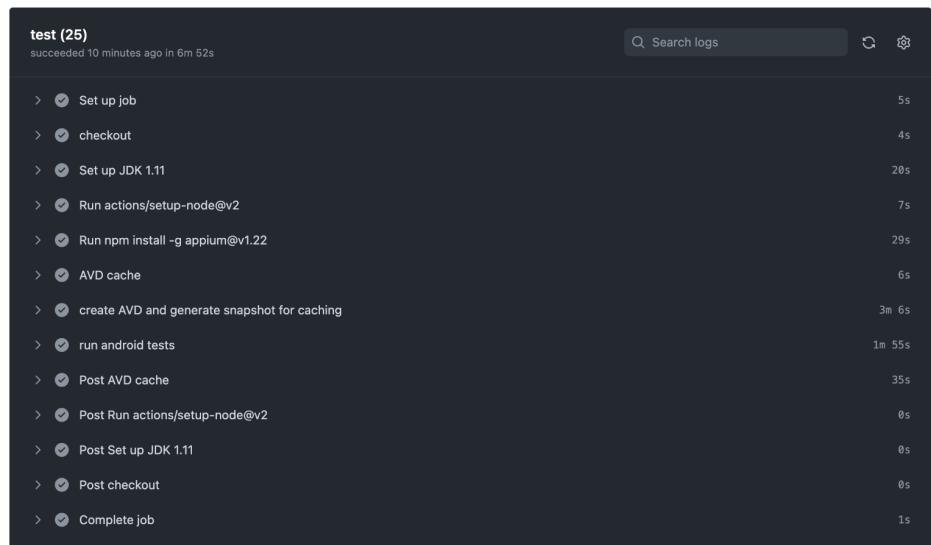
Stars: [Star 651](#)

Contributors: 

Categories: [Testing](#) [Mobile CI](#)

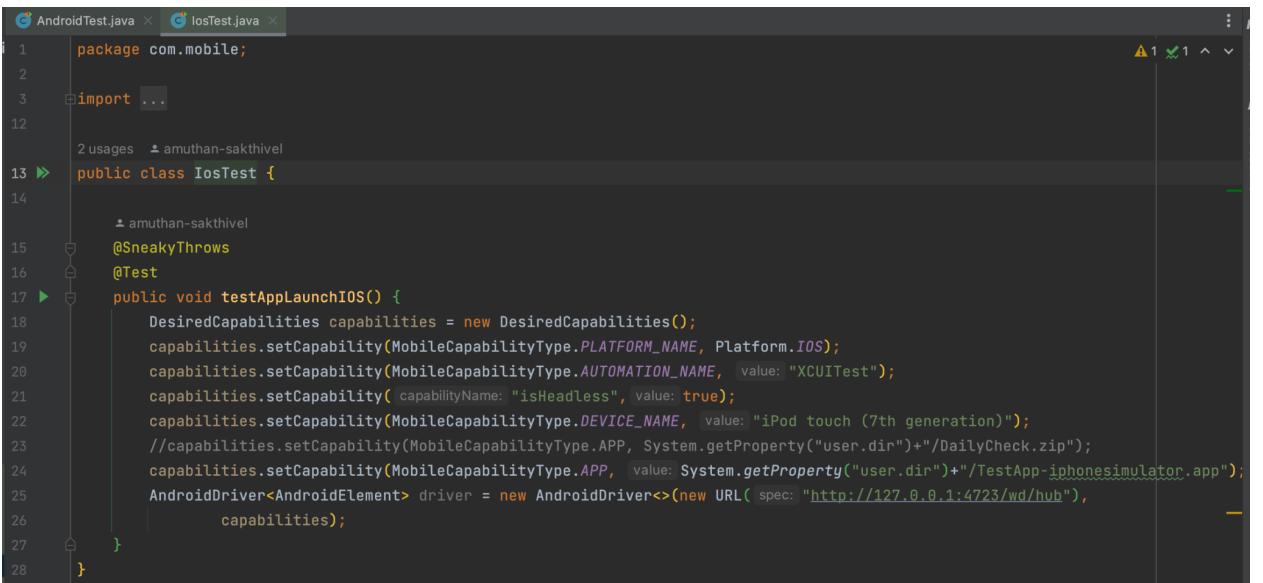
Links: [ReactiveCircus/android-emulator-runner](#)

1.



=====10_Part 10 | Run iOS Appium Tests in Github Runner 🔥 | No cloud providers needed |=====

1. We can run the iOS tests in the Github Runner itself
 - a. Setup Java
 - b. Setup Node
 - c. Download Appium
 - d. Run Appium
 - e. Start Emulator
 - f. Run the test
-



The screenshot shows a code editor with two tabs: 'AndroidTest.java' and 'IosTest.java'. The 'IosTest.java' tab is active, displaying the following Java code:

```
1 package com.mobile;
2
3 import ...
4
5 2 usages ▾ amuthan-sakthivel
6
7 ➤ public class IosTest {
8
9     ▾ amuthan-sakthivel
10    @SneakyThrows
11    @Test
12    public void testAppLaunchIOS() {
13        DesiredCapabilities capabilities = new DesiredCapabilities();
14        capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, Platform.IOS);
15        capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, value: "XCUITest");
16        capabilities.setCapability(capabilityName: "isHeadless", value: true);
17        capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, value: "iPod touch (7th generation)");
18        //capabilities.setCapability(MobileCapabilityType.APP, System.getProperty("user.dir")+"/DailyCheck.zip");
19        capabilities.setCapability(MobileCapabilityType.APP, value: System.getProperty("user.dir")+"/TestApp-iphonesimulator.app");
20        AndroidDriver<AndroidElement> driver = new AndroidDriver<>(new URL(spec: "http://127.0.0.1:4723/wd/hub"),
21                                capabilities);
22    }
23
24 }
```

```
demo-ios.yml
1  name: run ios tests in github runner
2
3  on: push
4
5  jobs:
6    run-ios-tests:
7      runs-on: macos-latest
8
9    steps:
10      - name: setup java
11        uses: actions/setup-java@v3
12        with:
13          distribution: 'temurin'
14          java-version: '11'
15
16      - name: pull the project in to the runner
17        uses: actions/checkout@v3
18
19      - name: setup appium
20        run:
21          npm install -g appium@v1.22
22          appium -v
23          appium &>/dev/null &
24
25      - name: run appium ios tests
26        run: mvn clean test -Pios
```

2.

← Run appium iOS test in Github Runner

Run appium iOS test in Github Runner #1

Re-run all jobs ...

Summary

Jobs

build

build succeeded 1 hour ago in 7m 9s

Set up job 4s

Run actions/checkout@v2 2s

Set up JDK 1.11 8s

Run actions/setup-node@v2 0s

Run npm install -g appium@v1.22 6m 51s

Post Run actions/setup-node@v2 0s

Post Set up JDK 1.11 0s

Post Run actions/checkout@v2 1s

Complete job 1s

1.

=====11_Part 11 | Trigger Automated Test workflow in one repo from Development Workflow in another repo=====

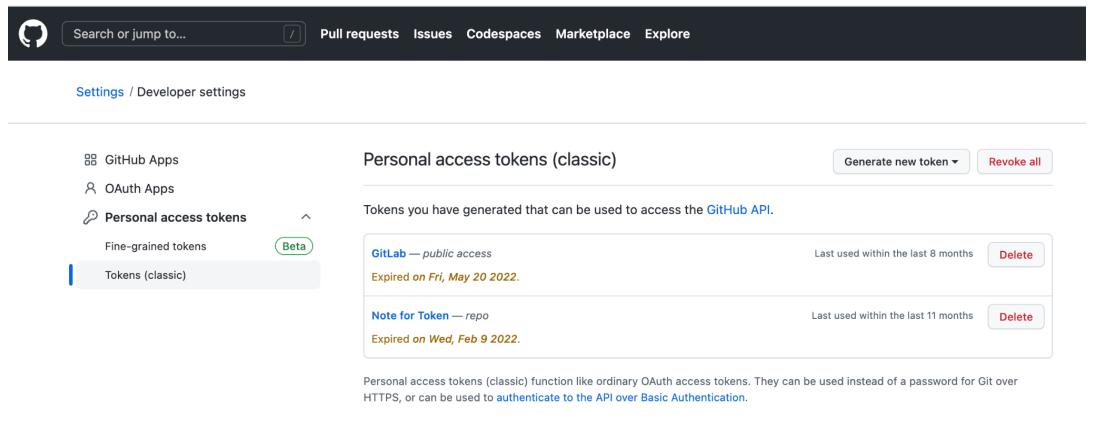
1. Triggering a Workflow in one repository from another repository using Repository dispatch
 - a. Example
 - i. We want to **run our automated tests** when there is a **successful merge in the development branch**

1. <https://github.com/marketplace/actions/repository-dispatch>

1. To generate Personal Token

- a. Go to Github Profile
- b. Go to Settings
- c. Go to Developer Settings
- d. Go to Personal Access Tokens
- e. Click on **Generate new Token**

i. <https://github.com/settings/tokens>

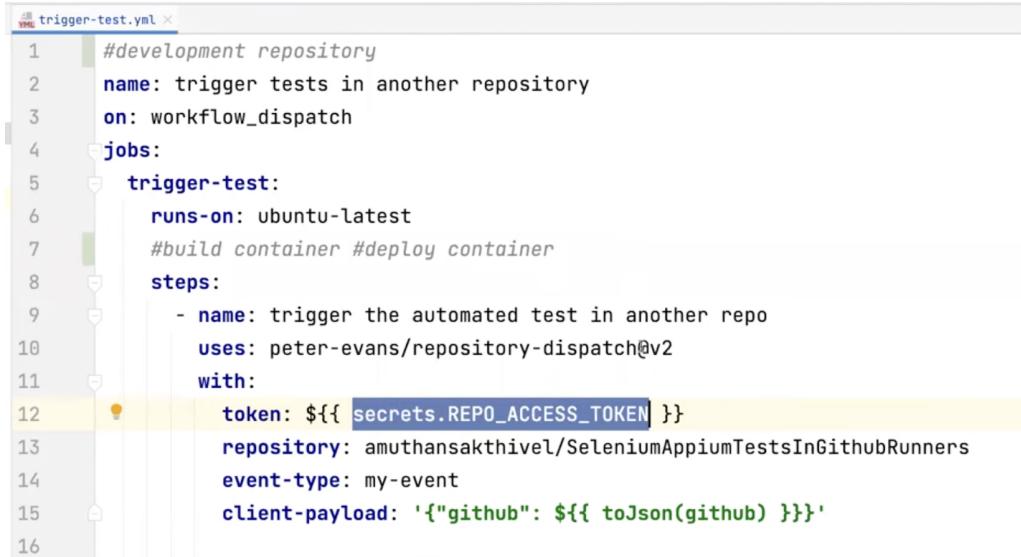


The screenshot shows the GitHub Developer settings page. The left sidebar has sections for GitHub Apps, OAuth Apps, Personal access tokens (classic), Fine-grained tokens (Beta), and Tokens (classic). The Personal access tokens (classic) section is selected. It displays a table of tokens:

Token Name	Scope	Last Used	Action
GitLab — public access		Within the last 8 months	Delete
Note for Token — repo		Within the last 11 months	Delete

Below the table, a note states: "Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication."

ii.

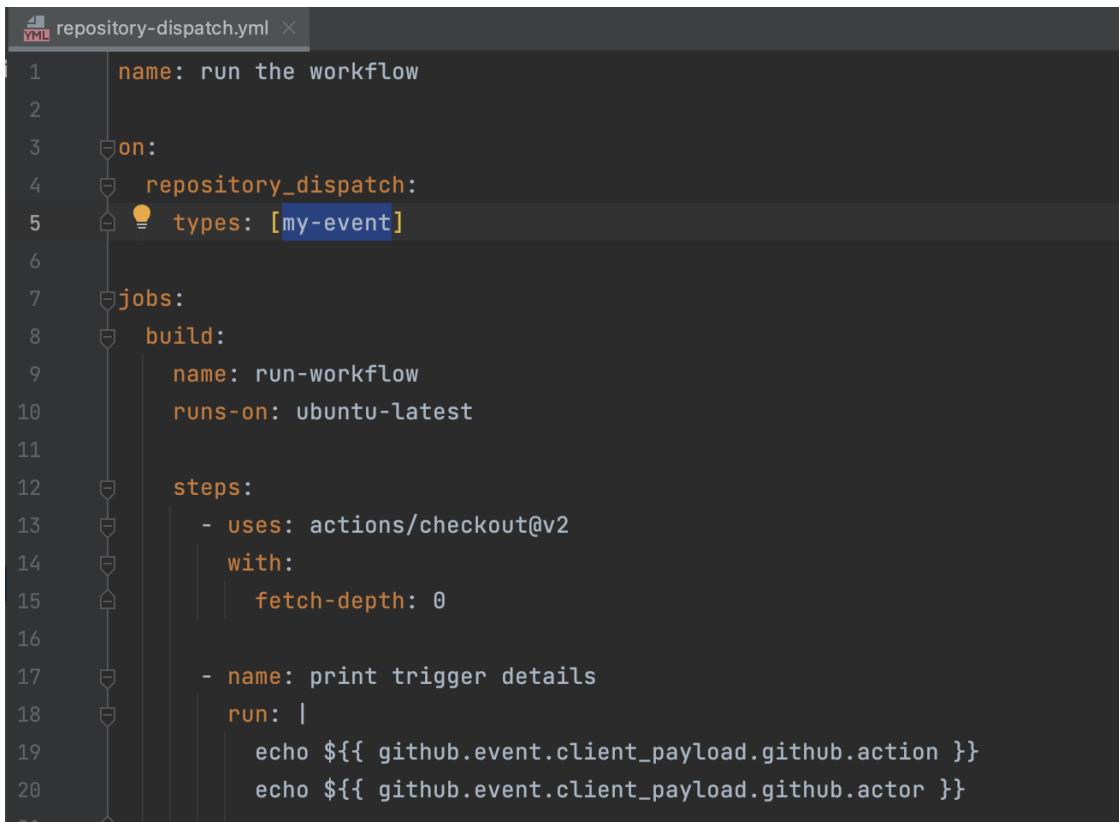


```
trigger-test.yml
1 #development repository
2   name: trigger tests in another repository
3   on: workflow_dispatch
4   jobs:
5     trigger-test:
6       runs-on: ubuntu-latest
7       #build container #deploy container
8       steps:
9         - name: trigger the automated test in another repo
10           uses: peter-evans/repository-dispatch@v2
11           with:
12             token: ${{ secrets.REPO_ACCESS_TOKEN }}
13             repository: amuthansakthivel/SeleniumAppiumTestsInGithubRunners
14             event-type: my-event
15             client-payload: '{"github": ${ toJson(github) }}'
16
```

1.

a. trigger-test.yml file

- This is for the development code.
 - Code push to develop branch -> Merge success will lead to the execution of automated tests (another repository)



```
repository-dispatch.yml
1   name: run the workflow
2
3   on:
4     repository_dispatch:
5       types: [my-event]
6
7   jobs:
8     build:
9       name: run-workflow
10      runs-on: ubuntu-latest
11
12      steps:
13        - uses: actions/checkout@v2
14          with:
15            fetch-depth: 0
16
17        - name: print trigger details
18          run:
19            echo ${{ github.event.client_payload.github.action }}
20            echo ${{ github.event.client_payload.github.actor }}
```

2.

a. repository-dispatch.yml

- This file is used to execute the automated tests.

1. More documents:

- a. <https://github.com/rajatt95/Documents>

2. Learnings from Tutor (Code Repository):

- a. This course

- i. <https://github.com/stars/rajatt95/lists/youtube-tmb-github-actions>

- b. Other course(s):

- i. <https://github.com/stars/rajatt95/lists/testing-mini-bytes-amuthan>

- c.

3. To connect:

- a. <https://www.linkedin.com/in/rajat-v-3b0685128/>
- b. <https://github.com/rajatt95>
- c. <https://rajatt95.github.io/>

THANK YOU!

